Department of Information and Computer Science

Learning Constructions of Natural Language: Statistical Models and Evaluations

Sami Virpioja

Aalto University

DOCTORAL DISSERTATIONS

Aalto University publication series
DOCTORAL DISSERTATIONS 158/2012

Learning Constructions of Natural Language: Statistical Models and Evaluations

Sami Virpioja

A doctoral dissertation completed for the degree of Doctor of Science in Technology to be defended, with the permission of the Aalto University School of Science, at a public examination held at the lecture hall TU1 of the school on 10 December 2012 at 12.

Aalto University
School of Science
Department of Information and Computer Science

Supervising professor
Prof. Erkki Oja

Thesis advisors
Doc. Mikko Kurimo
Dr. Krista Lagus

Preliminary examiners
Doc. Krister Lindén, University of Helsinki, Finland
Prof. Richard Wicentowski, Swarthmore College, USA

Opponents
Doc. Krister Lindén, University of Helsinki, Finland
Prof. Brian Roark, Oregon Health & Science University, USA

Aalto University publication series
DOCTORAL DISSERTATIONS 158/2012

ISBN 978-952-60-4882-6 (printed)
ISBN 978-952-60-4883-3 (pdf)
ISSN-L 1799-4934
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)
http://urn.fi/URN:ISBN:978-952-60-4883-3

Unigrafia Oy
Helsinki 2012

Finland

441 697
Printed matter

Aalto University

Abstract

Aalto University, P.O. Box 11000, FI-00076 Aalto www.aalto.fi

Author
Sami Virpioja

Name of the doctoral dissertation
Learning Constructions of Natural Language: Statistical Models and Evaluations

Publisher School of Science

Unit Department of Information and Computer Science

Series Aalto University publication series DOCTORAL DISSERTATIONS 158/2012

Field of research Computer and Information Science

Manuscript submitted 12 June 2012 **Date of the defence** 10 December 2012

Permission to publish granted (date) 4 October 2012 **Language** English

☐ **Monograph** ☒ **Article dissertation (summary + original articles)**

Abstract
The modern, statistical approach to natural language processing relies on using machine learning techniques on the increasing amount of text and speech data in electronic format. Typical applications for statistical methods include information retrieval, speech recognition, and machine translation. Many problems encountered in the applications can be solved without language-dependent resources, such as annotated data sets, by the means of unsupervised learning. This thesis focuses on one such problem: the selection of lexical units. It is the first step in processing text data, preceding, for example, the estimation of language models or extraction of vectorial representations. While the lexical units are often selected using simple heuristics or grammatical rule-based methods, this thesis proposes the use of unsupervised and semi-supervised machine learning. Advantages of the data-driven unit selection include greater flexibility and independence from the linguistic resources that exist for a particular language and domain.

Statistically learned lexical units do not always fit to the categories in traditional linguistic theories. In this thesis, they are called constructions according to construction grammars, a family of usage-based, cognitive theories of grammar. For learning constructions of a language, the thesis builds on Morfessor, an unsupervised statistical method for morphological segmentation. Morfessor is successfully extended to the tasks of learning allomorphs, semi-supervised learning of morphological segmentation, and learning phrasal constructions of sentences. The results are competitive especially for the morphology induction problems. The thesis also includes new techniques for using the sub-word constructions learned by Morfessor in statistical language modeling and machine translation. In addition to its usefulness in the applications, Morfessor is shown to have psycholinguistic competence: its probability estimates have high correlations with human reaction times in a lexical decision task.

Furthermore, direct evaluation methods for the unit selection and other learning problems are considered. Direct evaluations, such as comparing the output of the algorithm to existing linguistic annotations, are often quicker and simpler than indirect evaluation via the end-user applications. However, with unsupervised algorithms, the comparison to the reference data is not always straightforward. In this thesis, direct evaluation methods are developed for two unsupervised tasks, morphology induction and learning semantic vector representations of documents. In both cases, the challenge is to find relationships between the pairs of features in multidimensional data. The proposed methods are quick to use and they can accurately predict the performance in different applications.

Keywords morpheme segmentation, morphology induction, construction grammar, unsupervised learning, semi-supervised learning, probabilistic models, language models, vector space models, machine translation, speech recognition

ISBN (printed) 978-952-60-4882-6 **ISBN (pdf)** 978-952-60-4883-3

ISSN-L 1799-4934 **ISSN (printed)** 1799-4934 **ISSN (pdf)** 1799-4942

Location of publisher Espoo **Location of printing** Helsinki **Year** 2012

Pages 437 **urn** http://urn.fi/URN:ISBN:978-952-60-4883-3

A! Aalto-yliopisto

Tiivistelmä

Aalto-yliopisto, PL 11000, 00076 Aalto www.aalto.fi

Tekijä
Sami Virpioja

Väitöskirjan nimi
Luonnollisen kielen rakenteiden oppiminen: tilastollisia malleja ja evaluaatiomenetelmiä

Julkaisija Perustieteiden korkeakoulu

Yksikkö Tietojenkäsittelytieteen laitos

Sarja Aalto University publication series DOCTORAL DISSERTATIONS 158/2012

Tutkimusala Informaatiotekniikka

Käsikirjoituksen pvm 12.06.2012 **Väitöspäivä** 10.12.2012

Julkaisuluvan myöntämispäivä 04.10.2012 **Kieli** Englanti

☐ **Monografia** ☒ **Yhdistelmäväitöskirja (yhteenveto-osa + erillisartikkelit)**

Tiivistelmä
Luonnollisen kielen automaattinen käsittely pohjautuu yhä suuremmassa määrin tilastollisten koneoppimismenetelmien käyttöön jatkuvasti lisääntyvälle elektroniselle teksti- ja puheaineistolle. Tyypillisiä sovelluksia tilatollisille menetelmille ovat esimerkiksi tiedonhaku, puheentunnistus ja konekäännös. Monet sovellusten osaongelmat ovat ratkaistavissa ilman kieliriippuvaisia resursseja, kuten annotoituja aineistoja, käyttämällä ohjaamatonta koneoppimista. Tämä väitöskirja keskittyy erityisesti yhteen tällaiseen ongelmaan: leksikaalisten perusyksiköiden valintaan. Käytettävien yksiköiden valinta on tekstiaineiston käsittelyn ensimmäinen askel ja edeltää esimerkiksi kielimallien estimointia tai vektoriesitysten laskemista. Perinteisiä ratkaisuja yksiköiden valintaan ovat yksinkertaiset heuristiikat sekä kieliopilliset sääntöpohjaiset työkalut. Niiden sijaan tässä työssä esitetään datalähtöistä, ohjaamattomaan oppimiseen perustuvaa lähestymistapaa yksiköiden valintaan. Sen etuina ovat joustavuus ja riippumattomuus siitä, mitä lingvistisiä resursseja halutulle kielelle ja sovellusalueelle on saatavilla.

Koska tilastollisesti opitut yksiköt eivät aina osu yhteen perinteisten kielitieteellisten perusluokkien kanssa, niitä kutsutaan tässä työssä konstruktioiksi. Termi pohjautuu konstruktiokielioppeihin, jotka ovat käyttöpohjaisia, kognitiivisia teorioita kielestä. Väitöskirjassa esitetyt menetelmät konstruktioiden oppimiseen perustuvat Morfessor-nimiseen menetelmään, joka mallintaa morfologista pilkontaa tilastollisesti ja ohjaamattomasti. Uudet menetelmät käsittelevät allomorfian oppimista, morfologisen pilkonnan osittain ohjattua oppimista sekä lausetason konstruktioiden oppimista. Saadut tulokset ovat kilpailukykyisiä erityisesti morfologian oppimisessa. Työssä esitellään myös uusia tekniikoita Morfessorin tuottamien morfologisten konstruktioiden käyttöön tilastollisessa kielenmallinnuksessa ja konekäännöksessä. Käytännön sovellusten ohella Morfessorin osoitetaan toimivan myös psykolingvistisen datan mallinnuksessa: sen todennäköisyysestimaatit sanoille korreloivat vahvasti ihmisten reaktioaikoihin leksikaalisessa päätöksenteossa.

Lisäksi väitöskirjassa tutkitaan kielen ohjaamattoman oppimisen suoria evaluaatiomenetelmiä. Suora evaluaatio, esimerkiksi algoritmin tulosten vertaaminen olemassa oleviin kieliopillisiin annotaatioihin, on usein nopeampaa ja yksinkertaisempaa kuin epäsuora evaluaatio kielenkäsittelyn sovellusten toiminnan kautta. Ohjaamattoman oppimisen tapauksessa vertailu annotoituun dataan ei kuitenkaan aina ole suoraviivaista. Tässä väitöskirjassa kehitetään evaluaatiomenetelmiä erityisesti kahteen ongelmaan: sanojen morfologian oppimiseen ja vektorimuotoisten dokumenttiesitysten oppimiseen. Molemmissa on haasteena löytää moniulotteisesta datasta yhteydet eri piirreparien välille. Ehdotetut menetelmät ovat nopeita käyttää ja ne ennustavat hyvin sovelluksista saatuja tuloksia.

Avainsanat morfeemipilkonta, morfologian oppiminen, konstruktiokielioppi, ohjaamaton oppiminen, osittain ohjattu oppiminen, todennäköisyysmallit, kielimallit, vektoriavaruusmallit, konekäännös, puheentunnistus

ISBN (painettu) 978-952-60-4882-6 **ISBN (pdf)** 978-952-60-4883-3

ISSN-L 1799-4934 **ISSN (painettu)** 1799-4934 **ISSN (pdf)** 1799-4942

Julkaisupaikka Espoo **Painopaikka** Helsinki **Vuosi** 2012

Sivumäärä 437 **urn** http://urn.fi/URN:ISBN:978-952-60-4883-3

Preface

This work has been carried out in the Adaptive Informatics Research Centre (AIRC) located at the Department of Information and Computer Science (Laboratory of Computer and Information Science prior to January 2008) of Aalto University (Helsinki University of Technology prior to January 2010). Apart from AIRC, the department, and the university, I have been supported by two graduate schools: the Graduate School of Language Technology in Finland (KIT) funded my research between November 2007 and July 2011 and the Helsinki Graduate School of Computer Science and Engineering (Hecse) provided a half year funding for finalizing the thesis. Further external funding was provided by the Academy of Finland via the projects "Learning efficient and cognitive representations for natural language utterances" and "Computational modelling of brain's language"; Nokia Corporation and Tekes via the project "Search for Personal Media Content"; and the European Community's Seventh Framework Programme via the Simple4All project and PASCAL2 Network of Excellence. I am also very grateful for the personal grants provided by Tekniikan edistämissäätiö (TES) and Kaupallisten ja teknillisten tieteiden tukisäätiö (KAUTE).

While these institutions have provided me the valuable opportunity for doing research, it is the people that have made my work truly successful and enjoyable. It has certainly been a privilege to work with so many great people.

First, I would like to thank my supervisor, Prof. Erkki Oja, for his efforts to make everything run smoothly and efficiently. As the head of AIRC, he created an excellent environment for doing research. I am also grateful for his advice, encouragement, and patience for the time it took me to write the thesis.

I would equally like to thank my two instructors, Doc. Mikko Kurimo and Dr. Krista Lagus. Having two instructors and two research groups around me has been a great advantage. Both Mikko and Krista arranged a significant part of the funding for my work. Krista's ideas have had a major influence on my research topics and my views on language and cognition. She has also taught me a lot on good research practices and how to present ideas elegantly and focus on the central points. Mikko's Speech Recognition research group and excellent connections to research groups at other universities have made many collaborations and resources available to me. He has always been available for my questions and given me a great amount of practical help and valuable advice.

Although not my official instructors, there are two more people who have provided me remarkable amount of advice and support: Dr. Mathias Creutz and Doc. Timo Honkela. Mathias has been a great mentor and company since the beginning of my research career, and he has been willing to answer my detailed questions regarding Morfessor also when working elsewhere. I am also grateful

for his invitation to work at Nokia Research Center in 2008. Finally, by the time I started working on the thesis overview, Mathias took a part-time job in the department and became in practice my third instructor. Timo, as the leader of the Computational Cognitive Systems research group, has arranged numerous events and discussions that have introduced me to many interesting people and ideas. I am grateful for his constant kindness and the opportunity to participate in the activities of the Cog group.

Next, I wish to thank all my other co-authors. A large part of the research in my thesis has been done in close collaboration with Oskar Kohonen, whose ideas and clear insights on various topics have been essential for this thesis. I am sincerely grateful to Prof. William Byrne, Prof. Riitta Salmelin, Dr. Adrià de Gispert, Dr. Teemu Hirsimäki, Dr. Annika Hultén, Dr. Minna Lehtonen, Dr. Vesa Siivola, Dr. Sebastian Spiegler, Dr. Abhishek Tripathi, Dr. Ville Turunen, Tiina Lindh-Knuutila, Mari-Sanna Paukkeri, Markus Sadeniemi, and Jaakko Väyrynen for their contributions on the publications of this thesis.

Prof. Richard Wicentowski and Doc. Krister Lindén have acted as the preliminary examiners for this thesis. I greatly appreciate their effort and feedback that has helped me to improve the thesis.

Apart from my supervisor, instructors, and the pre-examiners, I have received feedback on different versions and parts of the manuscript of this thesis from many people. Invaluable help was provided by Dr. Mathias Creutz, who was the first one to read many parts of the text. I am also grateful to Dr. Ritabrata Dutta, Oskar Kohonen, Tiina Lindh-Knuutila, Jaakko Väyrynen, and Teemu Ruokolainen for pointing out my mistakes and suggesting improvements.

Furthermore, I would like to thank Dr. Graeme W. Blackwood, Dr. Arto Klami, Mikaela Kumlander, Laura Leppänen, André Mansikkaniemi, Janne Pylkkönen, and Tommi Vatanen for their collaboration on related research articles; Dr. Yoan Miche for translating the abstract of the journal article published in TAL; and the secretaries and other support personnel of the department for their practical help and efforts. Finally, an essential part of this work would not have been possible without the enthusiastic researchers all over the world who have participated in the Morpho Challenge competitions.

These years at the university have also included other things than research. I have had a fun time having daily lunch and occasional dinners together; chatting in the offices, coffee rooms and corridors; playing floorball and badminton; skiing and skating in the winter, cruising to Tallinn in the spring, picking mushrooms in the fall. For this, I want to thank everyone whose company I have enjoyed, and in particular all the former and present members of the Cog and Speech groups. Special thanks to Jaakko, Marisa, Oskar, and Tiina—I consider you friends as much as colleagues.

Finally, I wish to thank my family, relatives, and friends outside the university. Especially I thank my parents Ritva and Veikko, who have supported me and my education in many ways. Most of all, I thank Anna, who has not only encouraged and supported my work, but also made sure it is not the most important thing in my life.

Espoo, November 5, 2012,

Sami Virpioja

Contents

Preface 7

Contents 9

List of publications 13

List of abbreviations 15

List of symbols and notations 17

1. Introduction 19
- 1.1 Machine learning in natural language processing . . . 22
- 1.2 Contributions of the thesis . . . 25
- 1.3 Summary of publications and author's contributions . . . 27
- 1.4 Structure of the thesis . . . 29

2. Machine learning essentials 31
- 2.1 Random variables and distributions . . . 32
- 2.2 Graphical models . . . 34
  - 2.2.1 Directed graphical models . . . 34
  - 2.2.2 Undirected graphical models . . . 35
- 2.3 Markov models and finite-state machines . . . 36
  - 2.3.1 Markov processes . . . 36
  - 2.3.2 Markov model . . . 37
  - 2.3.3 Hidden Markov models . . . 37
  - 2.3.4 Conditional Markov models . . . 38
  - 2.3.5 Finite-state machines . . . 39
- 2.4 Information theory . . . 40
  - 2.4.1 Quantities of information and uncertainty . . . 40
  - 2.4.2 Source coding theorem and Kraft's inequality . . . 41
  - 2.4.3 Noisy channel coding theorem . . . 42
  - 2.4.4 Divergence of two distributions . . . 43
  - 2.4.5 Algorithmic information theory . . . 44
- 2.5 Learning setups . . . 45
  - 2.5.1 Supervised learning . . . 45
  - 2.5.2 Unsupervised learning . . . 46
  - 2.5.3 Semi-supervised learning . . . 47
  - 2.5.4 Multi-view learning . . . 48
  - 2.5.5 Multi-task and transfer learning . . . 49

2.5.6 Reinforcement learning 49
2.6 Parametric machine learning 50
2.6.1 Cost function 50
2.6.2 Maximum-likelihood estimate 51
2.6.3 Overfitting and underfitting 51
2.6.4 Cross-validation 52
2.6.5 Regularization 52
2.6.6 Bayesian parameter estimation 52
2.6.7 Bayesian model selection 53
2.6.8 Minimum description length principle 54
2.6.9 Learning algorithms 58
2.7 Non-parametric and semi-parametric models 59
2.7.1 Memory-based learning 59
2.7.2 Kernel methods 60
2.7.3 Mixture models 60
2.7.4 Non-parametric Bayesian methods 61
2.8 Common unsupervised learning methods 62
2.8.1 Matrix decompositions 62
2.8.2 Principal component analysis 63
2.8.3 Canonical correlation analysis 63
2.8.4 Hierarchical and K-means clustering 65
2.8.5 Expectation-maximization algorithm 66

3. Linguistic data and theories **67**
3.1 Linguistic units and their relations 68
3.1.1 Distributions of the units 69
3.1.2 Meaning and form 70
3.1.3 Syntagmatic and paradigmatic relations 71
3.1.4 Word forms and lexemes 72
3.1.5 Part-of-speech categories 73
3.1.6 Constituency 74
3.1.7 Morphology 74
3.1.8 Syntax 78
3.1.9 Semantics and pragmatics 80
3.2 On theories of grammar 83
3.2.1 Views on the scope of grammar 84
3.2.2 Poverty of the stimulus and universal grammar 85
3.2.3 Models of morphology 86
3.2.4 Phrase structure grammar 88
3.2.5 Dependency grammars 90
3.2.6 Context-sensitive grammars 91
3.2.7 Construction grammars 91

4. Statistical language modeling **93**
4.1 Evaluation methods and applications 95
4.2 N-gram models 95
4.2.1 Smoothing 96
4.2.2 Back-off and interpolation 98
4.2.3 Kneser-Ney smoothing 99
4.2.4 Variable length n-grams 101
4.2.5 Cluster n-grams 104

4.2.6 Back-off graph and skipping . . . 105
4.2.7 Factored models . . . 106
4.3 Beyond n-gram models . . . 106
4.3.1 Grammar-based language models . . . 106
4.3.2 Maximum-entropy language models . . . 107
4.3.3 Continuous-space language models . . . 109
4.3.4 Models for domain adaptation . . . 112
4.4 Kneser-Ney smoothing and pruned models . . . 114
4.4.1 Kneser-Ney distributions for varigram models . . . 115
4.4.2 Pruning and growing algorithms . . . 116
4.4.3 Experiments . . . 118
4.4.4 Discussion . . . 120
4.5 Clustering of n-gram histories . . . 120
4.5.1 Context cluster model . . . 121
4.5.2 Clustering algorithm . . . 122
4.5.3 Model prior . . . 123
4.5.4 Experiments . . . 124
4.5.5 Discussion . . . 126

5. Representation learning **129**
5.1 Vector space models . . . 130
5.1.1 Weighting . . . 131
5.1.2 Length normalization . . . 132
5.1.3 Dimensionality reduction . . . 132
5.2 Probabilistic topic models . . . 134
5.3 Evaluation of representation learning . . . 136
5.3.1 Application evaluations . . . 136
5.3.2 Direct evaluations . . . 138
5.4 Vector space model evaluation using CCA . . . 138
5.4.1 Mathematical foundation . . . 139
5.4.2 Evaluation setup . . . 140
5.4.3 Validation experiments . . . 142
5.4.4 Discussion . . . 144

6. Selecting lexical units **145**
6.1 Problem definition . . . 145
6.1.1 Qualitative criteria for unit selection . . . 147
6.1.2 Evaluating unit selection . . . 148
6.2 Comparison of linguistic units . . . 149
6.2.1 Letters and syllables . . . 149
6.2.2 Words . . . 150
6.2.3 Lexemes and stems . . . 151
6.2.4 Morphemes and morphs . . . 152
6.2.5 Phrases and phrasal constructions . . . 154
6.3 Evaluations for learning morphology . . . 156
6.3.1 Indirect evaluations . . . 157
6.3.2 Automatic linguistic evaluation . . . 164
6.3.3 Psycholinguistic evaluation . . . 178
6.4 MDL-inspired models for learning constructions . . . 184
6.4.1 Morfessor . . . 184
6.4.2 Learning of allomorphy . . . 191

6.4.3 The effect of corpus size and word frequencies 200
6.4.4 Semi-supervised learning of morphology 204
6.4.5 Learning of phrasal constructions 208

7. Conclusions and future directions **215**
7.1 Models for learning constructions 215
7.2 Direct evaluations 217
7.3 Applications 218

A. Appendices **221**
A.1 Proof for optimal feature generators in linear bilingual document model 221
A.2 Morpho Challenge evaluation results 222
A.3 Examples of transformations extracted by Allomorfessor 232

Bibliography **235**

Publications **267**

List of publications

This thesis consists of an overview and of the following publications which are referred to in the text by their Roman numerals. The numbering of the publications follows the order in which they are discussed in the overview.

I Vesa Siivola, Teemu Hirsimäki, and Sami Virpioja. On growing and pruning Kneser-Ney smoothed n-gram models. *IEEE Transactions on Audio, Speech and Language Processing*, 15(5):1617–1624, July 2007.

II Sami Virpioja and Mikko Kurimo. Compact n-gram models by incremental growing and clustering of histories. In *Proceedings of 9th International Conference on Spoken Language Processing (Interspeech 2006 — ICSLP)*, Pittsburgh, Pennsylvania, USA, pages 1037–1040, September 2006.

III Sami Virpioja, Mari-Sanna Paukkeri, Abhishek Tripathi, Tiina Lindh-Knuutila, Krista Lagus. Evaluating vector space models with canonical correlation analysis. *Natural Language Engineering*, 18(03):399–436, July 2012.

IV Sami Virpioja, Jaakko J. Väyrynen, Mathias Creutz, and Markus Sadeniemi. Morphology-aware statistical machine translation based on morphs induced in an unsupervised manner. *Proceedings of the Machine Translation Summit XI*, Copenhagen, Denmark, pages 491-498, September 2007.

V Adrià de Gispert, Sami Virpioja, Mikko Kurimo, and William Byrne. Minimum Bayes risk combination of translation hypotheses from alternative morphological decompositions. *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, Boulder, Colorado, USA, pages 73-76, June 2009.

VI Sami Virpioja, Ville T. Turunen, Sebastian Spiegler, Oskar Kohonen, and Mikko Kurimo. Empirical comparison of evaluation methods for unsupervised learning of morphology. *Traitement Automatique des Langues*, 52(2):45–90, 2011.

VII Sami Virpioja, Minna Lehtonen, Annika Hultén, Riitta Salmelin, and Krista Lagus. Predicting reaction times in word recognition by unsupervised learning of morphology. In *Artificial Neural Networks and Machine Learning — ICANN 2011, Espoo, Finland, June 14–17, 2011, Proceedings, Part I*, volume 6791 of Lecture Notes in Computer Science, pages 275–282, June 2011.

VIII Sami Virpioja, Oskar Kohonen, and Krista Lagus. Unsupervised morpheme analysis with Allomorfessor. In *Multilingual Information Access Evaluation I. Text Retrieval Experiments: 10th Workshop of the Cross-Language Evaluation Forum, CLEF 2009, Corfu, Greece, September 30 – October 2, 2009, Revised Selected Papers*, volume 6241 of Lecture Notes in Computer Science, pages 609–616, September 2010.

IX Sami Virpioja, Oskar Kohonen, and Krista Lagus. Evaluating the effect of word frequencies in a probabilistic generative model of morphology. In *Proceedings of the 18th Nordic Conference of Computational Linguistics (NODALIDA 2011)*, Riga, Latvia, pages 230–237, May 2011.

X Oskar Kohonen, Sami Virpioja, and Krista Lagus. Semi-supervised learning of concatenative morphology. In *Proceedings of the 11th Meeting of the ACL Special Interest Group on Computational Morphology and Phonology*, Uppsala, Sweden, pages 78–86, July 2010.

XI Krista Lagus, Oskar Kohonen, and Sami Virpioja. Towards unsupervised learning of constructions from text. In *Proceedings of the Workshop on Extracting and Using Constructions in NLP of the 17th Nordic Conference on Computational Linguistics (NODALIDA)*, Odense, Denmark, SICS Technical Report T2009:10, pages 16–21, May 2009.

List of abbreviations

ACC	Accumulative context clustering
AI	Artificial intelligence
AMM	Aggregate Markov model
ASR	Automatic speech recognition
BPR	Boundary precision and recall (evaluation)
BSS	Blind source separation
CCA	Canonical correlation analysis
CFG	Context-free grammar
CoMMA	Co-occurrence-based metric for morphological analysis
CRF	Conditional random field
DMM	Distributed Markov model
EM	Expectation-maximization (algorithm)
EMMA	Evaluation metric for morphological analysis
EP	Entropy-based pruning
FSA	Finite-state acceptor
FST	Finite-state transducer
GT	Good-Turing (smoothing)
IA	Item-and-arrangement (morphology)
IP	Item-and-process (morphology)
IC	Immediate-constituent
ICA	Independent component analysis
IPA	International phonetic alphabet
IR	Information retrieval
HMM	Hidden Markov model
HPY	Hierarchical Pitman-Yor (model)
KN	Kneser-Ney (smoothing)
KNG	Kneser-Ney growing
KP	Kneser pruning
LDA	Latent Dirichlet allocation
LER	Letter error rate
LSA	Latent semantic analysis
LSV	Letter successor variety
MAP	Maximum a posteriori; mean average precision
MBR	Minimum Bayes risk
MDL	Minimum description length
ME	Maximum entropy
MEMM	Maximum-entropy Markov model
ML(E)	Maximum likelihood (estimate)

MLP	Multi-layer perceptron
MT	Machine translation
NLP	Natural language processing
OOV	Out-of-vocabulary
PCA	Principal component analysis
PCFG	Probabilistic context-free grammar
PLSA	Probabilistic latent semantic analysis
POS	Part-of-speech
RBM	Restricted Boltzmann machine
RKP	Revised Kneser pruning
RNN	Recurrent neural network
SMT	Statistical machine translation
SOM	Self-organizing map
SVD	Singular value decomposition
TTS	Text-to-speech
TWOL	Two-level morphology
VSM	Vector space model
WER	Word error rate
WKP	Weighted Kneser pruning
WP	Word-and-paradigm (morphology)

List of symbols and notations

$A, B, \ldots$	Random variables or events; integers
$\mathcal{A}, \mathcal{B}, \ldots$	Domains of random variables
$a, b, \ldots$	Scalars
$\alpha, \beta, \ldots$	Scalars
$\mathbf{a}, \mathbf{b}, \ldots$	Vectors
$\boldsymbol{\alpha}, \boldsymbol{\beta}, \ldots$	Vectors
$\mathbf{A}, \mathbf{B}, \ldots$	Matrices
$\boldsymbol{\Lambda}, \boldsymbol{\Sigma}, \ldots$	Matrices
$\boldsymbol{a}, \boldsymbol{b}, \ldots$	Strings or other sequences
A, B, ...	Sets
$G, H, \ldots$	Random distributions
$\mathcal{N}$	Gaussian distribution
G, P, ...	Random processes
DP	Dirichlet process
PYP	Pitman-Yor process
$\lvert x \rvert$	Absolute value of x
$\lvert \mathbf{x} \rvert$	ℓ_1-norm of $\mathbf{x}$
$\lVert \mathbf{x} \rVert$	ℓ_2-norm of $\mathbf{x}$
$\lvert \boldsymbol{x} \rvert$	Length of $\boldsymbol{x}$
$\lvert \mathrm{X} \rvert$	Size of X
$d(x, y)$	Distance between x and y
$\mathrm{sim}(x, y)$	Similarity of x and y
$\mathrm{I}(A)$	Indicator function for A (one if A is true, zero otherwise)
$p(A)$	Probability of A
$p(x), q(x)$	Probability distributions for X
$E_X[f(x)]$	Expected value of $f(x)$ over $\mathcal{X}$
$H(X)$	Entropy of X
$I(X; Y)$	Mutual information between X and Y
$H_q(X)$	Cross-entropy of X with distribution q
$D(p \parallel q)$	Relative entropy between p and q
$\mathrm{IRad}(p \parallel q)$	Information radius between p and q
$l_C(x)$	Code/description length of x with coding C

$\boldsymbol{D}$	Data set
N	Number of samples in a data set
$\mathcal{M}$	Model
$\boldsymbol{\theta}$	Model parameters
$L(\boldsymbol{\theta}, \boldsymbol{D}, \ldots)$	Cost function
Σ	Alphabet; vocabulary
V	Alphabet or vocabulary size
w, v	Units from vocabulary (e.g. words)
$\boldsymbol{w}_i^j$	Sequence of units $(w_i, w_{i+1}, \ldots, w_j)$
$c(\boldsymbol{w})$	Token count for $\boldsymbol{w}$
$c^*(\boldsymbol{w})$	Adjusted token count for $\boldsymbol{w}$
$t(\bullet\boldsymbol{w})$	Type count for left contexts of $\boldsymbol{w}$
$t(\boldsymbol{w}\bullet)$	Type count for right contexts of $\boldsymbol{w}$
$t(\bullet\boldsymbol{w}\bullet)$	Type count for surrounding contexts of $\boldsymbol{w}$
$\bar{\boldsymbol{h}}$	Back-off context for $\boldsymbol{h}$ ($\boldsymbol{h} = v\bar{\boldsymbol{h}}$ for some v)
$\gamma(\boldsymbol{h})$	Back-off weight for $\boldsymbol{h}$
$\mathcal{L}$	Model lexicon
$\mathcal{G}$	Model grammar
$\phi(\boldsymbol{t})$	Tokenization for string $\boldsymbol{t}$
$\Phi(\boldsymbol{t})$	Set of tokenization parses for string $\boldsymbol{t}$
$\phi^{-1}(\boldsymbol{s})$	Detokenization for parse $\boldsymbol{s}$

1. Introduction

Language is a system of complex communication, predominant for the human species. The spoken, written, and sign languages, evolved for the need of communication between humans, are referred to as *natural languages* to separate them from artificially constructed languages (such as Lojban or Klingon) and formal languages (such as programming languages).[1]

Language is often seen as a central reflection or even component of human intelligence, separating us from other animals. Thus it is not a surprise that one of the goals in artificial intelligence (AI), a subfield of computer science that studies creating intelligent machines, is to get machines to understand and generate natural languages. One demonstration of this goal is the famous Turing test (Turing, 1950), which considers a machine to be intelligent if a human judge cannot distinguish it from a human during a real-time conversation. The annual Loebner prize competition awards the programs that are judged to be most human-like in the test.

While the Turing test is considered as a test for AI, the "chatterbots" that participate in the Loebner prize competition actually use a large set of predefined rules cleverly designed by their programmers. For example, the winner of the 2010 competition uses sophisticated pattern matching for the input sentences to find an appropriate answer from a list of predefined cases (Wilcox, 2011). For example, the following rule defines one answer to the input "What is your earliest childhood memory?" as well as "Can you tell me some early memory of yours?":[2]

```
u: ( « you [early childhood] memory » ) I remember playing with an HO
railroad set.
```

To help with matching and answering, also a large number of concepts and pieces of world knowledge are defined. For example, the following definition tells that *crew*, *personnel*, *staff*, and *team* are terms for the concept `staff_groups`:

```
concept: ~staff_groups ( crew personnel staff team )
```

Similarly, the following definition gives seven examples for things that can have orange color:

```
orange [orange marigold goldfish pumpkin cat carrot tangerine]
```

[1] Some constructed international auxiliary languages based on natural languages, such as Esperanto, can be considered almost natural especially if learned by a child as the first language (Karlsson, 2008).

[2] The examples are from the ChatScript software by Bruce Wilcox, available from `http://sourceforge.net/projects/chatscript/`.

While this kind of approach is useful, for example, in computer games and restricted natural language interfaces, where it is enough to communicate in a way that *looks* intelligent, it is far from the long term goal of AI, that is, creating machines with human-like general intelligence ("strong AI"). A central part in general intelligence, obviously missing from the chatterbot example, is *learning*. All syntactic and semantic knowledge of the language have to be explicitly written out for the program. In contrast, humans learn language based on the stimulus that they get from other humans and the environment.

The branch of AI that studies algorithms that allow computers to learn from empirical data is called *machine learning*. Using machine learning on natural language data is part of *computational linguistics*, an interdisciplinary subfield of linguistics dealing with statistical and rule-based modeling of natural languages. Computational natural language learning is an interesting research topic from several viewpoints: (1) as a way to improve the computer applications that deal with natural languages, (2) as a way to collect empirical evidence for theoretical models in linguistics and cognitive science, and (3) as an ultimate challenge for the machine learning research.

The main motivation for the work in this thesis is that machine learning can provide improvements to many important applications in human-computer and human-human interaction, such as automatic speech recognition, information retrieval, and machine translation. The subfield in the intersection of AI and computational linguistics that studies the computational aspects of these applications is called *natural language processing* (NLP). Developing hand-crafted tools for the need of NLP applications would not be a major problem with a single language, but the number of languages and dialects in the world is large: the estimates for the number of languages in the world usually vary from 4000 to 7000, and separating different dialects increases the number to above 20000 (Karlsson, 2008). In contrast to writing out rules and other pieces of information, learning from data requires little human work, once the learning algorithms and data sets are available. Unsupervised learning techniques, which try to find structure from unannotated data, also obviate the need of manual annotation of data. In addition to building applications for large, resource-rich languages, NLP also provides tools for language documentation of small languages facing extinction (Bird, 2009). Both supervised (cf. Palmer et al., 2009) and unsupervised (cf. Hammarström and Borin, 2011) methods may alleviate this vast and urgent task.

Machine learning can also contribute to the field of theoretical linguistics and cognitive science, especially with respect to the questions on language acquisition. The debate on whether the language ability of humans is enabled by an innate language-specific device (nativism) or just general-purpose learning mechanisms (empiricism), grounded already in the 17th-century philosophy, has been going on since Noam Chomsky's famous review article concerning the behaviorist psychology's account on language (Chomsky, 1959). The strongest division goes between the proponents of Chomskyan generative grammar, who believe in the innateness of language ability, and connectionists, who argue that language (among other mental and behavioral phenomena) can be modeled with emergent processes of interconnected networks of simple units. While the kind of neural networks proposed by connectionists are only one possible approach, strong results for language learning with any kind of machine learning method has implications on this debate (cf., e.g., Clark, 2001; Lappin and Shieber, 2007; Hsu et al., 2011). If even machines can learn the grammar of a language from raw data with little prior information, there is no reason why human brain would

Table 1.1. Examples of English constructions of varying size and complexity. Adapted from Goldberg (2003).

Construction	Form / Example	Function
Morpheme	**anti-**, **pre-**, **-ing**	
Word	**avocado**, **anaconda**, **and**	
Complex word	**daredevil**, **shoo-in**	
Filled idiom	**going great guns**	
Partially filled idiom	**jog X's memory**	
Covariational-conditional construction	Form: **the Xer the Yer** **"the more you think about it, the less you understand"**	Meaning: linked independent and dependent variables
Ditransitive construction	Form: **Subj V Obj1 Obj2** **"He baked her a muffin"**	Meaning: transfer (intended or actual)
Passive	Form: **Subj aux VPpp (PP_{by})** **"The armadillo was hit by a car"**	Discourse function: make undergoer topical and/or actor non-topical

need a separate "language acquisition device".

What is it exactly that should be learned by a machine to understand and produce human languages? A common answer is the linguistic knowledge on different categories such as phonology, morphology, syntax, semantics, pragmatics, and discourse (Jurafsky and Martin, 2008). However, a strict division into these categories may not be necessary or even useful: a growing number of linguistic theories (e.g., Lakoff, 1987; Langacker, 1987, 1991; Fillmore et al., 1988; Goldberg, 1995, 2006; Croft, 2001; Feldman, 2006) propose that linguistic knowledge is based on form-meaning pairs, called *constructions*, that may include all of them. Some examples of different constructions are shown in Table 1.1. From the perspective of machine learning, these theories provide both a single term for what should be learned—constructions—and evidence of what kind of data and type of learning are needed to learn them.

Even if the empiricist view on language acquisition is correct, natural language data poses major challenges to the machine learning techniques. First, the amount of available data varies a lot. For world languages and especially those functioning as *lingua franca*, such as English, the amount of data in electronic form is huge, largely owing to the growth in communication via Internet. For the most under-resourced languages, the only data sets might be those manually collected by linguists. Second, the data is always sparse. The number of ways to form meaningful utterances is so enormous, that no matter how large a text corpus is collected, it covers only a small fraction of the language. For morphologically rich languages, such as Finnish, even collecting all the possible word forms is infeasible. Third, the data is not stationary but languages change, often by adopting words and conventions from other languages. In addition to the diachronic variation, there is variation between the speakers of the same language (due to, e.g., dialects, social or socioeconomic classes, and age) and variations reflecting the situation and the medium of communication (e.g., colloquial language, formal speech, academic writing). Ultimately, the complexity of the language data is superior to most other types of data: a model that can encompass *all* aspects of human language also has to attempt to be a model for human minds.

1.1 Machine learning in natural language processing

The ultimate goal of NLP is to get machines to understand and generate natural languages. However, this is often considered an "AI-complete" problem, that is, something that requires all the other aspects of artificial intelligence to be solved. In practice, most of the NLP research concentrates on improving specific applications that assist interaction either between humans and computers or between humans that do not have a common language. The main applications include automatic speech recognition (ASR), speech synthesis (text-to-speech, TTS), information retrieval (IR), machine translation (MT), text summarization, and question answering. These are also components that can be combined to produce more complex applications such as spoken document retrieval (ASR+IR) or speech-to-speech translation (ASR+MT+TTS).

The modern, statistical approaches to the applications above rely heavily on machine learning techniques. Problems specific to individual applications include, for example, constructing acoustic models in ASR, categorizing a set of documents in IR, and finding the most probable translation candidate in statistical machine translation (SMT). However, there are also a few fundamental machine learning problems that are more independent of the application. Two easily identifiable problems are the following:

- *Statistical language modeling:* Given a certain fragment of written language, determine how probable it is in proportion to all other fragments of the same type. A typical example is the estimation of the probability mass function $p(S)$ for sentences S.
- *Representation learning:* Given a large collection of documents, encode their semantic content so that the documents most similar to a given document can be found quickly and accurately.

The former is needed, for example, in ASR or SMT for generating accurate and fluent text, while the latter is needed for all kinds of retrieval and categorization tasks. Both are inherently unsupervised learning problems: for efficient solutions, the learner has to find the hidden structure in the data.

Regardless of which of the two is required by the application, there is one more problem that precedes them:

- *Selecting lexical units:* Select the basic units of language that are used for further processing. In statistical language modeling, the lexical units define the set of possible values for the variables. For example, unigram models define a categorical distribution $p(W)$ over the units W, usually words. In representation learning, the lexical units define the set of variables that are modeled. For example, a bag-of-words model examines the distributions of different words over a set of documents.

The problem of unit selection has gained only limited attention. The main reason is that for English, which is the dominant language in methodological and empirical studies in the field, very simple heuristics work well enough. If the problem is density estimation, the most common words are selected as the units. If the problem is representation learning, inflected word forms are reduced to their stems or base forms and hand-collected stop word lists are applied to decrease the number of the units. In consequence, the unit selection is often treated as a trivial preprocessing task that has to be done before going into the actual

problems.

However, the heuristic approach for unit selection is not as simple for some other languages. In Chinese, words are not separated from each other by white space, so selecting words as the lexical units requires non-trivial preprocessing. In Finnish, the rich morphology results in a huge number of different word forms, some of them too rare to use as units in statistical models. While both problems can be solved by manual and rule-based approaches, they are more costly for less resourced languages.

Recalling the notion of construction from above, the unit selection problem can as well be called the problem of *learning constructions*. As the constructions of a chunk of text encode all of its semantics, they are the best possible starting point for learning any representations that try to encode semantic similarities. Or as Goldberg (2006, p. 228) argues, *"constructions are highly valuable both in predicting meaning, given the form, and in predicting form, given the message to be conveyed"*. Moreover, because most of the complex constructions are unpredictable (either in form or meaning) from their component parts or other existing constructions, they are relevant for estimating the probability of a text chunk. For example, while the words **"lion"** or **"sleeve"** are unlikely to occur alone in political speech, the idioms **"lion's share"** and **"roll up [one's] sleeves"** are more likely due to their metaphorical meaning.

Designing a method for one of the learning problems is only the first part of the work. In one way or another, the method also needs to be evaluated. Evaluations can be coarsely divided into two types (cf. Jurafsky and Martin, 2008, p. 129). Evaluations, in which the performance is measured in the end-user NLP applications, are called indirect, extrinsic, or *in vivo* evaluations. Evaluations that try to measure the quality of a certain model or result independent of any application are called direct, intrinsic, or *in vitro* evaluations.

The overview of the three machine learning problems defined above and their evaluations are shown in Figure 1.1. The box in the middle of the diagram marks the first problem, selecting the lexical units. It includes several subproblems, such as morphological analysis or word segmentation. After that, it is possible to go either to statistical language modeling (up) or learning new representations (down). A few common end-user applications are shown on the right-hand side of learning problems. Some applications might use either statistical language modeling, representation learning, or both. For example, while information retrieval is often done with the latter (vector space models), also the former (statistical language models) can be applied (Ponte and Croft, 1998). Moreover, some machine learning methods solve several problems together. For example, probabilistic models estimate the distribution of the data regardless of the main task being representation learning (e.g., Hofmann, 1999a; Blei et al., 2003) or unit selection (e.g., Brent, 1999; Goldwater et al., 2006; Creutz and Lagus, 2007).

Indirect evaluations are shown to the right of the applications in Figure 1.1. While it can be argued that only the evaluation in an application indicates the real usefulness of a method, it is often complicated and expensive in terms of time and manual work required. In fact, evaluating a system for a certain application is often a problem of its own. For example, how to measure the usefulness of an SMT-based translation service such as Google Translate?[3] While translations of single sentences are often far from perfect (at the present time), it certainly helps browsing Web pages in languages unknown to the user.

[3] Available from `http://translate.google.com`.

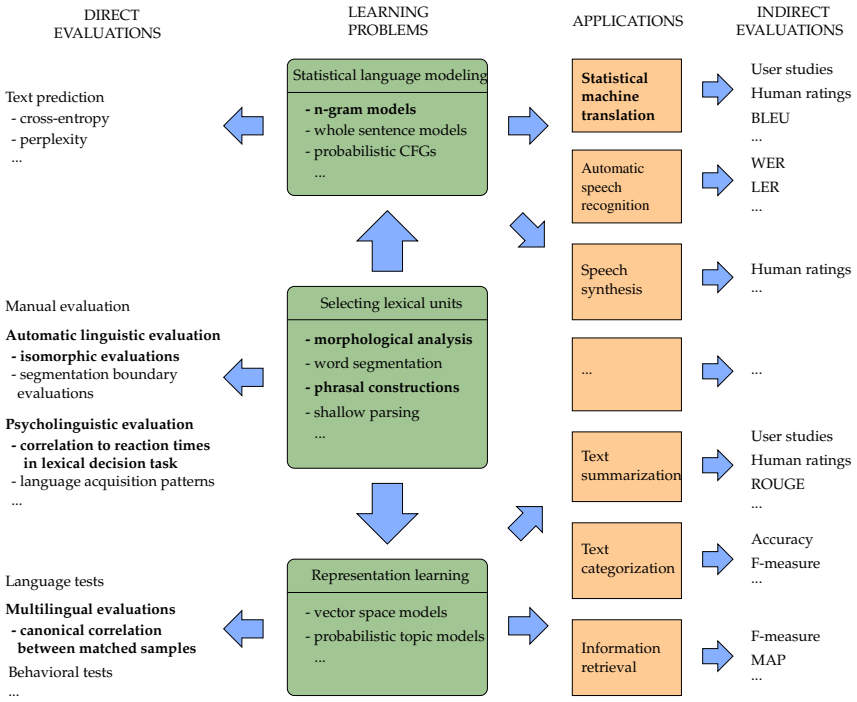


Figure 1.1. The main learning problems in statistical NLP and examples of the models, subtasks, and evaluations. The contributions of this thesis concern the models, tasks, and evaluations that are emphasized.

The most expensive and thorough way of indirect evaluation are user studies, rarely used in the field except for commercial products. A simpler way is to have people rate the performance of the application. However, extensive use of human ratings is expensive, too, and thus mostly limited to yearly evaluation campaigns. Quick and cheap evaluation is possible only by using *automatic metrics* such as word error rate (WER) for ASR or mean average precision (MAP) for IR. However, automatic metrics require reference results, which often require manual work. The cost of the work varies depending on the time and the skills required: while the transcriptions needed for ASR can be checked by almost any literate person, accurate reference translations needed for SMT require professional translators. In many cases, the indirect evaluations are restricted to a limited number of published data sets.

Another problem in indirect evaluations is that as the whole systems often require many components and steps, they may level out the differences or introduce biases that give disadvantage to non-standard approaches. For example, the full system may have to be redesigned to account for whether words or letters are selected as the lexical units.

Examples of direct evaluations considered in this thesis are given in the left side of Figure 1.1. As they are intended to be quicker and simpler than indirect evaluations, the direct evaluations are important for advancing the research and drawing contributions outside the NLP community. For example, researchers from machine learning groups would have to use a considerable amount of time to build an ASR system for application evaluation. While there is no guarantee that the direct evaluation results related to intermediate goals will correlate with the results of a specific application evaluation, using several well-designed di-

rect evaluations is likely to give the correct idea on the quality of the evaluated algorithm and give insight to its problems.

In some cases, direct evaluations are easy. A common example is calculating cross-entropy for a new text with an n-gram model. In other cases, an intermediate goal, such as getting a morphological analysis close to a linguistic reference analysis ("gold standard"), has to to be set. Moreover, if the learning method is unsupervised, the comparison of the results and the reference analysis may be non-trivial because of arbitrary labels. In yet other cases, there may not be any reference data. For example, if the task is to learn document representations, it is unclear how such representations could be evaluated in a direct manner.

1.2 Contributions of the thesis

The main hypothesis of this work is that instead of relying on word-based models and representations, there are essential benefits to using machine learning to select the lexical units for natural language processing tasks. Moreover, instead of learning how to predict classes or structures from data annotated by linguistic experts (supervised learning), it is often more useful to try to find the statistical regularities from unannotated data (unsupervised learning). With unsupervised machine learning approaches, the process of unit selection can be independent of the language as well as of the style and type of the text. In addition, the same unsupervised algorithm may provide units for multiple purposes, in contrast to a rule-based tool optimized for a specific application. Of course, this does not mean that linguistic expertise would be useless. Instead, it should be applied in a manner that allows effortless adaptation to different languages and applications. For example, language-universal knowledge should be used when designing the structure and priors of the models, and language-specific knowledge should be used to train the models in a semi-supervised manner.

It should be clear from the discussion of the previous section that testing the above hypotheses requires both unsupervised methods for selecting lexical units and methods for evaluating them. The contributions of this thesis include both.

Development of Morfessor. The family of methods for learning constructions that is developed and tested in this thesis is called Morfessor. Morfessor was originally developed by Creutz and Lagus (2002, 2004, 2005b,a, 2007) for unsupervised learning of morphological segmentation. The most important versions of Morfessor are Morfessor Baseline (Creutz and Lagus, 2002, 2005b) and Morfessor Categories-MAP (Creutz and Lagus, 2005a, 2007). Both are based on probabilistic generative models that use sparse priors inspired by the minimum description length (MDL) principle by Rissanen (1978).

This thesis presents three new extensions to Morfessor. The first extension considers the phenomenon of allomorphy (Publication VIII). While the previous versions of Morfessor only segment word forms, Allomorfessor captures orthographic variation in the surface forms of stem morphemes and thus reduces the size of the lexicon.

One known problem in Morfessor is that the segmentation results are greatly influenced by the size and type of the training data. A study of this effect is included in this thesis (Publication IX). If a single parameter that affects the weight of the data likelihood in the target function can be optimized for the evaluation score, the results of the algorithm can be drastically improved. Moreover, Mor-

fessor Baseline is extended to a complete semi-supervised setting, where a small set of annotated data is available for training (Publication X). As only a few hundred annotated words are enough to get a remarkable increase in the segmentation accuracy, the method increases the applicability of Morfessor also for poorly resourced languages.

Finally, Morfessor is extended outside the scope of morphological analysis. It is shown that the same type of model can learn meaningful phrasal constructions from sentence data (Publication XI).

Applications and evaluations for Morfessor. The main application for Morfessor has been automatic speech recognition, for which it has improved the results for several languages (Hirsimäki et al., 2006; Creutz et al., 2007; Hirsimäki et al., 2009). As most language models for ASR have been developed for words, it is important to find new solutions that may be better for other types of units. This thesis introduces new language modeling techniques that are useful for building compact models especially for sub-word units such as the morphs found by Morfessor (Publications I and II).

Another contribution is testing Morfessor in a machine translation application. Direct application of the morphs found by Morfessor Categories-MAP to a phrase-based statistical machine translation system instead of words is shown to produce relevant phrase pairs and reduce out-of-vocabulary words in translation (Publication IV). Moreover, if morph-based and word-based models are combined, significant gains for the evaluation scores are obtained (Publication V).

Finally, Morfessor has been evaluated on psycholinguistic data that has reaction times of humans in a lexical decision task (Publication VII). The probability estimates of Morfessor models, especially the Categories-MAP model, are found to predict the reaction times with a high accuracy.

Evaluation methods for unsupervised natural language learning. While the above contributions concerned Morfessor, the same evaluations can also be applied in other work on the same task. In fact, some of the work in this thesis is closely intertwined with the *Morpho Challenges*, annual evaluations for unsupervised morpheme analysis (Kurimo et al., 2010a). Morpho Challenges have been organized in 2005, 2007, 2008, 2009 and 2010. The objective of the participants is to "design a statistical machine learning algorithm that discovers which morphemes (smallest individually meaningful units of language) words consist of" (Kurimo et al., 2008). All Morpho Challenges have included direct evaluations that use linguistic reference analyses for several languages. In addition, the first Morpho Challenge (Kurimo et al., 2006a) included speech recognition tasks and the two next Challenges (Kurimo et al., 2008, 2009) included information retrieval tasks. The last two challenges (Kurimo et al., 2010c,b) have included both information retrieval and machine translation tasks, and the machine translation task was designed along the lines of Publications IV and V.

As mentioned in the previous section, comparing the results of an unsupervised method with a linguistic reference analysis is not trivial, because the exact correspondences of the predicted features and reference features is not known. For example, in the case of unsupervised part-of-speech tagging, there is no direct way to tell whether a certain predicted tag ("tag number 23") matches the reference tag ("adjective"). For multidimensional features, this problem is substantially more complex.

This thesis includes the first major empirical comparison of direct evaluation

methods for the task of unsupervised learning of morphology (Publication VI). The comparison is based on the database of results collected in Morpho Challenges. In addition to reviewing and studying the existing evaluation methods, a few new methods are introduced. The best of the new direct evaluation methods are quick, robust, and show reasonable correlations with the indirect evaluations in IR and SMT.

In addition to the various evaluation methods for morphology induction, a novel direct evaluation method for the learning of vector representations for documents is presented (Publication III). Direct evaluation of such representations is seemingly impossible, as there are no "reference vectors" to which they could be compared. This problem is circumvented by using a multilingual parallel corpus: a document and its translation can be seen as two views for the same underlying semantics, so high dependence between the features of the document pairs indicates meaningful representations. The assignment of the two sets of features is solved by an unsupervised learning technique called canonical correlation analysis. The evaluation method provides a simple way to evaluate both representation learning and the lexical units selected prior to it.

1.3 Summary of publications and author's contributions

Publications I and II study statistical language modeling in large vocabulary speech recognition tasks.

Publication I considers growing and pruning of n-gram models that use the current state-of-the-art smoothing method, modified Kneser-Ney smoothing. Improved algorithms for the growing and pruning of n-gram models based on the modified Kneser-Ney smoothing are introduced. The algorithms allow building very accurate models using less computational resources. The present author participated in developing the new algorithms and designing the experiments, and had minor contributions to the writing of the article.

Publication II introduces a cluster-based n-gram model that combines the predictive probability distributions of observed histories if the distributions are similar enough, thus decreasing the number of parameters in the model. The present author developed and implemented the proposed model and its training algorithm, designed and ran the experiments, and wrote the article.

Publication III proposes a new method for direct evaluation of vector space models of documents. Using canonical correlation analysis (CCA) on the features extracted for a set of documents and their translations to another language finds linear transformations for the two sets of features so that the transformed features are optimal with respect to correlations. In the simplest case, a correlated feature may be a certain word in the first language and its translation in the second language. It is argued that high correlations indicate that the representations encode information regarding the meaning of the documents, not just arbitrary features of the texts. The method is validated by a set of experiments on sentence representations. The present author invented the idea of using CCA as an evaluation method, and the details of the evaluation measure were developed by the author and Abhishek Tripathi. The experiments were designed by all the authors. The present author implemented the method, ran all the experiments except for the manual validation, and was the main writer of the article.

Publications IV and V provide a setup for applying the unsupervised models of morphology directly to existing statistical machine translation systems.

Publication IV proposes using Morfessor as a language-independent preprocessing tool in a phrase-based statistical machine translation system. Using the statistically extracted morphs as lexical units reduces the problem of out-of-vocabulary units for morphologically complex languages. The experiments were designed by all the authors. The present author performed most of the experiments and wrote a major part of the article.

Publication V describes a strategy for combining translation hypotheses from statistical machine translation systems that use different morphological decompositions. The idea is to first train several independent translation systems that use different lexical units—such as words, morphological tags, or morphs found in a unsupervised manner—and then improve the performance by selecting the final translation using minimum Bayes risk (MBR) combination of the best translation hypotheses from all the systems. The present author was responsible for designing, running, and reporting the Finnish–English translation experiments.

Publication VI considers the evaluation of unsupervised learning of morphology. The article reviews the previous work on evaluating methods and introduces new methods for linguistic evaluation based on co-occurrence analysis of the words and morphemes. The new methods were developed by the present author. The current author also performed all the experiments except for the information retrieval experiments and the study of robustness with shared morpheme padding, and was the main writer of the article.

Publication VII studies Morfessor as a model of morphological processing in humans. The probabilities given by the model are compared with human reaction times in a lexical decision task. The correlations are shown to be higher than those for simple word statistics previously identified as important factors affecting the recognition times. Moreover, it was found that both the type and the size of the training data have considerable effect on the results. Designing the experiments and interpreting the results was done jointly by all the authors. The present author and Minna Lehtonen carried out the experiments. The present author was also the main writer of the article.

Publications VIII, IX, X, and XI are continuations for the development of Morfessor, a statistical method for unsupervised learning of morphology.

Publication VIII extends Morfessor to the case that the morphemes of the analyzed language have orthographic variants, allomorphs. In Morfessor, two allomorphs, such as **"pretty"** and **"pretti"** (in **"prettier"**), are either stored as separate entries or segmented as spurious morphs (e.g., **"prett + y"**, **"prett + i"**). In the proposed Allomorfessor model, there is a third option, encoding the variations with string edit operations. The results show that the MDL-type prior favors linguistically sensible choices among these options.

Publication IX studies the effect of size and type of the training data for generative models of morphology, in particular for Morfessor. The size of the training corpus has considerable effect on the resulting model. The effect can be simulated by adding a weight function to the likelihood function, and optimizing the parameters of the function for linguistic evaluation can have drastic effect on the performance. It was found that using word frequencies may be useful if their effect on the data likelihood is compensated by decreasing the relative weight of

the likelihood compared with the prior.

In Publication X, Morfessor is extended to semi-supervised learning tasks. With the proposed algorithm, a small annotated training set of words provides large improvements to the results of linguistic evaluations.

The extensions of the model and the algorithms in Publications VIII–X were developed jointly by the present author and Oskar Kohonen. All three publications used a new modular implementation of the Morfessor method, first programmed by the present author and later extended also by Oskar Kohonen. The present author and Oskar Kohonen also designed all the experiments and were the main contributors for writing of the publications.

Finally, Publication XI applies a model similar to Morfessor to the problem of unsupervised learning of phrasal constructions of sentences. The model was tested on a small corpus of stories told by 1–7 year old Finnish children. The experiments demonstrate that Morfessor can be extended to find also other types of constructions than morphemes. The implementation of the method was based on the Morfessor implementation started by the present author, and extended for the new purpose by Oskar Kohonen. The present author also had minor contributions in writing the publication.

1.4 Structure of the thesis

The publications of this thesis encompass many topics in the fields of natural language processing and machine learning. The overview part of the thesis is written as a coherent and rather extensive presentation of these topics and the new contributions to them. It should be easy to follow without consulting the attached publications. While the details of some experiments have been omitted from the overview, it also provides details of some of the proposed methods that have not been described in the publications.

The overview is divided into seven chapters. After this introduction, there are two chapters that provide the central background information on machine learning methodology (Chapter 2) and linguistics concepts and theories (Chapter 3). The former is written from the viewpoint of NLP applications and the latter from the viewpoint of machine learning, so that a researcher or a student of only one of the fields can hopefully get a reasonable overview of the other field. A reader familiar to both of the fields should be able to skip these two chapters; the later chapters will frequently refer to the relevant sections in them.

The next three chapters discuss the three NLP problems on which this thesis concentrates: statistical language modeling (Chapter 4), representation learning (Chapter 5), and lexical unit selection (Chapter 6). In all three chapters, the first sections will introduce the relevant background and prior work, while the last sections will go through the main ideas and original results in the publications of the thesis. In particular, a reader interested only in the new contributions of this thesis should concentrate on Sections 4.4–4.5, 5.4, and 6.3–6.4.

Chapter 7 concludes the overview part of the thesis.

2. Machine learning essentials

In this chapter, the essential topics of machine learning related to the work in this thesis are presented. The presentation is based on several textbooks on machine learning and information theory (Alpaydin, 2004; Bishop, 2006; Cover and Thomas, 2006; MacKay, 2003; Mitchell, 1997). The emphasis is on predictive modeling, parametric models, and unsupervised learning.

There are two central objects in all machine learning settings: a *learner* and *data*. After processing the data, the learner should provide some useful information on it. This information should be useful either for describing the seen data, or for predicting future data, or both. The type of data and the type of the learned information varies in different settings.

In the context of this thesis, data is a collection of observations. Regardless of the collection being a set (unordered) or a sequence (ordered), it is called *data set* and denoted $\boldsymbol{D}$. An observation, or *sample*, may have one or more than one variables (features). Number of variables in a data denotes *dimensionality* of the data. The basic types of data samples are scalars, vectors, labels, strings, and sets (Table 2.1). A standard representation of a data set in $\mathbb{R}^d$ with n observations ($\boldsymbol{D} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$) is a matrix $\mathbf{X} \in \mathbb{R}^{d \times n}$.

Many learning problems can be formalized as *hypothesis selection* problems, where one selects a *point hypothesis* h out of the *hypothesis class* $\mathcal{H}$ based on explicit or implicit assumptions on the nature of the data $\boldsymbol{D}$ (Alpaydin, 2004, Ch. 2). The assumptions are generally called the *inductive bias*. The hypothesis class is assumed to consist of one or more *models* $\mathcal{M}$ such that each model has the same functional form (e.g., second degree polynomial, Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$, or categorical distribution $p(x_i \mid \boldsymbol{\pi}) = \pi_i$). The point hypothesis is then one instance of a particular model (e.g., a polynomial $y = x^2 - 2x + 2$, a Gaussian with zero mean and unit variance $\mathcal{N}(0, 1)$, or a categorical distribu-

Table 2.1. Basic data types for variable X and examples of actual data. Domain ($\mathcal{X}$) is the set of possible instances for the data type. Σ is an alphabet: a non-empty set of possible labels. Σ^n denotes all strings of length n from the alphabet, Σ^* all strings of any length from the alphabet (Kleene closure of Σ), and 2^Σ all the subsets of the alphabet (power set of Σ).

Type	Notation for instance	Domain	Examples of data
Scalar	x	$\mathbb{R}, \mathbb{Z}, \mathbb{Z}_+$	Time; temperature; grade
Vector	$\mathbf{x} = (x_1, x_2, \ldots, x_d)^\mathrm{T}$	$\mathbb{R}^d, \mathbb{Z}^d$	Coordinates; image; frequency spectrum
Label	x	Σ	Letter; class label; product or person ID
String	$\boldsymbol{x} = ab \ldots z$	Σ^n, Σ^*	Word; sentence; DNA sequence
Set	$\mathrm{X} = \{a, b, \ldots, z\}$	2^Σ	Set of labels or IDs

tion with parameters $\boldsymbol{\pi} = (0.5, 0.4, 0.1)$).

2.1 Random variables and distributions

This thesis will mostly focus on probabilistic (stochastic) models, which determine distributions over random variables. Random variables will be written using uppercase font (X) regardless of their type. An observation of X is denoted as x and the set of possible observations as $\mathcal{X}$, unless the type of the variable is specified to be one of those present in Table 2.1. $p(X)$ denotes the probability distribution for X. Unless there is a risk of confusion, the probability $p(X = x)$ is written briefly $p(x)$.

Using the standard notation, $X \mid \boldsymbol{\theta} \sim G(\boldsymbol{\theta})$ (or briefly $X \sim G(\boldsymbol{\theta})$) means that $p(X)$ is determined by the named probability distribution G and its parameters $\boldsymbol{\theta}$. There are two types of random variables and distributions: *discrete* and *continuous*.

Discrete distributions. For a discrete variable X, $p(X)$ is defined by a *probability mass function*. Each $p(X = x)$ gives the probability of the event that X has the value x, and the sum $\sum_{x \in \mathcal{X}} p(X = x) = 1$.

Only a few particular distributions will be referred to in this thesis. The most relevant discrete distribution is the *categorical distribution*. It is parametrized by the probabilities of the finite number of elements in $\mathcal{X}$. If $X \sim \text{Categorical}(\boldsymbol{\pi})$,

$$p(X = x_i \mid \boldsymbol{\pi}) = \pi_i, \tag{2.1}$$

where $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_k)$, $\sum_i \pi_i = 1$, is a vector of parameters. It is an extension of the Bernoulli distribution, for which $k = 2$. Categorical distributions are applied in many NLP models, the elements of $\mathcal{X}$ being, for example, characters or words. The parameters π_i are then their probabilities.

A *multinomial distribution* (or *binomial* if $k = 2$) gives the probability of the outcome for n independent trials from a categorical distribution. If X_i is the number of times event i occurred, the probability of the outcome $x_1, \ldots, x_k$ is

$$p(X_1 = x_1, \ldots, X_k = x_k \mid \boldsymbol{\pi}; n) = \begin{cases} \frac{n!}{x_1! \ldots x_k!} \pi_1^{x_1} \ldots \pi_k^{x_k} & \text{when } \sum_{i=1}^k x_i = n \\ 0 & \text{otherwise.} \end{cases} \tag{2.2}$$

If $X \sim \text{Categorical}(\boldsymbol{\pi})$ and Y is a random vector with $Y_i = \mathrm{I}(X = x_i)$, where $\mathrm{I}(\cdot)$ is a indicator function:

$$\mathrm{I}(A) = \begin{cases} 1, & \text{if } A \text{ is true} \\ 0, & \text{if } A \text{ is false} \end{cases}, \tag{2.3}$$

then $Y \sim \text{Multinomial}(\boldsymbol{\pi}; 1)$. Because of this relation, also the categorical distribution is sometimes called multinomial.

Continuous distributions. For a continuous variable, $p(X)$ is defined by a *probability density function*. The probabilities are non-zero only if they refer to intervals (e.g., $p(x_1 \leq X \leq x_2)$ with $x_1 < x_2$). The integral of $p(X)$ over $\mathcal{X}$ equals one.

The most commonly used continuous distribution is the *normal* or *Gaussian distribution*. For a scalar variable x, it is denoted $\mathcal{N}(\mu, \sigma^2)$ for mean μ and variance

σ^2 and the density function is

$$p(X = x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\Big(-\frac{(x-\mu)^2}{2\sigma^2}\Big). \tag{2.4}$$

A *multivariate Gaussian distribution* $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ for $\mathbf{x} \in \mathbb{R}^k$ is parametrized by the mean vector $\boldsymbol{\mu}$ and the covariance matrix $\boldsymbol{\Sigma}$. The probability density function is

$$p(X = \mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = (2\pi)^{-\frac{k}{2}} |\boldsymbol{\Sigma}|^{-\frac{1}{2}} \exp\Big(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\boldsymbol{\mu})\Big). \tag{2.5}$$

A *Dirichlet distribution* (or *beta distribution* if $k = 2$) is a continuous multivariate distribution with parameters $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_k)$. Its probability density function is

$$p(X_1 = x_1, \ldots, X_k = x_k \mid \boldsymbol{\alpha}) = \frac{1}{B(\boldsymbol{\alpha})} \prod_{i=1}^{k} {x_i}^{\alpha_i - 1}, \tag{2.6}$$

where $x_i \geq 0$ for all i, $\sum_{i=1}^{k} x_i = 1$, and

$$B(\boldsymbol{\alpha}) = \frac{\prod_{i=1}^{k} \Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^{k} \alpha_i)}. \tag{2.7}$$

The Dirichlet distribution is relevant for NLP especially because the samples can be used as the parameters of a categorical or multinomial distribution.

Joint distributions, independence, and marginalization. Consider a *joint distribution* $p(A, B)$ of two variables A and B. The *product rule* states that the joint distribution can be calculated as the product of the probability of one variable and the conditional probability of the other given the first:

$$p(A, B) = p(A)p(B \mid A) = p(B)p(A \mid B) \tag{2.8}$$

The variables are *independent* if $p(A, B) = p(A) \times p(B)$; then $p(A \mid B) = p(A)$.

For a set of random variables $X_1, X_2, \ldots, X_n$, using the product rule multiple times gives

$$p(X_1, \ldots, X_n) = \prod_{i=1}^{n} p(X_i \mid X_1, \ldots, X_{i-1}). \tag{2.9}$$

This is also called the *chain rule*. The set of variables is *independent and identically distributed*, briefly *i.i.d.*, if (1) each X_i has the same probability distribution, and (2) all are mutually independent. The independence assumption gives

$$p(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} p(X_i). \tag{2.10}$$

Marginalization refers to determining the distribution of a single variable $p(A)$ from a joint distribution of several random variables. Accordingly, $p(A)$ is sometimes called the *marginal distribution*. Marginalization of dependent variables is performed by applying the *sum rule*

$$p(A) = \sum_{b \in \mathcal{B}} p(A, B = b) = \sum_{b \in \mathcal{B}} p(B = b)p(A \mid B = b). \tag{2.11}$$

If B is a continuous variable, the sums in Equation 2.11 are replaced by integrals.

Bayes' theorem. If one of the variables is known, the probability of the other variable can be computed using the *Bayes' theorem*:

$$p(A \mid B) = \frac{p(A)p(B \mid A)}{p(B)} = \frac{p(A)p(B \mid A)}{\sum_{a \in \mathcal{A}} p(A = a)p(B \mid A = a)} \tag{2.12}$$

The most important implication of the Bayes' theorem is that given a *prior* belief in A and *likelihood* of an observation $b \in \mathcal{B}$ given A, one can obtain an updated *posterior* probability of A. In machine learning, a common setting is that the parameters of the model are in A and the observed data in B. The Bayes' theorem then shows how the data should modify the probability distribution of the parameters (Section 2.6.6).

An important concept for Bayesian modeling is that of the *conjugate distributions*. If

$$X \mid \boldsymbol{\theta} \sim G(\boldsymbol{\theta}); \qquad \boldsymbol{\theta} \mid \boldsymbol{\xi}_0 \sim H(\boldsymbol{\xi}_0) \tag{2.13}$$

yields to $\boldsymbol{\theta} \mid x, \boldsymbol{\xi}_0 \sim H(\boldsymbol{\xi}_1)$ for some parameters $\boldsymbol{\xi}_1$ (based on $\boldsymbol{\xi}_0$ and x), then H is a conjugate distribution for G. That is, using the conjugate distribution for the prior $p(\boldsymbol{\theta})$ gives the same functional form for the posterior $p(\boldsymbol{\theta} \mid x)$ (Bishop, 2006, Ch. 2). For example, Gaussian distribution, including the multivariate Gaussian, is self-conjugate with respect to the mean parameter: if the prior of the mean is Gaussian, then also the posterior is Gaussian. The Dirichlet distribution is a conjugate distribution for the categorical and multinomial distributions, and thus applied in various Bayesian models developed for language processing.

2.2 Graphical models

Graphical models are tools for representing the interactions between variables.[1] The visualization of a model as a graph helps to analyze the structures of known probabilistic models and design and motivate new ones. A graphical model does not only visualize the structure, but strictly defines the independence assumptions between the variables. Accordingly, some of the complex computations related to the inference problems in the models can be expressed by graph manipulations.

2.2.1 Directed graphical models

Directed graphical models, also called *Bayesian networks* or *belief networks*, are composed of vertices (nodes) V and directed edges (arcs) E. Each vertex corresponds to one random variable. An edge from vertex X to vertex Y indicates that X has direct influence on Y, specified by the conditional probability $p(Y \mid X)$. X is then *parent* of Y. An observed variable is marked by shading the vertex, as x in Figure 2.1(a). Figure 2.1(b) shows a model in which Z is parent of both X and Y. Figure 2.1(c) illustrates a *Markov* model in which each X_i is parent of X_{i+1} for $i < n$ (cf. Section 2.3).

The joint distribution of the variables can be *factorized* by observing the parents $\mathrm{pa}(X_i) = \{X_j : j \in \mathrm{V} \wedge (i, j) \in \mathrm{E}\}$ of each node X_i:

$$p(X_1, \ldots, X_n) = \prod_{i=1}^{n} p(X_i \mid \mathrm{pa}(X_i)). \tag{2.14}$$

[1] For a complete overview, see, e.g., Chapter 8 of Bishop (2006).

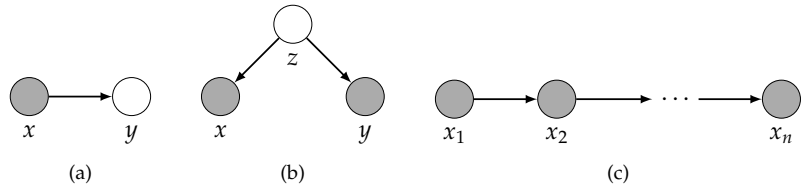


Figure 2.1. Examples of directed graphical models: (a) Y conditioned on observed X, (b) observed X and Y conditioned on Z, and (c) a first-order Markov model.

For example, the model in Figure 2.1(b) indicates

$$p(X, Y, Z) = p(Z)p(X \mid Z)p(Y \mid Z). \tag{2.15}$$

Then X and Y are *conditionally independent* given Z:

$$p(X, Y \mid Z) = \frac{p(X, Y, Z)}{p(Z)} = p(X \mid Z)p(Y \mid Z). \tag{2.16}$$

In general, finding conditionally independent sets of nodes in a directed model requires a graphical test called *d-separation*.

2.2.2 Undirected graphical models

Undirected graphical models or *Markov random fields* have, again, variables as vertices V and direct dependences of the variables as edges E. However, the edges are undirected so that the direction of the influence is not specified. For any two variables X_i and X_j for which $\{i, j\} \notin \mathrm{E}$, X_i is conditionally independent of X_j given all its neighbors $\mathrm{ne}(X_i) = \{X_k : k \in \mathrm{V} \wedge \{i, k\} \in \mathrm{E}\}$.

The joint distribution of an undirected model is factorized by maximal *cliques*, sets of vertices for which there is an edge between every pair of vertex. Each maximal clique $\mathrm{C} \subseteq \mathrm{V}$ has its *potential function* $\phi_{\mathrm{C}} : \mathcal{X}^{|\mathrm{C}|} \mapsto \mathbb{R}_+$, and the joint distribution is defined by the product of the potential functions:

$$p(X_1 = x_1, \ldots, X_n = x_n) = \frac{1}{Z} \prod_{\mathrm{C}} \phi_{\mathrm{C}}(\mathrm{x}_{\mathrm{C}}), \tag{2.17}$$

where x_{C} is the set of variables in clique C. For example, the graph in Figure 2.2 has three maximal cliques $\{x_1, x_2\}$, $\{x_2, x_3, x_4\}$, and $\{x_4, x_5\}$ and is thus factored by

$$p(x_1, \ldots, x_n) = \frac{1}{Z} \phi_{1,2}(x_1, x_2)\phi_{2,3,4}(x_2, x_3, x_4)\phi_{4,5}(x_4, x_5). \tag{2.18}$$

The *partition function*

$$Z = \sum_{\mathrm{x}} \prod_{\mathrm{C}} \phi_{\mathrm{C}}(\mathrm{x}_{\mathrm{C}}) \tag{2.19}$$

ensures that the distribution is correctly normalized. The need of explicit normalization is a major drawback for undirected models: calculating the sum over all combinations of states of all variables in the model is often infeasible.

The form of the potential functions is free, but due to the constraint on non-negative values, they are commonly expressed as exponentials of *energy functions* $E(\mathrm{x}_{\mathrm{C}})$:

$$\phi_{\mathrm{C}}(\mathrm{x}_{\mathrm{C}}) = \exp\big(-E(\mathrm{x}_{\mathrm{C}})\big). \tag{2.20}$$

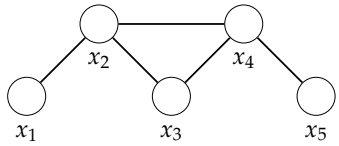


Figure 2.2. Example of an undirected graphical model.

The joint distribution is then

$$p(x_1, \ldots, x_n) = \frac{1}{Z} \exp\Big(-\sum_{\mathrm{C}} E(\mathrm{x}_\mathrm{C}) \Big). \tag{2.21}$$

The models of this form are referred to as *exponential* models. Exponential models can also be used for conditional distributions—including those in directed graphical models—by making the partition function dependent on the observed variables.

The energy functions can be arbitrary. However, an important family of models, *log-linear* models, have a linear energy function. A classic result by Jaynes (1957) is that if all information from an unknown distribution $p(X)$ are the expected values of functions $f_j(x)$, $j = 1, \ldots, N$, the model that has the maximal entropy (see Section 2.4) has the energy function:

$$E(x) = \sum_{j=1}^{N} \lambda_j f_j(x), \tag{2.22}$$

where λ_j are constant parameters to be solved. Accordingly, this kind of log-linear models are called *maximum-entropy* (ME, MaxEnt) models (Ratnaparkhi, 1996; McCallum et al., 2000; Rosenfeld et al., 2001). Some specific ME models are discussed in Sections 2.3 and 4.3.2. The maximization of entropy means that the model is as non-committal as possible with regard to missing information (Jaynes, 1957). In other words, it satisfies the given constraints, but assumes nothing else.

2.3 Markov models and finite-state machines

This section considers particular types of graphical models, Markov models, that have many applications in NLP. Markov models assume a particular type of a *random process*, in which a sequence of random variables have a very limited dependency structure. In addition to the standard Markov model, two important extensions, *hidden Markov models* and *conditional Markov models* are discussed. Markov models are also related to *finite-state machines*, behavioral computational models with the Markov property.

2.3.1 Markov processes

Random (or stochastic) process is a sequence of random variables $\{X_i\}$, usually indexed in time (Cover and Thomas, 2006, Ch. 4). An output of the process, $(x_1, x_2, \ldots, x_n) \in \mathcal{X}^n$, can be considered as a time series, and each X_i is the *state* of the process at time i. In a *stationary* process, the joint distribution of the variables is invariant with respect to shifts in the time index:

$$p(X_1, \ldots, X_n) = p(X_{l+1}, \ldots, X_{l+n}) \tag{2.23}$$

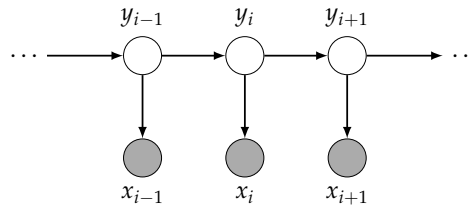


Figure 2.3. Graphical representation of hidden Markov model.

for every length n and time shift l.

In general, the dependencies between the variables are arbitrary, and only the chain rule can be applied to compute the joint distribution $p(X_1, \ldots, X_n)$. The other extreme is the i.i.d. assumption. Between these extremes, there are cases in which each X_i has a limited dependence on the previous variables. One specific case is the *Markov process*, for which each random variable X_i depends only on the preceding variable X_{i-1}. In other words, the process is "memoryless": the future states depend only on the current state, not any states the preceded it. This is called the Markov property. If the conditional probability $p(X_i \mid X_{i-1})$ does not depend on i, the process is time invariant and stationary.

The standard Markov process can be extended to the k^{th} order Markov process, for which

$$p(X_1, \ldots, X_n) = \prod_{i=1}^{n} p(X_i \mid X_{i-k}, \ldots, X_{i-1}). \tag{2.24}$$

In the case of a discrete $\mathcal{X}$, the process is also called a *Markov chain*.

2.3.2 Markov model

The Markov model is a directed graphical model that assumes that the observed variable is generated by a Markov process. A graphical representation of the first-order Markov model is shown in Figure 2.1(c). A typical example of such a model is the *bigram language model*, which predicts the next word in a text based on the previously seen word (cf. Section 4.2). The parameters of a Markov model include only the transition probabilities $p(x_i \mid x_j)$ between all $x_i, x_j \in \mathcal{X}$. The estimation of the parameters is often simple, because the model does not have any unobserved variables.

2.3.3 Hidden Markov models

The hidden Markov model (HMM) is an extension of the Markov model, in which the state sequence $Y_1, \ldots, Y_n$ is not observed. Instead, each state y_i emits an observation x_i according to the conditional distribution $p(X \mid Y_i)$. The joint distribution of the observations is thus

$$p(X_1, \ldots, X_n) = \prod_{i=1}^{n} p(Y_i \mid Y_{i-1}) p(X_i \mid Y_i). \tag{2.25}$$

Figure 2.3 shows the graphical representation of an HMM.

HMMs are popular for modeling sequential data such as speech and text. There are two basic algorithms for using an HMM: the *Viterbi algorithm* (Viterbi, 1967; Forney, 1973) computes the most likely sequence of the hidden states for a

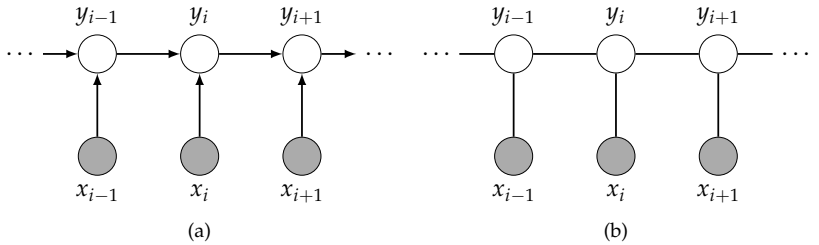


Figure 2.4. Graphical models with hidden Markov model structure: (a) maximum-entropy Markov model, and (b) linear-chain CRF.

sequence of observed events, and the *forward algorithm* computes the probability of a sequence of observed events. Both are dynamic programming algorithms that have time complexity of $O(nM^2)$, where n is the number of samples and M is the number of states. If the training data includes both the states and the observations, the parameters of an HMM can be estimated directly. If only observations are available, HMMs can be trained with the *Baum-Welch algorithm* (Baum, 1972), which is an adaptation of the general expectation-maximization (EM) algorithm described later in Section 2.8.5.

2.3.4 Conditional Markov models

During recent years, there has been an increasing amount of research on models that have the same structure as the HMM but estimate only the conditional distribution $p(Y_i \mid X_i, Y_{i-1})$. The benefit of the conditional model is that if the set of observations $\mathcal{X}$ is large, the model can use arbitrary feature functions $f_j(x_i, y_{i-1}, y_i)$ based on the observations and states, instead of using the observations x_i themselves. For example, if $\mathbf{x}$ is a document, certain words of the documents could be used as features, without making any assumptions on how the documents are generated.

A directed graphical model for $p(Y_i \mid X_i, Y_{i-1})$ is shown in Figure 2.4(a). Using an exponential model (Equation 2.21) with linear energy function gives

$$p(y_i \mid x_i, y_{i-1}) = \frac{1}{Z(x_i, y_{i-1})} \exp\Big(\sum_{j=1}^{N} \lambda_j f_j(x_i, y_{i-1}, y_i)\Big). \tag{2.26}$$

If the weight parameters λ_j are learned by maximizing the entropy of the model, this is the *maximum-entropy Markov model* (MEMM), proposed by Ratnaparkhi (1996) for POS tagging and McCallum et al. (2000) for document segmentation.

Lafferty et al. (2001) note that MEMMs suffer the property that all probability mass arriving to a state has to be distributed to the successor states, independent of the observation from the state. This so-called *label bias* problem is avoided using a family of undirected models called *conditional random fields* (CRFs). The *linear-chain CRF* model in Figure 2.4(b) corresponds to the HMM and MEMM models. It defines the probability for the complete sequences of states $\mathbf{y} = (y_1, \ldots, y_n)^{\mathrm{T}}$ given the sequence of observations $\mathbf{x} = (x_1, \ldots, x_n)^{\mathrm{T}}$:

$$p(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\Big(\sum_{i=1}^{n} \sum_{j=1}^{N} \lambda_j f_j(x_i, y_{i-1}, y_i)\Big), \tag{2.27}$$

where the transition and emission cliques for each time index i are modeled

using one set of N feature functions $f_j(\cdot)$. In contrast to MEMM, the partition function is global.

CRFs have been applied to several sequential language processing tasks, including POS tagging (Lafferty et al., 2001), shallow parsing (Sha and Pereira, 2003), and Chinese word segmentation (Peng et al., 2004). The conditional models can be estimated with general-purpose convex optimization techniques (see, e.g., Sha and Pereira, 2003). The optimization involves estimating expected frequencies of the features with dynamic programming algorithms similar to the Baum-Welch algorithm used for HMMs.

2.3.5 Finite-state machines

The HMM is a special case of a *weighted finite-state acceptor* (WFSA), one type of a finite-state machine.[2] A non-weighted finite-state acceptor (FSA) has a set of states Q, of which one is an *initial state* and one or more are *final states*. The set of transitions $E \subseteq Q \times Q$ between the states are associated with labels from set Σ. An FSA accepts a sequences of labels, if the transitions in some *path* from the initial state to one of the final states produces the sequence. The set of sequences accepted by FSAs are also called *regular languages*; other types of formal languages, context-free languages, are discussed in Section 3.1.8. Models that generate accepted outputs of formal languages are often called *generative grammars*.[3]

WFSA includes a weight for each state transition. The weights and their operations can be based on any algebraic structure called *semiring* (Roark and Sproat, 2007, p. 11); for an HMM, it is the probabilistic semiring $(\mathbb{R}, +, \times, 0, 1)$. The weight associated with a path is the product of an initial weight, weights of the transitions, and a final weight. The weight associated with a sequence of labels is the sum of the weights of all successful paths labeled with the sequence. Thus WFSA provides a mapping from label sequences to weights. One common application for WFSAs is to store the best output hypotheses from a speech recognition or machine translation system to an easily-accessible structure (called *lattice* or *confusion network*) for further processing, such as re-ranking with computationally intensive models.

Finite-state transducers (FSTs) and *weighted finite-state transducers* (WFSTs) replace the single transition labels in FSAs and WFSAs by pairs of input and output labels. A FST represents binary relations between the input and output sequences. That is, either the input–output pair is accepted or rejected. A WFST associates these relations to weights. Figure 2.5 shows an example of a WFST that maps between letters and phonemes of a word.

FSTs have been particularly successful in computational modeling of phonology (Kaplan and Kay, 1994) and morphology (Koskenniemi, 1983; Karttunen and Beesley, 2005). While FSTs are usually constructed from rules defined by hand, WFSTs can be learned automatically similar to HMMs (Clark, 2002; Chiang et al., 2010). WFSTs have been applied in many NLP problems, including speech recognition (Mohri et al., 2002), word segmentation (Sproat et al., 1996), machine transliteration (Knight and Graehl, 1998), and statistical machine translation (Kumar et al., 2006). Multiple finite-state acceptors and transducers can

[2] For an introduction on finite-state machines, see, e.g., Chapter 1 by Roark and Sproat (2007), or Chapter 2 by Jurafsky and Martin (2008).

[3] Generative grammars should not be confused with probabilistic generative models discussed later in this chapter.

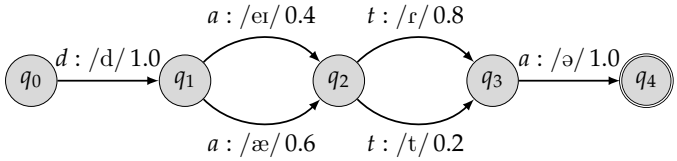


Figure 2.5. Weighted finite-state transducer for different pronunciations of word **data** (example from Mohri et al., 2002). Transitions are annotated with the label pair and the weight (probability) of the transition. The pronunciations and the probabilities of the respective paths are /deɪrə/ ($p = 0.32$), /deɪtə/ ($p = 0.08$), /dærə/ ($p = 0.48$), and /dætə/ ($p = 0.12$).

be combined, for example, for speech-to-speech translation (Mathias and Byrne, 2006).

2.4 Information theory

Information theory was developed by Claude Shannon (1916–2001) for finding the limits in data compression (*source coding theorem*) and transmitting data over an imperfect communication channel (*noisy channel coding theorem*). The main ideas were presented in 1948. Shannon proposed a number of measures to calculate the amount of uncertainty in random variables, including *entropy*, which was before that applied in statistical mechanics and thermodynamics. These basic measures are defined next, followed by brief descriptions of Shannon's coding theorems, a presentation of information theoretic measures for calculating divergences between two distributions, and finally a short introduction to algorithmic information theory. While all the measures can be defined both for discrete and continuous variables, only the discrete versions are given here.[4]

2.4.1 Quantities of information and uncertainty

Let X be a random variable. The *Shannon information content* or *self-information* of a single sample x of X is

$$I(x) = \log(\frac{1}{p(x)}) = -\log p(x). \tag{2.28}$$

The larger the probability of x is, the less information is gained from it; if $p(x) = 1$, the result is already known and there is no gain of information. The entropy (or self-information) of X can be defined as the expected value of the self-information over the possible outcomes $\mathcal{X}$:

$$H(X) = E_X[I(X)] = -\sum_{x \in \mathcal{X}} p(x) \log p(x). \tag{2.29}$$

Entropy is usually measured in *bits*, if using two as the base of the logarithm, or *nats*, if using the natural logarithm. By defining $0 \log 0 \equiv 0$, the value is always non-negative.

Next, consider two variables X and Y. The *joint entropy* of X and Y is

$$H(X, Y) = -\sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x, y) \log p(x, y). \tag{2.30}$$

[4] For more complete descriptions on the topics of information theory, see MacKay (2003) or Cover and Thomas (2006).

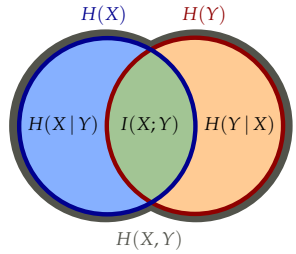


Figure 2.6. A Venn diagram illustrating the relations between individual entropies $H(X)$ and $H(Y)$, joint entropy $H(X,Y)$, conditional entropies $H(X \mid Y)$ and $H(Y \mid X)$, and mutual information $I(X;Y)$.

The joint entropy equals the sum of the individual entropies $H(X) + H(Y)$ only if the two variables are independent; otherwise it is less than $H(X) + H(Y)$. *Conditional entropy* is the expected entropy of X given Y:

$$\begin{aligned} H(X \mid Y) &= -\sum_{y \in \mathcal{Y}} p(y) \sum_{x \in \mathcal{X}} p(x \mid y) \log p(x \mid y) \\ &= -\sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x,y) \log \frac{p(x,y)}{p(y)}. \end{aligned} \tag{2.31}$$

Conditional entropy equals the difference $H(X,Y) - H(Y)$. It is always non-negative and zero only if X and Y are fully dependent.

The difference of joint entropy to the sum of the marginal entropies is called *mutual information*:

$$\begin{aligned} I(X;Y) &= H(X) + H(Y) - H(X,Y) \\ &= H(Y) - H(Y \mid X) \\ &= H(X) - H(X \mid Y) \\ &= \sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}. \end{aligned} \tag{2.32}$$

Mutual information measures the amount of information that X contains about Y and vice versa. It is zero only if the two variables are independent, and grows both with the degree on dependence and the entropy of the variables. Trivially, the mutual information $I(X;X)$ equals the self-information $H(X)$.

The relations between the individual entropies, joint entropy, conditional entropies, and mutual information are analogous to the relations between two sets, their union, differences, and intersection, respectively. This is illustrated by the Venn diagram in Figure 2.6.

2.4.2 Source coding theorem and Kraft's inequality

The basic motivation for the definitions of self-information and entropy is that they measure how well the outcome of a random source can be compressed. Consider a source $p(X)$ that generates instances of a variable X over a finite set $\mathcal{X}$. A *code* $C : \mathcal{X} \mapsto \mathcal{Y}$ maps each $x \in \mathcal{X}$ into a sequence $\boldsymbol{y} \in \mathcal{Y}$; the process is called *encoding*. The target sequence $\boldsymbol{y}$ is usually set to be binary: $\mathcal{Y} \subseteq \{0,1\}^*$. The *code length* for x, $l_C(x) = |C(x)|$, is then measured in bits. In *lossless* encoding, C has an inverse code $C^{-1} : \mathcal{Y} \mapsto \mathcal{X}$ such that $x = C^{-1}(C(x))$. The process of mapping from $\mathcal{Y}$ back to $\mathcal{X}$ is called *decoding*. Lossless source coding is illustrated in Figure 2.7.

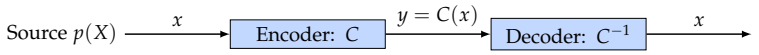


Figure 2.7. Lossless source coding.

Given an i.i.d. random source that generates a sequence $x_1, x_2, \ldots, x_N$, Shannon's source coding theorem (Shannon, 1948) states that for any $\epsilon > 0$ and a large enough N, there exists a lossless source coding such that the sequence can be encoded in $N \times (H(X) + \epsilon)$ bits. Conversely, there is no recoverable source coding that can compress the sequence to fewer than $N \times (H(X) + \epsilon)$ bits. Thus entropy determines the average number of bits required to encode the outcome of the random variable using the *optimal coding* $\hat{C}$, which uses code lengths $l_{\hat{C}}(x) = -\log_2 p(x)$. For example, the result of an 8-sided dice can be optimally encoded in $-8 \times \frac{1}{8} \log_2 \frac{1}{8} = 3$ bits.

Encoders and decoders should consider not only single items x but sequences $x_1, \ldots, x_n$ ($n < N$). The *extension* $C^{(n)}$ of code C maps a sequence $(x_1, \ldots, x_n) \in \mathcal{X}^n$ into $C(x_1) \ldots C(x_n) \in \mathcal{Y}^*$. If the mapping by the extension $C^{(n)}$ is invertible for any n, the code C is *uniquely decodable*. The inequality theorem by Kraft (1949) states that for any uniquely decodable code C with alphabet size V,

$$\sum_{x \in \mathcal{X}} V^{-l_C(x)} \leq 1. \tag{2.33}$$

The converse is also true: for any given set of natural numbers $l_1, l_2, \ldots, l_n$ such that $\sum_{i=1}^{n} V^{-l_i} \leq 1$, there exists a uniquely decodable code over an alphabet of size V. With strict equality, the code is called *complete*. The optimal coding for any given distribution $p(X)$ is complete: $\sum_{x \in \mathcal{X}} 2^{-l_{\hat{C}}(x)} = \sum_{x \in \mathcal{X}} 2^{\log_2 p(x)} = 1$. Note that this means not only that we can associate coding lengths with any probability distribution (Shannon's source coding theorem) but also that we can associate a probability distribution with any uniquely decodable code (Kraft's inequality). The probability distribution, however, may be *defective* ($\sum_{x \in \mathcal{X}} p(x) < 1$) if the code is not complete. The relation of complete codes and probability distributions connects the minimum description length principle (Section 2.6.8) to probabilistic model selection and parameter estimation methods.

In practice, the codes are usually limited to *prefix codes* that use a set of sequences (code words) in which there is no valid sequence in the system that is a prefix of any other valid sequence in the set. The advantage of prefix codes is that a sequence of code words $C(x_1)C(x_2) \ldots C(x_n)$ can be decoded without any marker between the code words. That is, prefix codes are uniquely decodable. The well-known algorithm by Huffman (1952) gives the optimal binary prefix code for a given distribution. For Huffman codes, the expected code length $E_X[l_C(x)] < H(X) + 1$.

2.4.3 Noisy channel coding theorem

Noisy channel model (Figure 2.8) considers the case where an encoded signal is exposed to random noise during the transmission. A message from a information source is encoded into signal x, and then transmitted over the noisy channel. The received signal y corrupted by the channel noise is decoded back into the original message.

Shannon's (1948) noisy channel coding theorem states that the maximal channel capacity $C = \sup_{p(X)} I(X; Y)$. That is, for information transmitted at rate R, if $R < C$, there exists codes that allow the probability of error at decoding to be

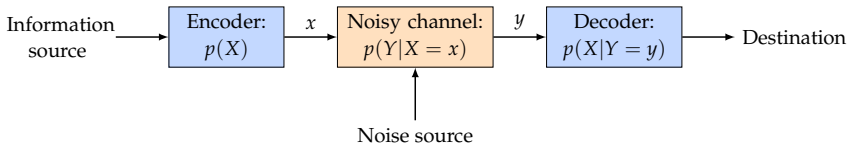


Figure 2.8. Noisy channel model.

arbitrarily small. However, if $R > C$, an arbitrarily small probability of error is not achievable, and the errors increase with the rate R.

Selecting the x that maximizes $p(X = x \,|\, Y = y)$ for a given y is called the *decoding problem*. As many NLP problems can be viewed as decoding problems, they are often illustrated with the noisy channel model. For example, in speech recognition, Y would be the speech signal and X would be a transcription of the speech (see Chapter 4).

2.4.4 Divergence of two distributions

The quantities above consider one probability distribution per each variable. However, in many machine learning problems, one has to consider many different distributions for one set of variables (e.g., one distribution per each point hypothesis). There exist several information theoretic measures that provide means of determining how similar or dissimilar two distributions are. The measures of dissimilarity for probabilities are called *divergences*.

First, a common situation is that the true distribution $p(x)$ is often approximated by another distribution $q(x)$. The *cross-entropy* of $q(x)$ is its expected self-information given the true distribution:

$$H_q(X) = E_X[-\log q(x)] = -\sum_{x \in \mathcal{X}} p(x) \log q(x). \tag{2.34}$$

In other words, it measures the number of bits required to encode the outcome of $p(x)$ with a coding based on $q(x)$. Obviously, $H_q(X) \geq H(X)$ and equal only if $p(x) = q(x)$ for all x. Moreover, it reaches infinity for any x such that $p(x) > 0$ and $q(x) = 0$.

Relative entropy or *Kullback–Leibler divergence* is the difference between the cross-entropy and the true entropy of $p(x)$:

$$\begin{aligned} D(q \,||\, p) &= -\sum_{x \in \mathcal{X}} p(x) \log q(x) + \sum_{x \in \mathcal{X}} p(x) \log p(x) \\ &= E_X\Big[\log \frac{p(x)}{q(x)}\Big] \end{aligned} \tag{2.35}$$

That is, it measures how many more bits are required on average if $q(x)$ is used to model the data generated by $p(x)$. Similar to cross-entropy, it is asymmetric, non-negative, and zero only if $q(x) = p(x)$ for all x.

Finally, *information radius* or *Jensen–Shannon divergence* is defined as the mean of the relative entropies between the two distributions and their average distribution $0.5 \times (p(x) + q(x))$:

$$\begin{aligned} \mathrm{IRad}(p \,\|\, q) &= \frac{1}{2} D\Big(p \,\Big\|\, \frac{p+q}{2}\Big) + \frac{1}{2} D\Big(q \,\Big\|\, \frac{p+q}{2}\Big) \\ &= \frac{1}{2} \sum_{x \in \mathcal{X}} \Big[p(x) \log \frac{2p(x)}{p(x)+q(x)} + q(x) \log \frac{2q(x)}{p(x)+q(x)}\Big]. \end{aligned} \tag{2.36}$$

In contrast to relative entropy, information radius is bounded ($0 \leq \text{IRad} \leq 1$) and symmetric with respect to the two distributions.

Because both cross-entropy and relative entropy measure how efficiently new data from $p(x)$ can be compressed and communicated with $q(x)$, they are theoretically well-motivated cost functions for a density estimation task (Ghahramani, 2004). However, the real distribution $p(x)$ is seldom known, and empirical estimates of the divergences have to be applied instead.

A simple way to estimate the divergences is to give an equal weight for every observed data sample x_i. For example, the empirical cross-entropy for distribution $q(x)$ given data $\boldsymbol{D}$ is

$$\tilde{H}_q(\boldsymbol{D}) = -\frac{1}{n}\sum_{i=1}^{n} \log q(x_i), \tag{2.37}$$

where $n = |\boldsymbol{D}|$ is the number of samples. The complement of this measure is *average log-likelihood*. A related measure, applied commonly in statistical language modeling, is *perplexity*:

$$\text{Perp}_q(\boldsymbol{D}) = 2^{\tilde{H}_q(\boldsymbol{D})} = \Big(\prod_{i=1}^{n} q(x_i)\Big)^{-\frac{1}{n}}. \tag{2.38}$$

It can be interpreted as the average branching factor for the model. For example, if the data samples are throws of a 6-sided dice, there are six possible choices for each sample. For the correct model, perplexity is 6, and any incorrect model will give a perplexity larger than that because $H_q(X) \geq H(X)$.

2.4.5 Algorithmic information theory

Algorithmic information theory combines the ideas from Shannon's information theory with the computability theory by Alan Turing (1912–1954). Instead of random variables and their distributions, it concerns the encoding of arbitrary data sets with general-purpose computer programs—programs that can simulate a Turing machine, a theoretical symbol-manipulation device described by Turing (1936).

The central notion in algorithmic information theory is *algorithmic complexity*, also known as *Kolmogorov complexity*, developed by Solomonoff (1964a,b) Kolmogorov (1965), and Chaitin (1969). The algorithmic complexity $K(\boldsymbol{D})$ of data $\boldsymbol{D}$ is defined as the number of bits required for the shortest universal Turning machine (i.e., computer program) that writes the data and then halts. This can be seen as compression: $\boldsymbol{D}$ is encoded into the program, and decoding means simply running the program code.

For completely random data, $K(\boldsymbol{D})$ cannot be any shorter than the data itself. However, very regular data, such as a long sequence of the same symbol, can be encoded very efficiently (e.g., by storing the symbol and the length of the sequence).

The algorithmic complexity can be used to assign probabilities to arbitrary data sets (Vitanyi and Li, 2000): Consider data $\boldsymbol{D}$ as a random variable. The probability of a data set can be defined by the probability of producing the shortest program code that generates the data. Probability of one bit is $\frac{1}{2}$, so $p(\boldsymbol{D}) = (\frac{1}{2})^{K(\boldsymbol{D})} = 2^{-K(\boldsymbol{D})}$. If the codes are prefix-free—a universal Turing machine with this property exists—Kraft's inequality confirms that the sum over all codes does not exceed one.

Algorithmic complexity is not directly applicable to learning problems, since it is impossible to design an algorithm that computes the Kolmogorov complexity of a given data set. Still, it has served as foundation for other methods such as the minimum description length principle (discussed in Section 2.6.8) and normalized compression distance (Li et al., 2004). Using a coding scheme less general than one based on a programming language provides the assumptions (inductive bias) needed for learning.

2.5 Learning setups

The problems in machine learning can be divided into different types according to what kind of data samples the learner gets and what it should do with them. The basic division is between *supervised* and *unsupervised* learning. Both *semi-supervised* and *reinforcement learning* problems can be seen to be the middle of the two, but while the former can be solved with relatively straightforward extensions of unsupervised and supervised learning algorithms, the latter requires a very different approach. In *multi-view learning*, there are two or more independent sets of features, views, of the same data, while the task may be either supervised, semi-supervised, or unsupervised. *Multi-task learning* and *transfer learning* concern setups in which there are multiple tasks to consider together.

2.5.1 Supervised learning

In supervised learning, the learner gets input–output pairs (x, y) and should learn to predict the output y given the input x. The two main types of supervised learning problems are *regression*, where the output is a numeric scalar or vector, and *classification*, where the output is a categorical label.

For supervised learning problems, there are two approaches, *generative* and *discriminative* modeling. The former approach is to build a model that is assumed to generate the observed data, that is, both the inputs and outputs. The latter approach is to solve the task (regression or classification) directly and neglect the aspect of data generation. In probabilistic models, the distinction is illustrated by whether there is a model for the joint distribution $p(X, Y)$ (generative models) or only for the conditional distribution $p(Y \mid X)$ (discriminative models).

Unlike generative modeling, discriminative modeling does not need to be probabilistic. A standard non-probabilistic approach is to find a function $f : \mathcal{X} \mapsto \mathcal{Y}$, where $\mathcal{X}$ is the input space and $\mathcal{Y}$ is the output space. In multivariate linear regression $f(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + b$, where weight vector $\mathbf{w}$ and bias term b are determined from the data. A linear binary classifier of the same functional form can be considered as a very simple artificial neural network, *perceptron* (Rosenblatt, 1958). More complicated models include, for example, *multi-layer perceptrons* (MLP) that combine multiple perceptrons with non-linear activation functions. MLPs work as universal approximators (Hornik et al., 1989): they can approximate arbitrary functions from $\mathcal{X}$ to $\mathcal{Y}$.

Another common non-probabilistic approach for discriminative modeling is to learn *decision trees* (e.g., Quinlan, 1986). Decision trees divide the input space $\mathcal{X}$ into equivalence classes that predict the same output y by making consecutive questions on the features of x. Each question forms a new branch in the tree, and

the leaf nodes correspond to different outcomes y.

2.5.2 Unsupervised learning

In the case of unsupervised learning, there are no samples of the desired output available. Instead, the learner has to assume some statistical regularities in the input data and use them to create a new *representation* for the input. The representation may then be used to predicting new inputs, communicating or compressing the observed input, or as an intermediate result for other learning problems. In other words, the desired output is the representation, and different goals for the representation lead to different unsupervised learning problems.

The learned representation can also be seen as a model of the data. In the case that the model is probabilistic, that is, it gives an estimate for the distribution $p(X)$ of the input variable X, the task is called *density estimation*. If the type of the probability distribution is known, the goal is simply to find the best-fitting parameters of the distribution. All probabilistic generative models do density estimation by definition. This includes many types of directed and undirected graphical models, as well as neural network inspired models such as deep belief networks (Hinton, 2009). The majority of density estimation in NLP is based on Markov and hidden Markov models of various order due to their low computational complexity. However, there are also some successful experiments with more complicated models such as log-linear models (Rosenfeld et al., 2001) and neural networks (Bengio et al., 2003; Hinton and Salakhutdinov, 2011).

Another typical case of unsupervised learning is *clustering*. The set of input samples X are assumed to have such a structure that within disjoint subsets $\mathrm{C}_i \subset \mathrm{X}$, where $\mathrm{C}_i \cap \mathrm{C}_j = \emptyset \quad \forall i \neq j$, the samples are more similar than the samples in different subsets. The desired output is the subsets $\mathrm{C}_1, \mathrm{C}_2, \ldots, \mathrm{C}_K$. Equivalently, the output space $\mathcal{Y}$ can be defined as a set of arbitrary cluster labels and the algorithm should return one $y \in \mathcal{Y}$ per each input sample, similarly to a supervised classification problem. The number of clusters K may be given beforehand or has to be determined from the data. Two very common clustering algorithms, hierarchical clustering and K-means, are described in Section 2.8.4. Clustering can be used for lossy compression: *vector quantization* refers to replacing the individual samples vectors by the mean vector over all samples in the same cluster.

A third large set of unsupervised problems are those of *blind source separation* (BSS) (for overviews, see, e.g., Cardoso, 1998; Hyvärinen et al., 2001). In BSS, the input variables X_j are assumed to be mixed signals of the k original source signals $y_1, \ldots, y_k$:

$$x_j = g_j(y_1, \ldots, y_k). \tag{2.39}$$

If the mixing functions $g_j(\cdot)$ are linear, then

$$x_j = \sum_{i=1}^{k} a_{ji} y_i. \tag{2.40}$$

The goal is to identify the source signals and sometimes the number of sources k. The separation of the sources is possible by assuming that they have certain properties. For example, the sources are assumed to be uncorrelated in *principal component analysis* (PCA), and independent in *independent component analysis* (ICA). The standard way to calculate PCA by eigenvalue or singular value decomposition is described in Section 2.8.2. A popular fixed-point algorithm for

ICA was presented by Hyvärinen and Oja (1997). Both PCA and ICA can also be solved by simple neural network models (Oja, 1982; Hyvärinen and Oja, 1996).

Equation 2.40 can be equivalently written as $\mathbf{X} = \mathbf{AY}$, where $\mathbf{X} \in \mathbb{R}^{d \times n}$ and $\mathbf{Y} \in \mathbb{R}^{k \times n}$ are input and output matrices, respectively. This task combines with *matrix factorization* methods. Especially if $\mathbf{X}$ is positive, it is useful also to restrict one or both of the factors to be non-negative (Paatero and Tapper, 1994; Lee and Seung, 1999). If $\mathbf{A} \in \mathbb{R}^{d \times k}$ is non-negative, its rows can be considered as *membership degrees* for x_j belonging to each y_i, or conditional probabilities $p(y_i \mid x_j)$ if the rows sum up to one. The task is then *soft* or *fuzzy* clustering in contrast to the *hard* clustering discussed above.

To solve Equation 2.40, one should find $\mathbf{W}$ such that $\mathbf{Y} = \mathbf{WX}$. If $k = d$ and $\mathbf{A}$ is invertible, $\mathbf{W} = \mathbf{A}^{-1}$. When $k < d$, the mapping from $\mathbf{X}$ to $\mathbf{Y}$ can been seen as *dimensionality reduction*. The applications of PCA and ICA to dimensionality reduction for vector space models of language is discussed later in Section 5.1.

The general task in dimensionality reduction is to find a function $f(\cdot)$ that maps the original samples into a low-dimensional space while preserving the pairwise distances of the samples:

$$f : \mathcal{X} \mapsto \mathcal{Y} \quad \text{s.t.} \quad d_x(\mathbf{x}_i, \mathbf{x}_j) \approx d_y(f(\mathbf{x}_i), f(\mathbf{x}_j)), \tag{2.41}$$

where $d_x(\cdot, \cdot)$ and $d_y(\cdot, \cdot)$ are distance or similarity functions. For non-linear dimensionality reduction, it is assumed that the data samples are in a low-dimensional manifold that is non-linearly embedded within the high-dimensional space. Finding the manifold requires non-linear dimensionality reduction, also known as *manifold learning*. Many linear methods such as PCA have non-linear variants; they can, for example, be extended with kernel functions (Section 2.7.2).

One type of artificial neural networks suited for non-linear dimensionality reduction is the *self-organizing map* (SOM) by Kohonen (1982, 1995). The SOM is usually constructed as a two-dimensional, one layer network. Each neuron, called *map unit*, is connected to adjacent map units thus forming its neighborhood. The map is trained with competitive learning, while having local cooperation between neighboring units. As a result, adjacent units model similar data samples and the samples appear close to each other in the map. The SOM has been applied to several NLP applications and problems, including syntactic and semantic categorization of words (Honkela et al., 1995), speech recognition (Kurimo, 1997, 2002), text retrieval and mining (Kaski et al., 1998; Kohonen et al., 2000; Lagus et al., 2004), statistical language modeling (Kurimo and Lagus, 2002; Lagus and Kurimo, 2002), and word sense disambiguation (Lindén and Lagus, 2002; Lindén, 2003).

2.5.3 Semi-supervised learning

Obviously, there are settings between the fully supervised and unsupervised cases. These are called either semi-supervised or partially supervised settings. In the most common setting, the learner gets input–output pairs for a part of the training data, but only the input values for the rest. The first part is often called *labeled* and the second part *unlabeled*, regardless of the type of the output variable. Usually the amount of labeled data is remarkably smaller than the amount of unlabeled data, so that supervised algorithms do not have enough data for high accuracy. The algorithms for the semi-supervised problems are

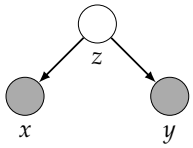


Figure 2.9. Graphical model for two-view learning and canonical correlation analysis.

usually extensions of either supervised or unsupervised ones (see, e.g., Zhu, 2005).

In the setting above, the data samples were separated into known (labeled) and unknown (unlabeled) samples. In contrast, there can also be a situation in which a part of the output is known for all training samples, but another part is not. For example, the training data for a task of learning morphology may include stems of the words as input, but the output should include also all the prefixes and suffixes of the words. To distinguish between these two cases, here the term "semi-supervised" is used only for the former, and the term "partially supervised" for the latter.

2.5.4 Multi-view learning

Multi-view learning refers to a setup where observed data is from multiple independent views from the same source. Such sources could, for example, be the speech signal and video of the lip movements of a speaker.

Consider two data matrices $\mathbf{X} \in \mathbb{R}^{d_x \times n}$ and $\mathbf{Y} \in \mathbb{R}^{d_y \times n}$. The data samples are assumed to be dependent on Z, which is the desired output. Furthermore, $\mathbf{X}$ and $\mathbf{Y}$ are independent conditioned on $\mathbf{Z}$, as shown by the graphical model in Figure 2.9. The question is how to best combine the information in the observed data. A naive approach is to concatenate the features to obtain $\mathrm{T} \in \mathbb{R}^{(d_x+d_y)\times n}$. In the supervised case, the relevance of the different views can be evaluated directly by the classification or regression accuracy: for example, the classifier can simply discard any features irrelevant to the task. However, for semi-supervised and unsupervised learning, it becomes essential to take the conditional independence into account.

Co-training was developed by Blum and Mitchell (1998) for semi-supervised learning on multiple views. The idea is to use the views to learn two models, h_x on $\mathbf{X}$ and h_y on $\mathbf{Y}$. First, both are trained on a small amount of labeled data. Then a random part of unlabeled data is sampled and the models are used for labeling it. The results from h_x are used as new training for h_y, and vice versa. Thus any learned information from one view can help learning from the second view. A similar approach was earlier proposed by Yarowsky (1995) for word sense disambiguation: the different views for a word were its local context and document it occurred in. Theoretical justification for the method has been presented by Dasgupta et al. (2001).

In unsupervised multi-view learning, there is no teaching signal that indicates what is relevant in $\mathbf{X}$ and $\mathbf{Y}$. Instead, the assumption is that what is shared by $\mathbf{X}$ and $\mathbf{Y}$ is relevant, and what is not shared is irrelevant. This leads to maximizing the statistical dependence between the views (for an overview, see Tripathi, 2011).

The information theoretic measure for the statistical dependence between the data sets is mutual information (Equation 2.32). However, it is possible to calcu-

late only for very simple distributions.

Pearson's product-moment correlation coefficient (Pearson, 1896) is a measure of linear dependence for two scalar variables:

$$\rho_{XY} = \frac{\mathrm{cov}(X, Y)}{\sigma_x \sigma_y} \tag{2.42}$$

where $\mathrm{cov}(X, Y) = E[(X - \mu_x)(Y - \mu_y)]$ is the covariance of the variables and σ_x and σ_y are their standard deviations. The value of ρ is in $[-1, +1]$. It is zero for uncorrelated variables and high absolute values of ρ indicate strong association. While correlation is always zero for independent variables, $\rho = 0$ indicates independence only for normally distributed variables. Moreover, for Gaussian variables, correlation is is related to mutual information by $I(X; Y) = -\log(1 - {\rho_{xy}}^2)/2$. If the variables are not Gaussian, correlation measures only the linear part of the dependence.

Another common measure for bivariate dependence is *Spearman's rank correlation coefficient* (Spearman, 1904). It is a non-parametric measure, which assesses how well the relationship between two variables can be described using the best-fitting monotonic function. The actual values of X and Y are replaced by their ranks, and a formula similar to Equation 2.42 is used to obtain the coefficient for the ranks. Using ranks instead of the actual values makes the measure robust for outliers. However, this kind of monotonic dependence is harder to interpret than linear correlation.

In multi-view learning, the variables X and Y have generally high dimensionality, and there is no known matching between their dimensions. *Canonical correlation analysis* (CCA; see Section 2.8.3) can be used to find the linear projections that maximize Pearson's correlations between the projected variables. For measures of non-linear dependence, CCA can be extended with non-linear kernel functions (see, e.g., Gretton et al., 2005).

2.5.5 Multi-task and transfer learning

In multi-task learning (Caruana, 1997) and transfer learning (Thrun, 1995), there are two or more related tasks, and the question is how to use the information from learning one task in the other tasks. Multi-task learning concerns using a shared representation for a set of subtasks, while transfer learning concerns how to use knowledge learned from one problem to another problem.

As an example of transfer learning from the area of NLP, Sutton and McCallum (2005) consider transfer learning with CRFs, undirected graphical models. Their main task is annotating a set of e-mails that announce seminars at a university. As it involves finding location and person names, named-entity tagging is a related task. A basic approach is to cascade the two tasks so that the tagged named-entities are used as features for the main task. Training a joint model for both tasks is usually expensive in computation time and requires joint training data. Instead, they train cascaded CRF models separately, but combine them for making new predictions.

2.5.6 Reinforcement learning

Reinforcement learning is concerned with a setting that differs from those mentioned above. It requires an active learner, called *agent*, that performs *actions* in some real or simulated environment. The outputs of the learner are thus ac-

tions. Input data for the agent are the state of the environment and a reward. Instead of having direct answers for which single output is desirable and which is undesirable (as in supervised learning), the learner has to first perform several actions, and only later it is either rewarded or penalized for them. In the extreme case, the reward is given only when the agent reaches its goal. Thus the agent has to learn a sequence of actions, a *policy*, that enables it to reach the goal state, by trial and error. In the more complex and realistic settings, the state of the environment is only partially observable. There may also be multiple agents that should learn to interact and co-operate or compete. The applications of reinforcement learning to NLP have been limited, but include dialogue systems (Singh et al., 1999) and mapping instructions to actions (Branavana et al., 2009).

2.6 Parametric machine learning

This section gives an overview of the components required by a parametric machine learning method: a model, a cost function, and a learning algorithm. Particular emphasis is given to the questions of parameter estimation and model selection.[5]

The selection of a point hypothesis in parametric methods can be formalized as choosing a finite-dimensional parameter $\boldsymbol{\theta} \in \Theta$, where Θ is the *parameter space* of the model $\mathcal{M}$. The model defines the collection of all possible outcomes of the learning, and $\boldsymbol{\theta}$ defines which of the possible outcomes is in use. The advantage of the methods is that the problem of learning reduces to the estimation of a usually small number of model parameters (for example, the mean and variance of a Gaussian distribution). However, if the selected model does not fit the data, the bias may result in large errors.

2.6.1 Cost function

The point hypothesis is generally selected by minimizing a *cost function* (also called loss or objective function) $L(\boldsymbol{\theta}, \boldsymbol{D}, \mathcal{M})$ that depends on the parameters, model, and the data set. In a simple case of predicting one observed variable y based on another observed variable x, the cost may simply be the number of errors the predictions give for the data set.

Selecting the parameters $\boldsymbol{\theta} \in \Theta$ for a fixed model $\mathcal{M}$ is called parameter estimation. When there are several choices of parameters that minimize the cost function for the given data, the optimization problem is *ill-posed* (Alpaydin, 2004, p. 32). For example, fitting a third degree polynomial to three data points has an infinite number of solutions. In this typical example, the model is too complex for the data. In order to get a unique solution, additional bias has to be entered either into the model (e.g., accept only second degree polynomials) or the cost function (e.g., give smaller cost to polynomials in which the high-degree coefficients are zeros).

If the hypothesis class is not restricted to a single model, the cost function will depend on both the model and its particular choice of parameters. For example, one might have $\mathcal{H} = \mathcal{M}^{(1)} \cup \mathcal{M}^{(2)} \cup \ldots$, where $\mathcal{M}^{(k)}$ denotes the k^{th} degree polynomials. A set of models that differ only in a limited manner (e.g., the

[5] Textbook introductions on parametric learning are found, for example, from Alpaydin (2004, Ch. 4) and Bishop (2006, Ch. 1).

degree of the polynomial) is usually referred to as *model family*. The term *model selection* refers to the selection of one model $\mathcal{M} \subset \mathcal{H}$ from a model family or a collection of model families in $\mathcal{H}$.

While parameter estimation and model selection are often considered separately, they are related in that both can be dealt within the choice of the cost function. The difference between the two is that in parameter estimation, there is one model with a fixed number of parameters, and in model selection, there is a model family or collection among which the number of parameters differ. However, the division is less clear if one strives, for example, for a sparse representation, in which most of the parameters should be close to zero.

2.6.2 Maximum-likelihood estimate

For a model that defines the density function for the training data, the simplest choice of cost function is the inverse of the data likelihood:

$$L(\boldsymbol{\theta}, \boldsymbol{D}) \propto \frac{1}{p(\boldsymbol{D} \mid \boldsymbol{\theta})} \tag{2.43}$$

That is, the goal is to select the model that maximizes the likelihood; hence the result is called *maximum-likelihood estimate* (MLE). To simplify the notation in the case of parameter estimation only, the model $\mathcal{M}$ is excluded from the formulae. Moreover, the best choice of parameters for a cost function L is denoted briefly $\hat{\boldsymbol{\theta}}_L(\boldsymbol{D}) = \arg\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta}, \boldsymbol{D})$. The MLE can be written also by using log-likelihood:

$$\hat{\boldsymbol{\theta}}_{\mathrm{ML}}(\boldsymbol{D}) = \arg\min_{\boldsymbol{\theta}} \big\{ -\log p(\boldsymbol{D} \mid \boldsymbol{\theta}) \big\}, \tag{2.44}$$

as the logarithm is a monotonic function. In regression tasks, assuming Gaussian distributed error and maximizing the likelihood of the model corresponds to minimizing the sum of squared errors.

2.6.3 Overfitting and underfitting

Usually the model should be able to predict new data, not just describe or visualize the observed data set. Then the learner has to *generalize* the observed data set to be able to predict the output for such input that was not present in the set. For evaluation purposes, available data is divided into two independent parts: *training data set* $\boldsymbol{D}_{\mathrm{train}}$ and *test data set* $\boldsymbol{D}_{\mathrm{test}}$. First, the training set is used to learn the model parameters by minimizing the value of the cost function:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta}, \mathcal{M}, \boldsymbol{D}_{\mathrm{train}}). \tag{2.45}$$

Then the test set is used to calculate the cost with the learned parameters: $L_{\mathrm{test}} = L(\hat{\boldsymbol{\theta}}, \mathcal{M}, \boldsymbol{D}_{\mathrm{test}})$. A much higher cost on the test data than on the training data is an indication of *overfitting*: the model was able to describe the training data accurately, but not to generalize to new data samples. Again, this is often due to permitting models that are more complex than the function underlying the data. The opposite, too simple a model, results in *underfitting* and shows in high cost for both the training and the test data.

The cost functions derived from MLE are concerned only with how accurately the training samples can be modeled. Combined with complex models and a limited number of training samples, the model is easily overfitted, as the best

option is to select such a complex model that it fits the training data perfectly. This is especially problematic if the training data has noise that should not be modeled.

2.6.4 Cross-validation

Cross-validation is a method for finding the model of optimal complexity while using a cost function based on maximum likelihood. The original training data set is divided into two distinct sets: a new training set $\boldsymbol{D}_{\text{train}}$ and a *validation set* $\boldsymbol{D}_{\text{valid}}$. The former is used for training the model, and the latter for calculating the cost for unseen data. Increasing the model complexity decreases the cost for training data, but as soon as the increased complexity results in overfitting, the cost for the validation set stops decreasing and may even start to increase. This point corresponds to the optimal level of complexity.

The drawback of cross-validation is that it requires a large number of data samples. This can be alleviated by k-fold cross-validation: the training data is divided into k subsets, of which one at a time is used as a validation set and the others as training set. Thus all the data samples are used both for training and validation. Obviously, k-fold cross-validation requires that the model is trained k times for each set of parameters, making it computationally burdensome.

2.6.5 Regularization

Regularization is a common heuristic for parameter estimation or model selection. The cost function is augmented to give a penalty for complex models. Given a function $C(\cdot)$ that evaluates how complex the model is for the current model parameters, the augmented cost function is

$$\hat{L}(\boldsymbol{\theta}, \boldsymbol{D}) = L(\boldsymbol{\theta}, \boldsymbol{D}) + \lambda C(\boldsymbol{\theta}), \tag{2.46}$$

where λ gives a weight for the penalty. The weight can be optimized using cross-validation. For numerical parameters, typical choices of $C(\cdot)$ are the number of non-zero parameters (ℓ_0 regularization), the sum of the absolute values (ℓ_1 regularization) or squared values of the parameters (ℓ_2 regularization). The ℓ_0 and ℓ_1 regularizations strive for sparse parameters, but the cost function is easier to optimize with ℓ_2 regularization: for ℓ_1, the function is not differentiable, and for ℓ_0, the function is not even continuous.

In *structural risk minimization* (Vapnik, 1999), the complexity of a binary classifier is measured by the Vapnik–Chervonenkis (VC) dimension of the model. In brief, VC dimension is the largest number of data points for which the classifier learns *any* binary labeling without errors. For example, the VC dimension of a linear classifier $y = \text{sign}(\mathbf{w}^{\mathrm{T}}\mathbf{x} + b)$ is three.

2.6.6 Bayesian parameter estimation

Bayesian inference provides a theoretically rigorous way to derive the cost function given that the model is correct and the learner has a prior belief in the model parameters. From Bayes' theorem, we can say that the probability of parameters $\boldsymbol{\theta}$ given the data $\boldsymbol{D}$, called the *posterior probability*, is

$$p(\boldsymbol{\theta} \,|\, \boldsymbol{D}) = \frac{p(\boldsymbol{\theta}) \times p(\boldsymbol{D} \,|\, \boldsymbol{\theta})}{p(\boldsymbol{D})}. \tag{2.47}$$

That is, the posterior probability of $\boldsymbol{\theta}$ is the product of the prior $p(\boldsymbol{\theta})$ and data likelihood $p(\boldsymbol{D} \mid \boldsymbol{\theta})$ up to a normalization term that does not depend on $\boldsymbol{\theta}$.

To proceed, one option is to use a cost function that maximizes the posterior probability; for example

$$L(\boldsymbol{\theta}, \boldsymbol{D}) \propto \frac{1}{p(\boldsymbol{\theta} \mid \boldsymbol{D})} \tag{2.48}$$

This gives the *maximum a posteriori* (MAP) estimate for the parameters:

$$\hat{\boldsymbol{\theta}}_{\text{MAP}} = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta} \mid \boldsymbol{D}) = \arg\max_{\boldsymbol{\theta}} \left\{ p(\boldsymbol{\theta}) \times p(\boldsymbol{D} \mid \boldsymbol{\theta}) \right\} \tag{2.49}$$

Another possibility, called the *Bayes estimator*, is to take an average over all $\boldsymbol{\theta}$ weighted by their posterior probabilities:

$$\hat{\boldsymbol{\theta}}_{\text{Bayes}} = \arg\max_{\boldsymbol{\theta}} \int p(\boldsymbol{\theta} \mid \boldsymbol{D}) \boldsymbol{\theta} d\boldsymbol{\theta}. \tag{2.50}$$

However, instead of selecting a point estimate of the parameters, the Bayesian approach supports using directly the posterior distribution, if the form of its density function is tractable. The probability of a sample x can then be calculated by marginalizing over the parameters:

$$p(x \mid \boldsymbol{D}) = \int p(x \mid \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \boldsymbol{D}) d\boldsymbol{\theta}. \tag{2.51}$$

This is sometimes referred to as the *full Bayesian* approach. A practical problem with this approach is that the integral often cannot be derived in analytical form, and approximation methods are required (see, e.g., Bishop, 2006, Ch. 10).

The Bayesian approaches are dependent on the selection of the prior probability, which may sometimes be hard to come up with. *Non-informative priors* are intended to have as little influence on the posterior distribution as possible. For example, if a discrete parameter has K possible values, the prior of each value can be simply set to $1/K$. Such priors have little effect on the cost function when the number of parameters is fixed.

2.6.7 Bayesian model selection

The Bayesian approach can also be applied to model selection. Consider a set of models $\mathcal{M}^{(1)}, \ldots, \mathcal{M}^{(n)}$. The Bayesian *marginal likelihood* for $\mathcal{M}^{(k)}$ is

$$p(\boldsymbol{D} \mid \mathcal{M}^{(k)}) = \int p(\boldsymbol{\theta}^{(k)} \mid \mathcal{M}^{(k)}) p(\boldsymbol{D} \mid \boldsymbol{\theta}^{(k)}, \mathcal{M}^{(k)}) d\boldsymbol{\theta}^{(k)}. \tag{2.52}$$

Bayesian model selection favors the model with the largest $p(\boldsymbol{D} \mid \mathcal{M}^{(k)})$. When comparing two models, the so-called *Bayes factor*

$$K = \frac{p(\boldsymbol{D} \mid \mathcal{M}^{(1)})}{p(\boldsymbol{D} \mid \mathcal{M}^{(2)})} \tag{2.53}$$

can be used to decide which of the models has more evidence. The Bayesian information criterion (BIC) by Schwarz (1978) approximates the integrals of the marginal likelihoods to obtain

$$\text{BIC} = -2 \ln p(\boldsymbol{D}_{\text{train}} | \boldsymbol{\theta}) + k \ln n, \tag{2.54}$$

where k is the number of free parameters to be estimated and n is the number of observed data samples. A smaller BIC score is favored. The Akaike information criterion (AIC) gives a similar result with an information theoretic approach (Akaike, 1974).

If model priors $p(\mathcal{M}^{(k)})$ can be defined, an alternative to selecting a single model is *model inference*:

$$p(\mathcal{M}^{(k)} \mid \boldsymbol{D}) = \frac{p(\mathcal{M}^{(k)})p(\boldsymbol{D} \mid \mathcal{M}^{(k)})}{\sum_{i=1}^{n} p(\mathcal{M}^{(i)})p(\boldsymbol{D} \mid \mathcal{M}^{(i)})}, \tag{2.55}$$

Similar to Equation 2.51, $p(x \mid \boldsymbol{D})$ can be obtained by marginalizing over the models; this is called Bayesian model averaging (Hoeting et al., 1999).

The model priors can also be applied to get a point hypothesis $h = (\boldsymbol{\theta}^{(i)}, \mathcal{M}^{(i)})$ with the MAP estimate

$$\begin{aligned} p(\boldsymbol{\theta}^{(i)}, \mathcal{M}^{(i)} \mid \boldsymbol{D}) &\propto p(\boldsymbol{\theta}^{(i)}, \mathcal{M}^{(i)})p(\boldsymbol{D} \mid \boldsymbol{\theta}^{(i)}, \mathcal{M}^{(i)}) \\ &= p(\mathcal{M}^{(i)})p(\boldsymbol{\theta}^{(i)} \mid \mathcal{M}^{(i)})p(\boldsymbol{D} \mid \boldsymbol{\theta}^{(i)}, \mathcal{M}^{(i)}). \end{aligned} \tag{2.56}$$

It is more or less arbitrary which differences between two point hypotheses are considered to originate from different models and which from different parameters: one can always do reparametrization that eliminates the difference (e.g., $\phi = (i, \theta^{(1)}, \ldots, \theta^{(n)})$, $\bar{\mathcal{M}} = \bigcup_{i=1}^{n} \mathcal{M}^{(i)}$). Still, it is often clearer to set a prior first for $\mathcal{M}^{(i)}$ and then for $\boldsymbol{\theta}^{(i)}$ given $\mathcal{M}^{(i)}$. A model prior that gives a higher probability the simpler the model is may help with overfitting, even when non-informative priors are used for the parameters. This approach is very similar to the minimum description length principle, discussed next.

2.6.8 Minimum description length principle

The minimum description length (MDL) principle by Rissanen (1978, 1989) is an information-theoretic approach to model selection. It has several variants as well as closely related approaches. The basic motivation that is common to them is the following: Learning is finding regularities in the data. As any regularity in the data can be used to compress it, *learning can be viewed as compression*. Other philosophical views endorsed by MDL developers are that models are languages (or less universally, codes) describing useful properties of the data, and that in practice there is no "true" model that could be identified from the hypothesis class. Accordingly, a learning method should have an interpretation which depends only on the data, as MDL does.

Given an arbitrary data set, the shortest imaginable code length is provided by a universal machine: the algorithmic complexity of the data. As the complexity cannot be calculated, this is called *ideal MDL* in contrast to the "practical" MDL versions discussed next.

The first version of MDL introduced by Rissanen (1978) is the *two-part coding scheme*, referred to as crude MDL by Grünwald (2005). It can be explained in the terms of source coding (Figure 2.7, page 42): the goal is to encode a set of data $\boldsymbol{D}$ from source $p(X)$ with the minimum possible number of bits so that it can still be recovered by the decoder. The encoding is based on a point hypothesis h from the selected hypothesis class $\mathcal{H}$. Each hypothesis is associated with a probability distribution $p(X \mid h)$. While the hypothesis class is assumed to be known to both encoder and decoder, the point hypothesis is not. Thus first the hypothesis is encoded, using description length of $l(h)$ bits, and then the data is encoded with

the help of the hypothesis, using description length of $l_h(\boldsymbol{D})$ bits. The sum of the two lengths is the total code length:

$$l(\boldsymbol{D}, h) = l(h) + l_h(\boldsymbol{D}). \tag{2.57}$$

The best point hypothesis h minimizes this sum, and the best model $\mathcal{M} \in \mathcal{H}$ is the smallest model containing the selected h.

To apply MDL, the description lengths for both the hypothesis and the data given the hypothesis have to be decided. The latter is evident: The optimal coding (cf. Section 2.4.2, page 41) associated with the distribution $p(\boldsymbol{D} \mid h)$ gives the minimum coding length: $l_h(\boldsymbol{D}) = -\log_2 p(\boldsymbol{D} \mid h)$. However, the two-part coding does not give guidelines for choosing a code for the hypothesis other than that (1) the decoder has to be able to define $p(\boldsymbol{D} \mid h)$ based on it, and (2) the code length should be somehow *minimal*.

For a single parametric model $\mathcal{M}$, the parameters $\boldsymbol{\theta}$ wholly encode the point hypothesis:

$$l(\boldsymbol{D}, \boldsymbol{\theta}) = l(\boldsymbol{\theta}) - \log_2(\boldsymbol{D} \mid \boldsymbol{\theta}). \tag{2.58}$$

By setting $l(\boldsymbol{\theta}) = -\log_2 p(\boldsymbol{\theta})$, this is equivalent to the MAP estimate with the prior $p(\boldsymbol{\theta})$. However, there is a corresponding (non-defective) prior distribution only if a complete code is used in the two-part coding. That is, MAP estimation is a special case in two-part MDL.

The theory of the MDL principle is only concerned with code lengths, not the actual codes. However, in applications such as language modeling, there is often a trade-off between the model size and its accuracy also in practice, and one might want to use the real encoding of the model also for the model selection.

Example. Consider a categorical source distribution $p(\mathrm{X})$ with parameters $\pi_1, \ldots, \pi_k$, $\sum_i \pi_i = 1$. A naive coding for this model is to encode the parameters as decimal numbers with fixed precision of P bits[6], providing coding length

$$l(\boldsymbol{D}, \boldsymbol{\theta}) = k \times P - \log_2 p(\boldsymbol{D} \mid \boldsymbol{\theta}). \tag{2.59}$$

Assume each x_i has been observed c_i times in total n samples. As $k \times P$ is a constant, minimizing the total length is equivalent to MLE and $\hat{\pi}_i \approx c_i / n$. The approximation error depends on P and increases the description length of the data. The more accurately the parameters are described, the more $l(\boldsymbol{\theta})$ increases, but at the same time $-\log_2 p(\boldsymbol{D} \mid \boldsymbol{\theta})$ decreases. Thus the optimal P must make a compromise between the two parts.

A more clever two-part coding stores directly the counts c_i. Then $\pi_i = c_i / \sum_i c_i$ gives the exact MLE solution. Encoding integers $0 \leq c_i \leq n$ requires $\log_2(n+1)$ bits, as there are $n+1$ possible values to choose from. First, however, n has to be encoded, as it is not known to the decoder. Positive integers can be encoded with a complete code of $\log_2 c + \log_2 n + \log_2 \log_2 n + \ldots$ bits, where $c \approx 2.865$ is constant and the sum involves only finite non-negative terms (Rissanen, 1983). While the overall coding is valid for decoding, the code for the counts c_i is still defective:

$$\sum_{\boldsymbol{x} \in \mathcal{X}^n} 2^{-k \log_2(n+1)} = \binom{n+k-1}{n} (n+1)^{-k} < 1. \tag{2.60}$$

[6] For example, binary encoding $\hat{\pi}_i = 1 - \sum_{j=1}^{P} b_j 2^{-j}$, $b_j \in \{0, 1\}$, gives $0 < \hat{\pi}_i \leq 1$. The decoder may have to normalize the probabilities $\hat{\pi}_i$ so that they sum up to one.

The factor $\binom{n+k-1}{n}$ gives the number of possible data sets of length n.[7] This observation indicates a design for a complete code: each possible combination for the parameters is given its own $\log_2 \binom{n+k-1}{k}$ bit code. The corresponding prior can be considered non-informative: each $\boldsymbol{\theta} \in \Theta$ has an equal probability.

Figure 2.10 illustrates code lengths for a k-dimensional categorical model with different parameter encodings discussed above: fixed precision codings for $P \in \{2, 4, 8\}$ bits, integer coding, and combinatorial coding. For the graphs in the top row, the number of samples n in the training data varies from 10 to 500. The left-side graph shows how the model coding length for integer and combinatorial coding increases with n. Using more bits to encode parameters estimated on larger number of samples makes sense, as with a small number of samples, the parameters will not be accurate anyway. Combinatorial coding is clearly more efficient than the integer coding.

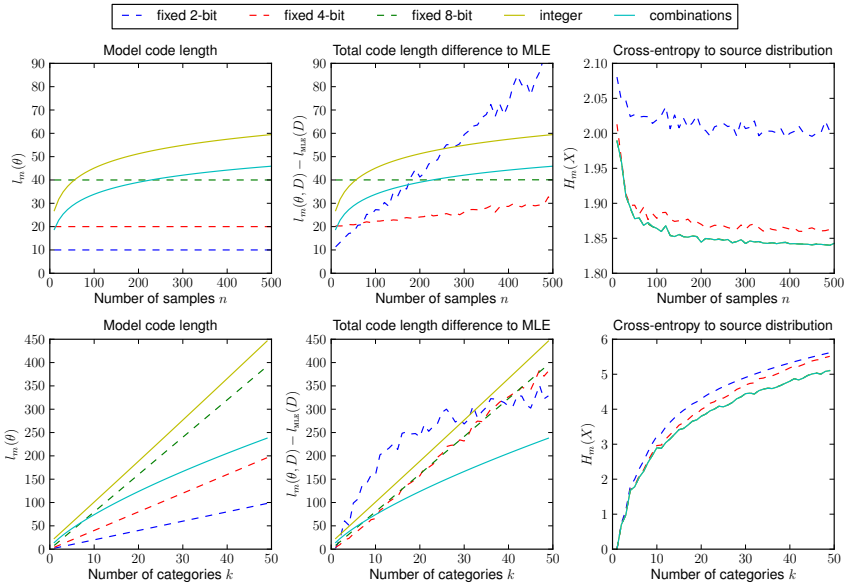


Figure 2.10. Comparison of different coding schemes for model parameters of categorical distribution. Each point is average results over 20 distributions, for which the parameters were generated from a Dirichlet distribution with $\boldsymbol{\alpha} = (1, 1, \ldots, 1)$. In the top row, the number of samples n is varied and the number of the categories $k = 5$. In the bottom row, $n = 400$ and k is varied. Left: model code length $l_m(\boldsymbol{\theta})$. Middle: difference between the total code length $l_m(\boldsymbol{\theta}, \boldsymbol{D})$ and the data cost $l_{\text{MLE}}(\boldsymbol{D})$ for the maximum-likelihood parameters. Right: cross-entropy $H_m(X)$ of the model and original source distribution. The cross-entropies of the fixed 8-bit and integer coding are not shown, as they are almost equal to those of the combinatorial coding.

The difference between total code lengths and the MLE data code length is shown in the middle of Figure 2.10. The approximation error clearly increases the total code lengths of 2 and 4 bit fixed precision encodings, but not anymore the length of the 8-bit encoding. On the right, cross-entropy of the models for the original distribution shows that 8 bits is indeed required to get as good accuracy as with exact ML estimates.

While the fixed 8-bit coding seems to be a very good option for this model if $n > 250$, the situation is different if the number of categories is increased. The bottom-left graph of Figure 2.10 shows that the model code length grows

[7] The number of ways to choose k non-negative integers that sum up to n is $\binom{n+k-1}{n}$.

linearly for the integer and fixed-precision codings, but only sub-linearly for the combinatorial coding. Thus the combinatorial coding obtains a small $l_m(\boldsymbol{\theta})$ even for large k and n while having the best possible $l_m(\boldsymbol{D} \mid \boldsymbol{\theta})$.

Theoretically optimal codes, such as the combinatorial coding above, are often impractical to implement. Thus it may sometimes be more useful to apply a defective code that follows the real implementation of the model—for example using a fixed number of bits to store each real-valued parameter as a floating point number. Then the model instance that minimizes the total code length will be optimal for the corresponding compression task.

2.6.8.1 Refined MDL

Refined MDL (Rissanen, 1996; Barron et al., 1998) differs from crude MDL in that it tries to directly select the best model instead of first finding the best point hypothesis. In the parametric case, the model parameters are not explicitly encoded. This results in *one-part* codes, which are theoretically shorter than the two-part codes.

A central concept in refined MDL is *universal coding*. Consider a set of candidate codes $\mathcal{H}$. The encoder cannot directly select the best $h \in \mathcal{H}$ based on the observed data $\boldsymbol{D}$ and send the data compressed by h, as then the decoder would not know which of the codes was used. Thus there cannot be a coding that would yield a code length equal to or smaller than $\min_{h \in \mathcal{H}} l_h(\boldsymbol{D})$ for *all* data sets $\boldsymbol{D}$. However, universal codes can do almost as well: a code u that is universal relative to the candidate codes $\mathcal{H}$ is able to compress every data sequence so that the *regret*

$$R(u) = l_u(\boldsymbol{D}) - \min_{h \in \mathcal{H}} l_h(\boldsymbol{D}) \tag{2.61}$$

grows sublinearly in the number of samples n. As the data code length typically grows linearly in n, a universal code can do in practice as well as the best choice of hypothesis for a large enough data set.

If the number of candidate codes is finite ($\mathcal{H} = \{h_1, h_2, \ldots, h_M\}$), there is a simple universal two-part code. First, encode which code to use, taking $\log_2 M$ bits, and then encode the data with selected code. The resulting code length

$$l_{\text{2-p}}(\boldsymbol{D}) = \min_{h \in \mathcal{H}} l_h(\boldsymbol{D}) + \log_2 M \tag{2.62}$$

does not depend on the number of samples, so the coding is universal. For another concrete example, consider the k-dimensional categorical distribution with combinatorial parameter encoding discussed above. The ML estimate is optimal among the model instances. Using an upper bound for the binomial coefficient,

$$\begin{aligned} l_{\text{2-p}}(\boldsymbol{D}) &= -\log_2 p(\boldsymbol{D} \mid \hat{\boldsymbol{\theta}}_{\text{ML}}) + \log_2 \binom{n+k-1}{n} + \log_2 n + O(\log_2 \log_2 n) \\ &\leq -\log_2 p(\boldsymbol{D} \mid \hat{\boldsymbol{\theta}}_{\text{ML}}) + \log_2 \frac{(n+k-1)^k}{k!} + \log_2 n + O(\log_2 \log_2 n). \end{aligned} \tag{2.63}$$

As the model code length increases sublinearly in n, the coding is universal.

The *optimal* universal code should minimize the maximal regret

$$R_{\max}(u) = \max_{\boldsymbol{D} \in \mathcal{X}^n} \left\{ l_u(\boldsymbol{D}) - \min_{h \in \mathcal{H}} l_h(\boldsymbol{D}) \right\}, \tag{2.64}$$

which measures the additional number of bits needed to encode the data compared with the best h in the worst case. For a parametric model $\mathcal{M}$, the minmax regret is

$$\min_{q} \max_{\boldsymbol{D} \in \mathcal{X}^n} \Big\{ -\log_2 q(\boldsymbol{D}) - \big[-\log_2 p(\boldsymbol{D} \mid \hat{\boldsymbol{\theta}}_{\mathrm{ML}}(\boldsymbol{D})) \big] \Big\}, \tag{2.65}$$

where $\hat{\boldsymbol{\theta}}_{\mathrm{ML}}(\boldsymbol{D})$ denotes the ML estimate of parameters for given data $\boldsymbol{D}$. The distribution q that uniquely achieves this is called the *normalized maximum-likelihood* distribution, proposed by Shtarkov (1987):

$$p_{\mathrm{nml}}(\boldsymbol{D}) = \frac{p(\boldsymbol{D} \mid \hat{\boldsymbol{\theta}}_{\mathrm{ML}}(\boldsymbol{D}))}{\sum_{\boldsymbol{D}' \in \mathcal{X}^n} p(\boldsymbol{D}' \mid \hat{\boldsymbol{\theta}}_{\mathrm{ML}}(\boldsymbol{D}'))} = \frac{p(\boldsymbol{D} \mid \hat{\boldsymbol{\theta}}_{\mathrm{ML}}(\boldsymbol{D}))}{\mathrm{COMP}_n(\mathcal{M})}. \tag{2.66}$$

The denominator is the larger the more there are different data sets that have high probabilities with the respective ML estimates. As complex models are more likely to accomplish in this than simple models, it is called *model complexity*. It has several interpretations (Grünwald, 2005). One is that it measures the number of *distinguishable distributions* in the model. If the parameter set is finite ($\Theta = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_M\}$), the complexity converges to $\log_2 M$ as $n \to \infty$, showing that two-part code with a uniform prior on the parameter space is asymptotically optimal in the sense of Equation 2.65. A similar result holds for Bayesian model selection with certain non-informative priors for the models of the exponential family.

While being theoretically motivated, refined MDL has many computational problems: $\mathrm{COMP}_n(\mathcal{M})$ may be infinite, in which case the p_{nml} is undefined. Even if finite it may be impossible to calculate, as the number of possible data sets often grows exponentially with n. For example, if $n = 100$ and $\mathcal{X}$ has $k = 10$ distinct items, the number of possible data sets is 10^{100}. Even if restricted to the first-order Markov models, there are $\binom{100+10^2-1}{100} \approx 10^{58}$ data sets that give different ML estimates. Different types of extensions are required to deal with infinite model complexity as well as non-parametric inference. In the present work, only the two-part codes is applied.

2.6.8.2 Minimum message length principle

An approach that has many similarities to MDL is the *minimum message length* (MML) principle (Wallace and Boulton, 1968; Wallace and Freeman, 1987). Both MML and MDL are inspired by the notion of algorithmic complexity, study learning in the terms of encoding, and set the goal to minimizing the total code length of the data and model. The main differences are the following (Wallace and Dowe, 1999; Grünwald, 2005): First, the goal of MML is to select the best point hypothesis, whereas MDL is designed for model selection. Consequently, the one-part codings studied in refined MDL are not relevant for MML. Second, MML is fully Bayesian and assumes subjective priors, while MDL uses non-informative priors and codes that do not always give proper probabilities.

2.6.9 Learning algorithms

If there is no analytical solution for calculating the parameters that minimize the selected cost function, a *learning algorithm*, or *optimization procedure*, is needed (Alpaydin, 2004, p. 36). The complexity of the algorithm, in both time and

space, should be low enough for the processing power and memory in use. The complexity is often analyzed as a function on the properties of the data, such as the number and dimensionality of the data samples, as well as the number of parameters in the model.

The goal of the learning algorithm is to find the global optimum of the cost function, $\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta}, \mathcal{M}, \boldsymbol{D}_{\text{train}})$. However, global optimization is often impossible in practice. The only general way, *brute force search*, requires that all possible choices of parameters are tested, and the number of choices tends to be huge, if not infinite, as in any continuous parameter space. Often one has to settle for finding a local optimum, with no guarantee of how close it is to the global optimum. *Greedy* algorithms search for a local optimum by making locally optimal changes to the parameters at each step. A simple way to improve the situation with many local optima is to run the search algorithm several times with different starting conditions, and select the best overall result. Another common heuristic is *simulated annealing* (Kirkpatrick et al., 1983), in which non-optimal choices are made during the search with a probability that decreases in time.

2.7 Non-parametric and semi-parametric models

Non-parametric machine learning differs from the general parametric setup described above. The term non-parametric is sometimes misleading, as the models may still have parameters. However, the parameters are more flexible in some manner: they either coincide directly with the training data samples, grow with the number of samples, or the parameter vector may be very high or infinite dimensional, so that it is not possible to estimate and store it directly. The term semi-parametric is used for models that have both parametric and non-parametric components.

2.7.1 Memory-based learning

Memory-based or *instance-based* learning algorithms are strictly non-parametric: instead of selecting parameters based on the training samples, they directly store the samples. The prediction on a new sample is then based on those stored samples that are most similar to it. The basic assumption required by this approach is that the similar inputs have similar outputs. The advantage is that the model is trivial to adapt when new samples are observed, without the need to re-estimate any parameters. The model complexity grows automatically with the amount of data. A drawback is that the worst case memory and time complexities are at least $O(n)$ for n samples.

One of the oldest and most popular non-parametric methods is the *histogram* estimator. A histogram estimates probability density function $p(X)$ by dividing the input space into equal-sized intervals called *bins*, and gives each point in the input space a probability proportional to the number of training samples in the corresponding bin. *Kernel density estimators* are extensions of histograms that use smooth regions of influence for the training samples.

Another simple, memory-based algorithm is *k-nearest neighbors* (KNN) classification. For a given sample x, the k most similar stored samples are searched. The prediction y is based on majority vote of the classes of the nearest samples.

Memory-based learning has been studied for several supervised tasks in lan-

guage processing (Daelemans and van den Bosch, 2005). Daelemans et al. (1999) argue that keeping exceptional training instances in memory is beneficial for generalization accuracy. One practical application for the approach is example-based machine translation (van den Bosch et al., 2007).

2.7.2 Kernel methods

Kernel methods are a class of non-parametric extensions for various types of learning methods. One of the first and most popular algorithms using kernels is the *support vector machine* (SVM) by Cortes and Vapnik (1995). SVM is used in classification and regression problems, but kernel methods can also be applied, for example, in dimensionality reduction and multi-view learning.

Consider a function $\phi(\cdot)$ that projects the data samples into a feature space $\mathcal{Z}$ of high or infinite dimensionality. If $\phi(\cdot)$ is non-linear, even a very simple algorithm applied in $\mathcal{Z}$ can solve complicated, non-linear problems. Kernel methods provide a way to do that in practice.

If the cost function used by the algorithm can be written in a form that includes only scalar products $\mathbf{x}_i^{\mathrm{T}}\mathbf{x}_j$, often referred to as the *dual form*, it is possible to apply a *kernel function*. The kernel function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ defines the product of two vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ in $\mathcal{Z}$:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^{\mathrm{T}}\phi(\mathbf{x}_j) \tag{2.67}$$

Valid kernel functions, for which this decomposition exists, are finitely positive semi-definite, and the corresponding space $\mathcal{Z}$ is called *reproducing kernel Hilbert space* (Shawe-Taylor and Cristianini, 2004, Ch. 3). Typical kernels for real-valued data include polynomial kernels and Gaussian kernels. Kernels can also be devised for discrete data such as sets or strings. For example, Lodhi et al. (2002) use string kernels for text classification.

Replacing the scalar product $\mathbf{x}_i^{\mathrm{T}}\mathbf{x}_j$ with a kernel function is called *kernel substitution* or *kernel trick*. The kernel function has to be calculated over all pairs of training samples. For n samples, this gives an $n \times n$ matrix $\mathbf{K}$, where $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$. As n is usually larger than the original dimension of the data, the kernel trick comes at the cost of increased computational complexity.

2.7.3 Mixture models

Mixture models can be seen as semi-parametric density estimators (Alpaydin, 2004, Ch. 7). Instead of using a single parametric distribution or a non-parametric histogram based on the observed samples, a set of parametric distributions are fitted into the space according to the location of the samples.

Mathematically, the samples $\mathbf{x}$ are assumed to come from a mixture of K independent *mixture components* c_j. Each component is associated with a density $p(\mathbf{x} \mid c_j, \boldsymbol{\theta})$ of a parametric distribution. The mixture density is

$$p(\mathbf{x} \mid \boldsymbol{\theta}) = \sum_{j=1}^{K} p(c_j \mid \boldsymbol{\theta}) p(\mathbf{x} \mid c_j, \boldsymbol{\theta}), \tag{2.68}$$

where $\boldsymbol{\theta}$ includes the prior probabilities $p(c_j \mid \boldsymbol{\theta})$ and parameters of the component distributions. The number of parameters in $\boldsymbol{\theta}$ is finite and independent of the data samples. However, in order to estimate the parameters of the model, there has to be a way to decide which data samples are generated from which

mixture components. The assignment can be considered as a hidden variable Y, where $y \in \{1, \ldots, K\}$ assigns sample $\mathbf{x}$ to mixture component c_y. The dimensionality of the parameter vector $\mathbf{y}$ grows with the number of the samples, which justifies the categorization to semi-parametric methods. The EM algorithm used for learning the parameters is described in Section 2.8.5.

2.7.4 Non-parametric Bayesian methods

Non-parametric Bayesian methods apply models that have potentially infinitely many parameters (Müller and Quintana, 2004). As in frequentist non-parametric methods such as histograms, they try to avoid parametric assumptions. For example, *infinite mixture models* (Rasmussen, 2000) estimate an arbitrary number of mixture components, and *infinite hidden Markov models* (Beal et al., 2002) estimate an arbitrary number of states.

The non-parametric Bayesian models are usually based on stochastic processes such as the *Dirichlet process* (Ferguson, 1973) and its generalization, the *Pitman-Yor process* (Pitman and Yor, 1997), whose domains are themselves random distributions. That is, the input of a random process P is a base distribution G_0 and optionally some parameters $\boldsymbol{\theta}$, and its output is a new distribution G:

$$G \mid G_0, \boldsymbol{\theta} \sim \mathrm{P}(G_0, \boldsymbol{\theta}) \tag{2.69}$$

Both the Dirichlet process (DP) and the Pitman-Yor process (PYP) implement a "rich gets richer" phenomenon. Thus they are suitable for modeling power-law distributed data such as language (Goldwater et al., 2006; Teh, 2006; Huang and Renals, 2010a). In consequence, some details are provided here.

Goldwater et al. (2006, 2011) describe language models based on DP or PYP as *two-stage* models. At the first stage, a sequence $\boldsymbol{l} = (l_1, \ldots, l_K)$ of K discrete lexical items is generated by a *generator* G. Each l_i is generated independently, which means that there may be repetitions. At the second stage, *adaptor* P generates a sequence of n integers $\boldsymbol{z}$, where $1 \leq z_i \leq K$. The output of the two-stage process TwoStage(P, G) is then the sequence $\boldsymbol{w} = (w_1, \ldots, w_n)$, where $w_i = l_{z_i}$.

The adaptor for DP is a *Chinese restaurant process* (CRP) and the adaptor for PYP is *Pitman-Yor Chinese restaurant process* (PYCRP) (Goldwater et al., 2011). The names originate from the analogy of a restaurant that has an infinite number of tables with infinite number of seats. The current seating arrangement for i customers is $\boldsymbol{z}_1^i = (z_1, \ldots, z_i)$. When the $(i+1)^{\text{th}}$ customer enters the restaurant, she selects the k^{th} table with a probability that depends on how many people are already sitting at the table, denoted $c_k(\boldsymbol{z}_1^i)$. In PYCRP,

$$p(z_{i+1} = k \mid \boldsymbol{z}_1^i, d, \theta) = \begin{cases} \dfrac{c_k(\boldsymbol{z}_1^i) - d}{\theta + i} & \text{if } 1 \leq k \leq t(\boldsymbol{z}_1^i) \\ \dfrac{\theta + dt(\boldsymbol{z}_1^i)}{\theta + i} & \text{if } k = t(\boldsymbol{z}_1^i) + 1 \end{cases}, \tag{2.70}$$

where $t(\boldsymbol{z}_1^i)$ is the total number of occupied tables, and $0 \leq d < 1$ and $\theta > 0$ are parameters of the process. When $d = 0$, PYCRP is equivalent to the CRP process, in which the probability of an occupied table is proportional to the number of customers seated and the probability of a new table is proportional to θ.

In TwoStage(PYCRP(d, θ), G) or PYP(G, d, θ), the k^{th} table is labeled according to the lexical item l_k generated by G. The probability that the outcome w_{i+1} is w

is given by

$$p(w_{i+1} = w \mid \boldsymbol{z}_1^i, \boldsymbol{l}(\boldsymbol{z}_1^i), d, \theta, G) = \sum_{k=1}^{t(\boldsymbol{z}_1^i)} \mathrm{I}(l_k = w) \frac{c_k(\boldsymbol{z}_1^i) - d}{\theta + i} + \frac{dt(\boldsymbol{z}_1^i) + \theta}{\theta + i} G(w)$$
$$= \frac{c_w(\boldsymbol{w}_1^i) - dt_w(\boldsymbol{z}_1^i)}{\theta + i} + \frac{\theta + dt(\boldsymbol{z}_1^i)}{\theta + i} G(w), \quad (2.71)$$

where $\boldsymbol{l}(\boldsymbol{z}_1^i)$ are the labels for the tables in $\boldsymbol{z}_1^i$, $t_w(\boldsymbol{z}_1^i)$ is the number of tables labeled with w, and $c_w(\boldsymbol{w}_1^i)$ is the number of occurrences of word w in $\boldsymbol{w}_1^i$. The new outcome w may then be generated either by any of the current tables labeled by w (the first term of Equation 2.71) or from the base distribution G (the second term). The two-stage language models are further discussed in Section 4.2.3 and Section 6.4.3.

2.8 Common unsupervised learning methods

This last section on machine learning describes a few common unsupervised learning methods and algorithms that are relevant for the rest of the thesis.

2.8.1 Matrix decompositions

Let $\mathbf{X}$ be a square $n \times n$ matrix. Eigenvectors of the matrix are non-zero vectors that either remain proportional to the original vector when multiplied by the matrix, or become zero. The corresponding eigenvalues are the factors by which the eigenvectors change when multiplied by the matrix. That is, for eigenvector $\mathbf{v}$ and eigenvalue λ,

$$\mathbf{X}\mathbf{v} = \lambda\mathbf{v}. \quad (2.72)$$

If $\mathbf{X}$ has n linearly independent eigenvectors (i.e., there is no eigenvector $\mathbf{v}_i$ such that $\mathbf{v}_i = \sum_{j \neq i} a_j \mathbf{v}_j$), $\mathbf{X}$ can be factorized as

$$\mathbf{X} = \mathbf{V}\mathbf{D}\mathbf{V}^{-1}, \quad (2.73)$$

where $\mathbf{V}$ is square matrix of the eigenvectors $\mathbf{v}_i$ and $\mathbf{D}$ is the diagonal matrix of the eigenvalues. This is called *eigenvalue decomposition*. If none of the eigenvalues are zero, $\mathbf{X}$ is non-singular and has the inverse $\mathbf{X}^{-1} = \mathbf{V}^{-1}\mathbf{D}^{-1}\mathbf{V}$.

Singular value decomposition (SVD) of matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$ is

$$\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^{\mathrm{T}}, \quad (2.74)$$

where the orthogonal matrices $\mathbf{U} \in \mathbb{R}^{m \times m}$ and $\mathbf{V} \in \mathbb{R}^{n \times n}$ contain the *left* and *right singular vectors* of $\mathbf{X}$ and the diagonal $m \times n$ matrix $\mathbf{D}$ contains the respective *singular values*. SVD is related to eigenvalue decompositions of $\mathbf{X}\mathbf{X}^{\mathrm{T}}$ and $\mathbf{X}^{\mathrm{T}}\mathbf{X}$:

$$\mathbf{X}\mathbf{X}^{\mathrm{T}} = \mathbf{U}\mathbf{D}\mathbf{V}^{\mathrm{T}}\mathbf{V}\mathbf{D}\mathbf{U}^{\mathrm{T}} = \mathbf{U}\mathbf{D}^2\mathbf{U}^{\mathrm{T}}; \quad (2.75)$$
$$\mathbf{X}^{\mathrm{T}}\mathbf{X} = \mathbf{V}\mathbf{D}\mathbf{U}^{\mathrm{T}}\mathbf{U}\mathbf{D}\mathbf{V}^{\mathrm{T}} = \mathbf{V}\mathbf{D}^2\mathbf{V}^{\mathrm{T}}. \quad (2.76)$$

That is, $\mathbf{U}$ and $\mathbf{V}$ are the eigenvectors of $\mathbf{X}\mathbf{X}^{\mathrm{T}}$ and $\mathbf{X}^{\mathrm{T}}\mathbf{X}$, respectively, and the diagonal of $\mathbf{D}^2$ has the corresponding eigenvalues.

Truncated SVD, where only the k highest singular values and the corresponding singular vectors are selected to obtain $\widehat{\mathbf{X}} = \mathbf{U}_k\mathbf{D}_k{\mathbf{V}_k}^{\mathrm{T}}$ gives the best rank k solution for

$$\min_{\widehat{\mathbf{X}}} \|\mathbf{X} - \widehat{\mathbf{X}}\|_F, \quad (2.77)$$

where $\|\mathbf{A}\|_F = \sqrt{\sum_i \sum_j a_{ij}^2}$ is the Frobenius norm. A k-dimensional representation for the samples can be obtained by projecting $\widehat{\mathbf{X}}$ into the space spanned by the left singular vectors corresponding to the k largest singular values:

$$\mathbf{Y} = \mathbf{U}_k{}^{\mathrm{T}}\widehat{\mathbf{X}} \tag{2.78}$$

The same projection can be used for any new samples $\mathbf{x} \in \mathbb{R}^m$.

2.8.2 Principal component analysis

Principal component analysis, first proposed by Pearson (1901), is the projection of a d-dimensional variable $\mathbf{x}$ into the space spanned by the orthogonal components of the largest variance. The first principal component is

$$\mathbf{w}_1 = \arg\max_{\|\mathbf{w}\|=1} \mathrm{Var}[\mathbf{w}^{\mathrm{T}}\mathbf{x}] = \arg\max_{\|\mathbf{w}\|=1} E[(\mathbf{w}^{\mathrm{T}}\mathbf{x})^2] - (\mathbf{w}^{\mathrm{T}}E[\mathbf{x}])^2. \tag{2.79}$$

The subsequent components are maximized similarly but with the restriction that they are uncorrelated with the previous components.

Let $\bar{\mathbf{X}}$ be a centered data matrix, where the sample mean vector $\boldsymbol{\mu}$ of the data is subtracted from each column of $\mathbf{X}$. PCA can be calculated by eigenvalue decomposition of the sample covariance matrix

$$\mathbf{C}_x = \frac{1}{N-1}\bar{\mathbf{X}}\bar{\mathbf{X}}^{\mathrm{T}}, \tag{2.80}$$

or equivalently, singular value decomposition of $\bar{\mathbf{X}}$ (cf. Equation 2.75). The first k eigenvectors or left singular vectors give the k first principal components $\mathbf{W}$. The projected data is obtained by

$$\mathbf{Y} = \mathbf{W}^{\mathrm{T}}\bar{\mathbf{X}}. \tag{2.81}$$

The projection is called the *Karhunen-Loève transform*. If $\mathbf{X}$ has a zero mean, it coincides with the truncated SVD projection in Equation 2.78. The new features are orthogonal. The *whitening* transformation $\mathbf{Z} = \mathbf{D}^{-1}\mathbf{Y}$, where $\mathbf{D}$ is the matrix of singular values, converts the covariance matrix further into identity matrix $\mathbf{I}$.

2.8.3 Canonical correlation analysis

Canonical correlation analysis, originally proposed by Hotelling (1936), is a linear method for finding relationships between two sets of variables. It finds linear projections for each set of variables so that the correlation between the projections is maximized (Borga, 1998; Bach and Jordan, 2003; Hardoon et al., 2004).

Consider two column vectors of random variables $\mathbf{x} = (x_1, \ldots, x_{D_x})^{\mathrm{T}}$ and $\mathbf{y} = (y_1, \ldots, y_{D_y})^{\mathrm{T}}$ with zero means. For each variable pair, we want to find linear transformations into scalars, $u_1 = \mathbf{a}^{\mathrm{T}}\mathbf{x}$ and $v_1 = \mathbf{b}^{\mathrm{T}}\mathbf{y}$, so that the correlation between the scalars is maximized:

$$\rho_1 = \max_{\mathbf{a},\mathbf{b}} \mathrm{corr}(u_1, v_1) = \max_{\mathbf{a},\mathbf{b}} \frac{E[\mathbf{a}^{\mathrm{T}}\mathbf{x}\mathbf{y}^{\mathrm{T}}\mathbf{b}]}{\sqrt{E[\mathbf{a}^{\mathrm{T}}\mathbf{x}\mathbf{x}^{\mathrm{T}}\mathbf{a}]E[\mathbf{b}^{\mathrm{T}}\mathbf{y}\mathbf{y}^{\mathrm{T}}\mathbf{b}]}}. \tag{2.82}$$

Correlation ρ_1 is the first canonical correlation and u_1 and v_1 are the first canonical variates. The subsequent canonical variates u_i and v_i are set to be maximally correlated as in Equation 2.82 with the restriction that they are uncorrelated with

all the previous variates. That is, $E[u_i u_j] = E[v_i v_j] = E[u_i v_j] = 0$ for all $i \neq j$. In total, there can be $D = \min(D_x, D_y)$ canonical variates and correlations.

There is a simple relationship between canonical correlation and mutual information for Gaussian random variables (Bach and Jordan, 2003). If $\mathbf{x}$ and $\mathbf{y}$ are Gaussian, the mutual information $I(\mathbf{x}; \mathbf{y})$ can be written as

$$I(\mathbf{x}; \mathbf{y}) = -\frac{1}{2} \ln \Big(\frac{|\mathbf{C}|}{|\mathbf{C}_{xx}||\mathbf{C}_{yy}|} \Big), \tag{2.83}$$

where $|\cdot|$ denotes the determinant of a matrix and

$$\mathbf{C} = \begin{pmatrix} \mathbf{C}_{xx} & \mathbf{C}_{xy} \\ \mathbf{C}_{yx} & \mathbf{C}_{yy} \end{pmatrix} \approx \frac{1}{N-1} \begin{pmatrix} \mathbf{X} \\ \mathbf{Y} \end{pmatrix} \begin{pmatrix} \mathbf{X} \\ \mathbf{Y} \end{pmatrix}^{\mathrm{T}}. \tag{2.84}$$

is computed from observation matrices $\mathbf{X} \in \mathbb{R}^{D_x \times N}$ and $\mathbf{Y} \in \mathbb{R}^{D_y \times N}$. $\mathbf{C}_{xy} = \mathbf{C}_{yx}{}^{\mathrm{T}}$ is a between-sets covariance matrix and $\mathbf{C}_{xx}$ and $\mathbf{C}_{yy}$ are within-sets covariance matrices. If $\mathbf{C}_{xx}$ and $\mathbf{C}_{yy}$ are invertible, the product of the eigenvalues ρ_i is equal to the ratio of determinants in Equation 2.83. Consequently, mutual information can be written in terms of canonical correlations (Kay, 1992):

$$I(\mathbf{x}; \mathbf{y}) = -\frac{1}{2} \sum_{i=1}^{D} \ln(1 - \rho_i{}^2). \tag{2.85}$$

To calculate the canonical correlations from observed data, the expectations in Equation 2.82 are replaced by sample-based estimates:

$$\rho_1 \quad = \quad \max_{\mathbf{a},\mathbf{b}} \frac{\mathbf{a}^{\mathrm{T}} \mathbf{C}_{xy} \mathbf{b}}{\sqrt{\mathbf{a}^{\mathrm{T}} \mathbf{C}_{xx} \mathbf{a}} \sqrt{\mathbf{b}^{\mathrm{T}} \mathbf{C}_{yy} \mathbf{b}}}. \tag{2.86}$$

Since the solution is not affected by re-scaling $\mathbf{a}$ or $\mathbf{b}$, the maximization problem is equal to maximizing the numerator subject to

$$\mathbf{a}^{\mathrm{T}} \mathbf{C}_{xx} \mathbf{a} = \mathbf{b}^{\mathrm{T}} \mathbf{C}_{yy} \mathbf{b} = 1. \tag{2.87}$$

As reviewed for example by Bach and Jordan (2003), CCA reduces to solving the following generalized eigenvalue problem:

$$\begin{pmatrix} 0 & \mathbf{C}_{xy} \\ \mathbf{C}_{yx} & 0 \end{pmatrix} \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \end{pmatrix} = \rho \begin{pmatrix} \mathbf{C}_{xx} & 0 \\ 0 & \mathbf{C}_{yy} \end{pmatrix} \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \end{pmatrix}. \tag{2.88}$$

The results is $D_x + D_y$ eigenvalues $\{\rho_1, -\rho_1, \ldots, \rho_D, -\rho_D, 0, \ldots, 0\}$, such that $\rho_1 \geq \rho_2 \geq \cdots \geq \rho_D$. The eigenvectors $\mathbf{A} = (\mathbf{a}_1, \ldots, \mathbf{a}_D)^{\mathrm{T}}$ and $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_D)^{\mathrm{T}}$ corresponding to D non-zero canonical correlations are the basis vectors for the canonical variates $\mathbf{U} = (\mathbf{u}_1, \ldots, \mathbf{u}_D)^{\mathrm{T}} = \mathbf{A}^{\mathrm{T}}\mathbf{X}$ and $\mathbf{V} = (\mathbf{v}_1, \ldots, \mathbf{v}_D)^{\mathrm{T}} = \mathbf{B}^{\mathrm{T}}\mathbf{Y}$.

In order to get decent estimates, the sample size N has to be large enough compared with the dimensionalities D_x and D_y. A standard condition in CCA is $N/(D_x + D_y) \gg 1$. If the ratio is small, the sample covariance matrix $\mathbf{C}_{xy}$ may become ill-conditioned for the eigenvalue problem, leading to an overfitted CCA solution with canonical correlation of exactly one. Furthermore, the sample covariance matrices $\mathbf{C}_{xx}$ and $\mathbf{C}_{yy}$ may also be singular or near singular, leading to unreliable estimates of their inverses. This problem can be alleviated by using regularization in the constraints of Equation 2.87 (Leurgans et al., 1993; De Bie and De Moor, 2003; Hardoon et al., 2004). In practice, a small positive value is added to the diagonal of $\mathbf{C}_{xx}$ and $\mathbf{C}_{yy}$ in the eigenvalue problem.

The applicability of classical CCA is restricted by its reliance on a linear dependence measure. Several authors (Lai and Fyfe, 2000; Akaho, 2001; Melzer et al., 2001) have presented a CCA extension that enables nonlinear transformations using kernel functions. In kernel canonical correlation analysis (KCCA), the data is projected into a feature space of high dimensionality using kernel functions, and CCA is computed in the kernel space. Because of the high dimensionality in the kernel space, KCCA overfits without proper regularization (Bach and Jordan, 2003; Hardoon et al., 2004).

2.8.4 Hierarchical and K-means clustering

There are two basic approaches to hard clustering: *hierarchical* clustering and *partitional* clustering. Hierarchical clustering builds a hierarchy of clusters called *dendrogram* based on the distances $d(\cdot, \cdot)$ between the samples. In the top of the hierarchy, all samples are in one cluster, and in the bottom, all are in their own clusters. In between, there exists a threshold t such that all pairs of clusters have $d(\mathrm{C}_i, \mathrm{C}_j) > t$, where the distance function is extended to distances between clusters. There are several options for the extension, including average of the cluster samples, minimum distance between the samples (*single-linkage*), or maximum distance between the samples (*complete-linkage*).

Hierarchical clustering may start with all samples in their own cluster and proceed by combining two nearest clusters at a time (*agglomerative* or *bottom-up* clustering), or start with all samples in one cluster and proceed by finding the split that gives the largest distance between the two resulting clusters (*divisive* or *top-down* clustering). One step of agglomerative clustering requires N^2 comparisons for N clusters, while an exhaustive search for divisive clustering requires 2^{N-1} comparisons for a cluster of N samples, making the latter approach infeasible for large N.

Partitional clustering directly attempts to find the best single *partition* of the samples into a set of clusters. The standard partitional clustering problem, introduced by MacQueen (1967), is called *K-means* and formulated as follows: Given the set of n points X in $\mathbb{R}^d$ and an integer K, select a set of cluster centers or *centroids* $\mathrm{M} = \{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K\}$ that minimize the potential function

$$\phi(\mathrm{M}; \mathrm{X}) = \sum_{\mathbf{x} \in \mathrm{X}} \min_{\boldsymbol{\mu} \in \mathrm{M}} \|\mathbf{x} - \boldsymbol{\mu}\|^2. \tag{2.89}$$

Assignment of the samples to the closest centroids describes a partitioning of the samples into the clusters as well as a partitioning of the data space into a *Voronoi diagram*. Finding the solution with minimum potential has an exponential time complexity. The standard K-means algorithm for finding a local minimum for the problem was first proposed by Lloyd (1982): In one iteration, each sample is assigned to its nearest centroid, and centroids are moved to the mean of their data points. This two-step process is repeated unit convergence. After each iteration, the potential is guaranteed not to increase, so the algorithm will always converge.

The two obvious problems are how to select the number of clusters and their initial positions. The former is a model selection problem, and a common procedure is to cluster with different values of K and then select one using cross-validation or regularization heuristics. For initialization, the K-means++ method developed by Arthur and Vassilvitskii (2007) has both a theoretical bound on accuracy and good performance in practice compared with other methods.

2.8.5 Expectation-maximization algorithm

The expectation-maximization (EM) algorithm (Dempster et al., 1977) solves maximum-likelihood problems in which the likelihood is assumed to depend on latent (hidden) variables. The most common application of EM is in mixture models, where the observed variables are assumed to be generated by a mixture of latent variables. For Gaussian mixture models, EM can be seen as a probabilistic extension to the K-means algorithm: the centroids in K-means are replaced by Gaussian distributions, and the assignment of the data points is soft instead of hard. EM can also be applied to find the parameters of a hidden Markov model. In this case it is also known as the *forward-backward* or *Baum-Welch algorithm* (Baum, 1972).

In maximum-likelihood estimation, one tries to find model parameters $\boldsymbol{\theta}$ that would maximize the probability of the observed data $\mathbf{X}$. In latent variable models, the observed data is incomplete, that is, it does not contain the instances of the latent variable Y. Let the complete data set be $\mathbf{Z} = (\mathbf{X}, \mathbf{Y})$. The joint distribution for $\mathbf{Z}$ can be calculated from

$$p(\mathbf{z} \mid \boldsymbol{\theta}) = p(\mathbf{x}, \mathbf{y} \mid \boldsymbol{\theta}) = p(\mathbf{x} \mid \boldsymbol{\theta}) p(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}), \tag{2.90}$$

and the complete likelihood function is $p(\mathbf{X}, \mathbf{Y} \mid \boldsymbol{\theta})$.

In the EM algorithm, the first step (E-step) is to find the expected value of the complete-data log-likelihood $\log p(\mathbf{X}, \mathbf{Y} \mid \boldsymbol{\theta})$ with respect to the unknown data $\mathbf{Y}$, seen as a random variable, given the observed data $\mathbf{X}$ and the current parameter estimates $\boldsymbol{\theta}^{i-1}$. The expectation can be written as a function Q:

$$\begin{aligned} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{i-1}) &= E_Y\left[\log p(\mathbf{X}, \mathbf{Y} \mid \boldsymbol{\theta}) \mid \mathbf{X}, \boldsymbol{\theta}^{i-1}\right] \\ &= \int_{\mathbf{y} \in \mathcal{Y}} \log p(\mathbf{X}, \mathbf{y} \mid \boldsymbol{\theta}) p(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta}^{i-1}) d\mathbf{y}, \end{aligned} \tag{2.91}$$

The second step (M-step) is maximization of the expectation computed in the first step, that is:

$$\boldsymbol{\theta}^i = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{i-1}) \tag{2.92}$$

After each iteration, the log-likelihood is guaranteed to increase until the algorithm converges.

In mixture models (Section 2.7.3), the samples come from a mixture of K independent sources c_j with priors π_j. To apply the EM algorithm, the latent variable Y is set to give the assignments of the samples to the components: $y_i = j$, if i^{th} sample $\mathbf{x}_i$ was generated by the source c_j. Given the values, the complete-data log-likelihood becomes

$$\log p(\mathbf{X}, \mathbf{y} \mid \boldsymbol{\theta}) = \sum_{i=1}^{n} \log(p(y_i) p(\mathbf{x}_i \mid y_i)) = \sum_{i=1}^{n} \log(\pi_{y_i} p(\mathbf{x}_i \mid c_{y_i}, \boldsymbol{\theta})). \tag{2.93}$$

To be able to use Equation 2.91, also the marginal distribution for Y is required. For an instance of $\mathbf{y} = (y_1, \ldots, y_n)$,

$$p(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta}) = \prod_{i=1}^{n} p(y_i \mid \mathbf{x}_i, \boldsymbol{\theta}) = \prod_{i=1}^{n} \frac{\pi_{y_i} p(\mathbf{x}_i \mid c_{y_i}, \boldsymbol{\theta})}{\sum_{j=1}^{K} \pi_j p(\mathbf{x}_i | c_j, \boldsymbol{\theta})}. \tag{2.94}$$

Calculating $p(y_i = j \mid \mathbf{x}_i, \boldsymbol{\theta})$ for each component c_j and sample $\mathbf{x}_i$ is the E-step of the algorithm. The next step is to derive new parameters that maximize Q; see Bilmes (1998) for details.

3. Linguistic data and theories

This thesis concerns mainly written language and its representation, text. Most of the language data stored in electronic form is text, which makes it the most important type of input for NLP applications. There are two main reasons for the dominance of text: First, text is a very suitable representation for one-way communication because reading is, in many cases, faster than listening to speech. Second, encoding text in computers is much more efficient than encoding audio. The current memory capacity of computers enables storing large amounts of both uncompressed and compressed text. For example, an English corpus of one million sentences and 24 million word forms takes 127 megabytes in raw ASCII format, and 51 megabytes when encoded by the all-round compressor `gzip`. In comparison, with medium-quality audio compression that uses 64 kbit/s, fifty megabytes is enough for less than two minutes of speech data.

The high amount of information per byte already indicates that the text representation is very compact. As a matter of fact, text can be considered as a compact, symbolic representation for speech. While it is also easy to think the other way around—that is, speech signal as a medium for transferring the discrete symbols of the language—speech is the original and richer representation. Spoken languages, as well as sign languages, predate written languages, and each of us has learned to speak (or sign) before reading or writing. Moreover, stress and tone of the speech can certainly contain relevant information, which is usually disregarded in the text form.[1]

Text data, as any other data, can be considered from two viewpoints related to statistical modeling. First, one can look into the basic properties of the data: What kind of units does it have? What are the distributions of the units? What kind of visible structure is there? Second, one can look into how the data has been generated: What kind of hidden variables affect the distributions and structures? What are the underlying processes of generation?

The first view is that of a machine learning researcher who gets a new data set but knows nothing of its structure or origins. To build a model for unknown data is hard. Evidently, the more knowledge of the data we have, the better chances we have to build a good model. This gets us to the second point of view. As Wintner (2009) argues, computational linguistics should not be only a branch of applied statistics:

> Surely there's something *unique* to the strings that our systems manipulate, something that can be theorized about and can be scientifically investigated.

[1] Of course, neither does the speech signal alone contain all aspects of the act of communication, such as expressions and gestures of the speaker.

Language is a special kind of data in a way that we all use it in an effortless manner for our communication purposes. Unfortunately, it is neither easy to describe how we understand language nor to tell what are the actual cognitive mechanisms that provide us the understanding. This is why input from both linguistics and cognitive sciences is important for NLP.

Many conceptual distinctions and ideas were brought to linguistics by Ferdinand de Saussure (1857–1913), identified as the founder of the school of structuralism. First, he distinguished the abstract language system (*langue* in French) from its external manifestation, which includes the concrete acts of speech (*parole*) and those who participate in the communication. Second, he separated *synchronic* and *diachronic* axes from the study of languages. The former considers language as a complete system at a given point of time, while the latter considers the historical development of languages.

Following the structuralist tradition, the synchronic study of language is divided according to different structures and phenomena of language. Commonly identified subfields include *phonetics* (study of speech production and perception), *phonology* (study of sounds as abstract elements), *morphology* (study of internal structure of words), *syntax* (study of how words are combined to form sentences), *semantics* (study of meaning of words, word combinations, and sentences), *pragmatics* (study of how utterances are used in communicative acts considering the role of context and non-linguistic knowledge), and *discourse* (study of language use in texts larger than a single sentence). While some of the divisions have been questioned, they are useful at least for defining the different aspects of languages.

In addition to pure linguistics, this chapter considers a few results from the subfields of cognitive science: *Psycholinguistics* studies how language is acquired, comprehended, remembered, and produced by humans. Evidence for psycholinguistic theories are collected from experiments that record reaction times of humans, neuropsychological studies, and computer simulations (Harley, 1995). The subfield that especially studies the neural mechanisms that control the language-related processes in the human brain is called *neurolinguistics*.

This chapter is divided into two sections. The first one starts from the "raw" properties of the language data, and gradually moves towards more complex phenomena that can be observed in different levels of language. Special emphasis is given to morphology, which is one of the central topics of the thesis. The second section provides an overview of some influential linguistic theories and grammars, and considers them from the perspective of machine learning and statistical modeling. The descriptions of the linguistic phenomena and theories are based on several textbooks on linguistics (Karlsson, 2008; Leino, 1999; Levinson, 1983; Matthews, 1991; Saeed, 1997), psycholinguistics (Harley, 1995), and natural language processing (Manning and Schütze, 1999; Jurafsky and Martin, 2008; Roark and Sproat, 2007). Earlier attempts to describe central linguistic phenomena from the point of view relevant for statistical modeling include Honkela (1997), Lagus (2000), and Creutz (2006).

3.1 Linguistic units and their relations

In contrast to many other types of natural data (such as measurement of position, temperature, brightness or price) text is *discrete*—there is a finite set of

possible items to observe—and *symbolic*—the items are not comparable as such but refer to some external object. In electronic format, the basic low-level representation of text is a *string*, a finite sequence of symbols (characters) chosen from a fixed alphabet (character set, such as ASCII, ISO 8859-1 or UTF-8).

Unlike other common types of symbolic data (such as records of products bought by the customers of a shop, answers to multiple-choice questions in a questionnaire, or DNA sequences) language is symbolic on *multiple* levels. We already mentioned letters, which exist in *phonemic* writing systems and more or less correspond to the phonemes in speech.[2] But the letters are not the only symbols in text. Written language is often structured as sequences of letters separated from each other by empty space, and those sequences, *words*, are symbols themselves.

For some Asian languages, such as Chinese and Japanese, the writing system is *logosyllabic* and the boundaries of words are not marked in text. In logosyllabic systems, the characters are not symbols for phonemes but *morphemes* or *syllables*. While syllables are purely phonological units (acceptable phoneme sequences), morphemes are yet another type of symbols. They are the smallest meaning-bearing parts of words, and can be identified in most of the languages regardless of the writing system. For example, **birds** has two morphemes, stem **bird** and suffix **s**, which refers to multiple objects. The average number of morphemes per word form indicates the *morphological complexity* of the language. Morphemes and word formation are discussed in more detail in Section 3.1.7.

3.1.1 Distributions of the units

Given the multi-symbolic nature of written language, it can be studied as a sequence of letters, words, or some other units. The choice of units has a remarkable effect on the observed distribution. The alphabet size for letters is small (typically 20–40), and while there are differences in the frequencies of the letters, all of them are likely to occur at least once in a document of a reasonable size. The number of characters in logosyllabic systems can be several thousands, and even a person considered literate may not know all of them. The size of the set of different words, called *vocabulary* or *lexicon*, is at least tens of thousands, but can reach millions. If we consider all the possible sentences of a language, the number of unique items is practically infinite.

The lexicon stores all distinct *types* of linguistic units. An occurrence of a unit in a fragment of text or utterance is called a *token*. Thus the sentence **"The dog chased the cat that chased the mouse"** has six word types (**the**, **dog**, **chased**, **cat**, **that**, **mouse**) and nine word tokens. Considering the sentence as a corpus, the word **the** has count three and frequency $1/3$, the word **chased** count two and frequency $2/9$, and the rest of the word types have count one and frequency $1/9$. However, the term frequency is quite commonly used also for the (unnormalized) counts.

A well-known observation by Zipf (1932, 1949) is that the numbers of occurrences of words follow a *power-law distribution*. Zipf's law, in its simplest form, says that the k^{th} most common word among the words of a language has a frequency that is inversely proportional to k. For example, if the rank of a word is 100, it would occur about once in a text of one hundred words. This means

[2] For direct encoding of phonemes, there are specific alphabets such as the International Phonetic Alphabet, IPA. For example, the phonetic representation for English **bird** is ['bɜːd], while Finnish **lintu** is pronounced as ['lintu].

that few words are frequent and many words are very infrequent, and results in *sparsity* of the data: given a document, even a large one, only a small fraction of possible words occur in it.

Figure 3.1 demonstrates a practical problem with sparsity, namely the high *out-of-vocabulary* (OOV) rates. Assume that we are scanning through a corpus, and collecting all linguistic units (here words or characters) that we encounter into a vocabulary. Given a new chunk of the corpus, we can calculate the ratio of the number of units that we have not encountered before to the total number of units in the chunk. This is the OOV rate. To compare the OOV rates between different languages, a multilingual corpus for which some chunks of text are aligned between languages is required. On the left side of Figure 3.1, we observe a small corpus containing the Universal Declaration of Human Rights. For English, even by the end of the corpus, more than one third of the words encountered in the last lines are previously unseen. For morphologically more complex languages, Finnish and Turkish, the OOV rate is almost 50%. Using characters as units, the OOV rates drop to zero after twenty lines for all languages written with Latin alphabets. For Chinese and Japanese written with logograms, the rate is still about 5% by the end of the corpus.

On the right side of Figure 3.1, the rates in a larger corpus of 2000 sentences are shown for Finnish, German, and English. While the OOV rates drop below 10% for English and below 30% for Finnish by the end of the corpus, the decrease becomes slower all the time. This means that the size of the vocabulary is growing almost linearly with the size of the corpus.

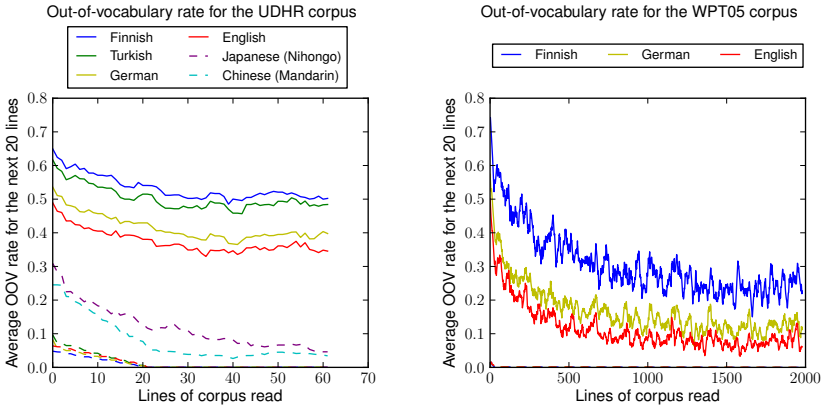


Figure 3.1. Out-of-vocabulary rates for running text for a set of languages. Solid lines show the rates for words and dashed lines for characters. Left: Universal Declaration of Human Rights corpus (Vatanen et al., 2010) aligned by articles. Right: WPT05 corpus (Koehn and Monz, 2005) aligned by sentences.

3.1.2 Meaning and form

Despite the evident problem of sparsity and potentially high OOV rates, most work on natural language processing is based on words. Of course, the words are usually easy to extract from text, but the fundamental reason for their usefulness is different: while the letter **b** and the word **bird** are both technically symbols, only the latter refers directly to something that is useful for understanding what the text is about. To describe this more formally, some ideas and

terminology are needed from semantics (see, e.g., Saeed, 1997).

Saussure considered language as a symbolic system and separated the *form* of the symbols from their *meaning*. The forms and meanings are combined in linguistics units called *signs*. A basic example of a sign is a single word such as **bird**. The *signifier* is the abstract acoustic image of the word, and *signified* is the concept the sign refers to: a feathered, winged, egg-laying, vertebrate animal. An important aspect in languages is the *arbitrariness* of the sign, meaning that there is no specific reason why a particular sign is attached to a particular concept.[3] This is demonstrated by the different forms for the same meaning in different languages: for example, **bird** is **lintu** in Finnish, **fågel** in Swedish, and **oiseau** in French.

While each word has its own capacity for reference, the whole set of words in a language, lexicon, is connected, and the meaning of one word is not independent of the meanings of the other words. The *semantic scope* of a word is defined by the related words. For example, English **sheep** and French **mouton** both refer to the same animal, but English has a separate word, **mutton**, for the meat of the animal. Chinese has a word for the color that covers both the colors called blue and green in English. Some languages, such as Arabic, have *dual* forms in addition to singular and plural, which means that the scope of the Arabic plural differs than the scope of the English plural.

3.1.3 Syntagmatic and paradigmatic relations

From the machine learning perspective, the arbitrariness of the word symbols prevents using similarity functions that can be defined between the two sequences of letters. For example, consider the string edit distance that is the least number of insert, delete, and replace operations required to transform one string to another. Distance d between two words with a very different meaning, such as **talk** and **walk**, can be smaller ($d = 1$) than that of two words of similar meaning, such as **talk** and **chat** ($d = 4$).

Clearly, it is not possible to estimate how close two symbols are with respect to their meaning based solely on their forms. Can then anything related to meaning be extracted from text data alone? Structuralists such as Leonard Bloomfield (1887–1949) and Zellig S. Harris (1909–1992) found a positive answer by a *distributional* approach. In linguistics, the distribution of a unit is all the contexts (such as sentences or neighboring words) in which it can occur. Comparing the distributions or co-occurrences of two units gives information on the relation of their meaning.

Already Saussure had identified two basic types of relations between the signs: *syntagmatic* relation exists between units that co-occur in a sequence (called *syntagm*), while *paradigmatic* relation exists between units that are interchangeable in the sequence. A set of units that have paradigmatic relations form a *paradigm*. These relations are not present only between words, but also between many different linguistic units, as shown by the examples in Figure 3.2.

Harris (1955) found that the positions of morpheme and word breaks could be located surprisingly well from a continuous phoneme sequence by observing the syntagmatic relations of the phonemes. He studied how many different phonemes can occur after the first n phonemes of test utterances. This measure

[3] As an exception from arbitrariness, one may consider onomatopoeic words such as **bang** or **meow**; the equivalent words in Finnish are ***pam*aus** and ***nau*kua** or ***mau*kua**.

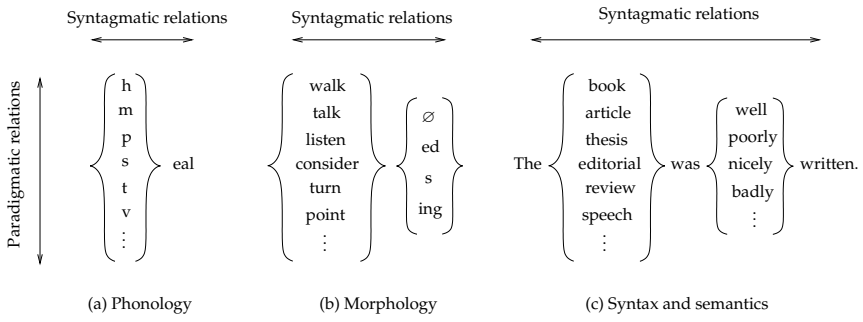


Figure 3.2. Syntagmatic and paradigmatic relations for different units of language. (a) Several consonants appearing before the phoneme sequence [i:l] ("eal") show a phonological paradigm. (b) A set of verbs and their inflectional suffixes form a morphological paradigm. (c) A set of nouns and a set of adverbs have internal paradigmatic relations and syntagmatic relations with the other words in the sentence.

is commonly called *letter successor variety* (LSV). A high LSV value indicates that there are many possible continuations of the observed sequence. Harris demonstrated that a peak in the LSV indicates a boundary of a semantic unit. While he did not have the tools to use his method on large data sets, Hafer and Weiss (1974) later confirmed the practical applicability of the method. LSV is still one of the most common approaches to unsupervised segmentation of words (Section 6.2.4).

Also finding paradigmatic and syntagmatic relations of words from a large corpus is relatively simple (see, e.g., Rapp, 2002). Words that occur in similar context have paradigmatic relations. The contexts of word i can be encoded in co-occurrence vector $\mathbf{x}_i$, where each element gives the number of times a certain other word has occurred in a certain position with respect to word i (e.g., how many times the word **well** is immediately before **written**, or how many times **book** is three words to the left from **written**). Limiting the set of contexts words and the positions to the near context provides a finite-dimensional vector that can be compared with the vectors of other words. In effect, two words that have similar context vectors have paradigmatic relations. Finding syntagmatic relations is even simpler: two words that occur close to each other more often than by chance usually have syntagmatic relations.[4] The use of co-occurrence vectors is the basic idea of vector space models of language, discussed later in Chapter 5. Indeed, the distributional approach for language has given the basis for many statistical and machine learning approaches for language processing, including statistical language modeling (Chapter 4) and lexical unit selection (Chapter 6).

3.1.4 Word forms and lexemes

The term "word" actually has at least three different senses (Matthews, 1991, Ch. 2). The first is that word is a concrete sequence of letters (or, in spoken language, a sequence of phonemes). When referring specifically to this sense, the term *word form* is used.

The second sense is used if the word forms **bird** and **birds** or **run** and **ran** are

[4] However, statistical tests for the co-occurrences return also word pairs that have paradigmatic relations (Rapp, 2002). For example, **hot** and **cold** can co-occur in a sentence such as "Is it hot or cold?".

considered as instances of the *same* word. When looking into a dictionary, they are indexed under the same "dictionary form" of the word. The specific term for this sense is *lexeme*. Each lexeme is an abstract class of word forms; a standard convention is to mark them with the dictionary form using small capital letters (e.g. RUN, BIRD).

Third, when the word form is analyzed according to its grammatical categories (see below), one can say that the word **birds** is plural of the noun BIRD. This sense can be referred to as *grammatical word*.

For a more complex example, consider the word form **matches**. It is an example of an *ambiguous* form. At least three different lexemes can be found for it: MATCH_1 as in **"He won the match"**, MATCH_2 as in **"He had to strike two matches"**, and MATCH_3 as in **"Our system matches with theirs"**. The grammatical word can be identified either as the plural of the noun MATCH_1, plural of the noun MATCH_2, or third person singular of the verb MATCH_3.

3.1.5 Part-of-speech categories

Part-of-speech (POS) categories, identified already by scholars of the ancient Indian and Greek cultures, are a central concept in syntactic as well as morphological analysis. Considering the paradigmatic relations introduced earlier, a set of words in one POS category are those that have their lexemes in a (theoretical) paradigmatic relation based purely on syntactic restrictions of the language.

Some of the POS categories are *open*, that is, they accept new morphemes and words. The open categories are nouns (e.g., **book**, **house**), verbs (**walk**, **take**), adjectives (**red**, **compelling**), adverbs (**well**, **easily**), and interjections (**bye**, **oh**). In contrast, *closed* categories contain usually a fixed set of words. Typical closed categories are pronouns (e.g., **I**, **them**), prepositions (**in**, **without**), conjunctions (**and**, **but**), determiners (**a**, **that**). Words in closed categories are considered to be more at the core of the syntax, while words in open categories usually carry the primary semantic content in a sentence.

The set of part-of-speech categories varies depending on the language and the granularity of the classification. For example, in addition to prepositions, there can be postpositions; pronouns can be divided into subcategories such as possessive pronouns (**mine**, **its**) and personal pronouns (**I**, **you**, **he**); determiners into articles (**a**, **an**, and **the**) and quantifiers (**all**, **some**); and nouns into singular and plural nouns. Thus the number of different tags in the POS tagsets vary from few dozens to more than one thousand (Jurafsky and Martin, 2008, Ch. 5).

Due to their importance in syntactic analysis, classifying words into their POS categories is a common task in NLP. The methods include rule-based taggers, supervised hidden Markov model taggers, as well as unsupervised taggers based on clustering methods.

However, regardless of the importance of the part-of-speech categorization for linguistic description, the existence of such grammatical categories of words in the human mind is controversial. A recent review by Vigliocco et al. (2011) on psycho- and neurolinguistic studies of verbs and nouns concludes that while object words (nouns) and action words (typically verbs) have clear neural separability and different processing demands, grammatical category is not likely to be an organizational principle of knowledge in the brain. Instead, the observed categories can emerge from the semantic and pragmatic differences of the words and distributional cues in language.

3.1.6 Constituency

Constituency is a basic notion in the structuralist approach to syntax (Schönefeld, 2006). Two or more words that act like a single unit in a sentence are called *constituents*. There are two simple ways of identifying words that form constituents: they can be *substituted* by another constituent of the same type (i.e., with a paradigmatic relation), or sometimes *permuted* within the sentence. For example, in **"The book was well written."**, the phrase **"The book"** can be substituted by **"His new book"** or **"The book I read yesterday"**. Thus it is a constituent of the sentence. The sentence **"By all means you can ask me again."** can be permuted by moving the phrase **"by all means"** to the middle (**"You can, by all means, ask me again."**) or to the end (**"You can ask me again by all means."**).

The procedure of stepwise segmentation of a sentence to its smaller constituents is called *immediate-constituent* (IC) analysis (Wells, 1947). For example, in the following, the constituents of a sentence are marked with square brackets (called *bracketing*):

[[The book] [I [read yesterday]]] [was [well written]]

As the structure is hierarchical, it can also be drawn as a tree. Whenever constituents are grouped together, they make up a *construction*. The top-level construction is the sentence itself.

Similar to paradigmatic relations, constituency is not observed only for syntax, but also for morphology. While permutation tests rarely apply because of strict morphotactic rules, substitution tests can identify morphological constructions. Several constituents and constructions can be observed in a complex word form such as **unfeignedness**. A reasonable IC analysis for it would be **[[un [feign ed]] ness]**, in which each construction can occur as a word and the constituents are replaceable. For example, **feign** could be replaced with **affect**, **feigned** with **aware**, and **unfeigned** with **mad**.

3.1.7 Morphology

The internal structure of words can be looked at from several viewpoints. The first one is to consider the different ways that word forms can be related. The second is to study the linguistic units that occur within word forms. Moreover, different languages can be categorized according to the morphological phenomena that is observed in them.

3.1.7.1 Inflection, derivation, and compounding

The set of word forms that have the same lexeme (e.g., **bird** and **birds** or **run**, **runs**, **ran**, and **running**) are called *inflections* of the lexeme. The dictionary form—usually the simplest form such as singular **bird** and present tense **run**—is called *lemma* of the lexeme. The sets of lexemes that are inflected in a similar manner form *inflectional paradigms*. The so-called word-and-paradigm model, described below, is a description of these paradigms. However, inflection of a lexeme is not the only reason for two word forms to be related.

First, consider **matching** in **"I cannot find a matching pair of socks"** and **unmatched** in **"He has unmatched talent"**. Both word forms are related to the

lexeme MATCH$_3$ in Section 3.1.4 above, but they are not verbs but adjectives. Moreover, in most dictionaries, they have their own entries. Also the adjectives **matched** and **unmatched** are related, but are word forms of different lexemes because of their difference in meaning. A more complex lexeme (such as **unmatched**) is called a *derivation* of the simpler lexeme (**matched**).

Second, consider **matchbox** and **matchmaker**. They have compound lexemes MATCHBOX and MATCHMAKER, which are related to simple lexemes MATCH$_2$ and BOX, and MATCH$_3$ and MAKER, respectively. Also **matchbox paper** can be considered as a compound lexeme of MATCHBOX and PAPER, although the two words are not written together. The formation of new, complex lexemes from more than one simple or compound lexemes is called *compounding*.

Derivation and compounding together are called *word formation* in contrast to inflection. However, the distinction between inflections and derivations is not always evident (Matthews, 1991, Ch. 3).

3.1.7.2 Inflectional paradigms

The oldest and the most well-known approach to morphology is called *word-and-paradigm* (WP) (Hockett, 1954). As it is a description for inflectional morphology, word formation is not considered. The goal is to generalize the patterns that occur when inflecting individual words. For example, in English, the verbs MATCH and WAIT have analogous inflections: **match**/**wait** (infinitive), **matches**/**waits** (3rd person present tense), **matching**/**waiting** (present participle), **matched**/**waited** (past tense), and **matched**/**waited** (past participle). This can be named as verb paradigm I. Most of the verbs for which the infinite form ends with an **e**, for example BAKE, are members of the verb paradigm II: **bake**–**bakes**–**baking**–**baked**–**baked**. Next we have irregular verbs that are inflected similar to CUT (paradigm III), similar to TELL (paradigm IV), and similar to WRITE, and so on. Any new lexeme that is a verb belongs to one of the paradigms; nouns and adjectives have their own paradigms. Thus the components of the WP model are, as suggested by the name, the words (lexemes) and the inflectional paradigms.

3.1.7.3 Morphemes

An alternative way to look at the structure of word forms is to observe *morphemes*, the smallest meaning-bearing units of the language, and study how they are combined to form words. For a complex example, consider again the word **unfeignedness**. It consists of a number of segments that (a) occur frequently in other English word forms, and (b) add something to the meaning of the full word form: **un**, **feign**, **ed**, and **ness**. Any other segment of the word does not fulfill both of these criteria. Using the Saussurean terminology, the segments above are the signifiers of some signs. Depending of the exact theory, the morphemes are either these minimal forms, their meanings, or the form–meaning pairs (signs). The terminology used in this thesis is that the phonetic or orthographic surface forms (signifiers) are called *morphs*, and morphemes refer to the more abstract combinations of form and meaning. Similarly to lexemes, morphemes will be marked with small capital letters. For example, **walked** has two morphs, **walk** and **ed**, corresponding to abstract morphemes that refer to the meaning of walking (WALK) and past tense (+PAST), respectively.

Morphemes are called *free* or *bound* depending on whether they can occur in-

dependently as a word form or not. For example, FEIGN and WALK are free morphemes, while UN+, +NESS and +PAST are bound morphemes. Bound morphemes are either affixes or so called fossilized or "cranberry" morphemes (consider **cran** in **cranberry**) that cannot be assigned a clear meaning or a grammatical function. The part of the word that a bound morpheme attaches to is called *stem*. A stem may correspond to a complex lexeme, whereas *root* is the minimal part of the stem that corresponds to a single simple lexeme. For example, **buildings** has the stem **building** and the root **build**. Affixes are divided into *prefixes*, *suffixes*, *infixes*, and *circumfixes*, depending on whether they occur before, after, in between, or both after and before the stem, respectively.

As morpheme-based approaches consider also derivation and compounding, they are more detailed than the word-and-paradigm model. Two particular morpheme-based models, called *item-and-arrangement* and *item-and-process*, are discussed in Section 3.2.3.

3.1.7.4 Allomorphy and syncretism

In general, there is no one-to-one correspondence (bijection) between morphemes and morphs. Sometimes one morpheme can correspond to several morphs, and sometimes one morph can be a realization of many potential morphemes.

Two or more morphs that are the formal variants of the same morpheme are called *allomorphs*. Often the allomorphs of the same morpheme occur in *complementary distributions*: if one variant occurs in some context, the other variants will not occur in the same context.[5] For example, the allomorph **d** of +PAST can occur only after verbs that end with an "**e**", while **ed** can occur only if the verb does *not* end with an "**e**".

A common reason for allomorphy are the phonological processes that occur at morph or word boundaries, called *sandhi*. The processes include assimilation (**invite +ed** → **invited**; **wife +s** → **wives**), as well as insertion (**fish +s** → **fishes**). Some other *morphophonological* phenomena that produce allomorphic variants include the following:

- In *consonant gradation*, the consonants of the stem alternate between various grades. Finnish examples: **ta*kk*a** (fireplace), **ta*k*a+n** (fireplace + possessive case); **sa*d*e** (rain), **sa*t*ee+ssa** (rain + inessive case).
- In *vowel harmony*, the vowel in a suffix is changed to match some property (e.g., height, backness, roundedness) of a vowels in the stem. Finnish example: **alu+*ssa*** (beginning + inessive case), **äly+*ssä*** (intelligence + inessive case).
- In *apophony* or *ablaut*, vowels (or sometimes consonants) alternate in inflections. English example: **s*i*ng–s*a*ng–s*u*ng**; Swedish example: **fl*y*ga** (fly), **fl*ö*g** (fly + past tense), **fl*u*git** (fly + present/past participle).

The case in which distinct morphological forms of a word have the same surface realization is called *syncretism*. In one type of syncretism, the grammatical category of a word does not show in the form. For example, the pronoun YOU has the same form independent of the grammatical case, whereas SHE or HE have distinct nominative and accusative forms (**"you saw her"**; **"she saw you"**),

[5] An example of allomorphs in *non-complementary* distribution are the third person possessive suffixes **+an** and **+nsa** in Finnish. For instance, "in his car" can be either **auto+ssa+an** or **auto+ssa+nsa** (Karlsson, 2008, p. 95). The style of the latter form is more colloquial.

and the verb PUT has the same form regardless of the tense, unlike most of the other verbs. In another type of syncretism, two morphemes have the same surface forms. That is, they are *homographs* (in written language) or *homophones* (in spoken language). An example is English plural and 3rd person singular, which are both realized either by the suffix **s** or **es**.

Syncretism is one cause for morphological *ambiguity* of the word form. For example, **bites** may either be a noun in plural form or a verb in 3rd person singular form. However, the full word form can be ambiguous even without syncretism. For example, the Finnish word **istuin** may refer either to the noun "seat" or the verb "sit" in 1st person past tense ("I sat"). The context of the word is required to solve the potential ambiguity.

3.1.7.5 Morphological typology

The richness and type of morphological phenomena vary between different languages. Specifically, some languages are *analytic* or *isolating*, where each word is a single morpheme, and some are *synthetic*, where there may be several morphemes per word. *Polysynthetic* languages have particularly many bound morphemes per word, to the extent that the difference between words and sentences is not evident. The number of morphemes per word, or *index of synthesis*, measures this dimension. In the one extreme, isolating languages include, for example, Vietnamese (index of synthesis 1.06).[6] In the other extreme, there are polysynthetic languages such as Eskimo languages (index 3.72). English is usually considered a mixed type; its index of synthesis is 1.68. Synthetic languages include Russian (index 3.33), Turkish (index 2.86), and Uralic languages such as Finnish.

Synthetic languages can furthermore be divided into *agglutinative* (*concatenative*) and *fusional* languages depending on the typical processes of derivation and inflection. In the former case, the morphs of a word are clearly separable from each other. In the latter case, the surface form does not have separable morphs that would correspond to the morphemes, as it is the case with several irregular verbs in English. However, most languages are mixed types also in this dimension. The index of fusion measures the ratio of morphemes and morphs: it is low for agglutinative and high for fusional languages. Apart from the irregular inflections, English is mostly agglutinative. Strongly fusional languages include French and Russian. Another type of non-concatenative process is *transfixation*, present for example in Arabic languages, where consonantal roots are modified by vowel patterns.

3.1.7.6 Psycholinguistic results

Psycholinguistic studies on morphology concentrate on the question of how morphologically complex words are processed (Harley, 1995, pp. 287–288). One line of research focuses on properties of words that affect the processing latencies of the test subjects. Such properties include cumulative base frequency, i.e., the summative frequency of all the inflectional variants of a single stem (Taft, 1979; Baayen et al., 1997; Bertram et al., 2000), surface frequency of the whole word form (Alegre and Gordon, 1999; Sereno and Jongman, 1997; Baayen et al., 1997; New et al., 2004; Laine et al., 1995), and morphological family size, i.e., the

[6] The mentioned numbers for the indexes of synthesis are from Karlsson (2008, p. 118).

number of derivations and compounds in which the stem occurs as a constituent (Bertram et al., 2000).

The fact that the surface frequency of a complex word is predictive for processing times, independently of the frequencies of its constituents indicates either separate independent representations for the complex words or storage of the probabilities of the co-occurrences of the constituents (Baayen, 2007). However, studies of Finnish word recognition have associated a processing cost (i.e., long reaction times and high error rates) with inflected Finnish nouns in comparison to matched monomorphemic nouns (Niemi et al., 1994; Lehtonen and Laine, 2003), which indicates the existence of morphological decomposition for most Finnish inflected words. A possible exception are very high frequency inflected nouns (Soveri et al., 2007). The correlations of entropy-based measures of paradigmatic complexity with processing times (Kostić, 1991; Moscoso del Prado Martín et al., 2004; Milin et al., 2009) give further evidence that probabilities play a strong role in human processing of morphology.

3.1.8 Syntax

In the study of syntax, the largest linguistic form considered is a *sentence*, which is seen as an independent linguistic form that is not included in any larger linguistic form. There are several types of syntactic and sentence analyses; a few common approaches are mentioned here. The word order varies between languages, which causes problems for certain computational approaches to syntax. Another common problem for computational models are the long-distance dependencies between the words.

3.1.8.1 Phrase categories

In syntactic constituency, constituents that are grouped around words of certain POS categories form phrases that are categorized according to that part-of-speech. For example, noun phrases (NP) contain at least one noun or pronoun and occur in similar distributions as single nouns. Other typical phrase categories are adjective phrases (AP), adverb phrases (ADVP), pre- and postpositional phrases (PP), and verb phrases (VP).

The immediate-constituent (IC) analysis of a sentence commonly includes classification of the constituents by their grammatical features, such as parts-of-speech and phrase categories:

$$\textbf{[[[The}_{\textbf{Det}}\ \textbf{book}_{\textbf{Noun}}\textbf{]}_{\textbf{NP}}\ \textbf{[I}_{\textbf{Pron}}\ \textbf{[read}_{\textbf{Verb}}\ \textbf{yesterday}_{\textbf{Adv}}\textbf{]}_{\textbf{VP}}\textbf{]}_{\textbf{PP}}\textbf{]}_{\textbf{NP}}$$
$$\textbf{[was}_{\textbf{Verb}}\ \textbf{[well}_{\textbf{Adv}}\ \textbf{written}_{\textbf{Adj}}\textbf{]}_{\textbf{AP}}\textbf{]}_{\textbf{VP}}\textbf{]}_{\textbf{S}}$$

The above notation can also be represented as a tree. If the category labels and words are considered as nodes, the tree of this form is a parse tree from a *phrase structure grammar*, discussed later in Section 3.2.4.

3.1.8.2 Sentence elements

Traditional sentence analysis divides the words in a sentence into different sentence elements. The following two examples illustrate the main elements:

$$\textbf{[The book]}_{\textbf{subject}}\ \textbf{[was]}_{\textbf{predicate}}\ \textbf{[good]}_{\textbf{predicative}}$$

[She]$_{\text{subject}}$ [kicked]$_{\text{predicate}}$ [the ball]$_{\text{object}}$ [hard]$_{\text{adverbial}}$

In contrast to the IC analysis, this type of analysis is flat except for possible division into subject and predicate parts of the sentence. For many languages, a subject and a predicate are also the minimum number of elements that a sentence contains.[7] The predicate determines the minimum number of the elements; in addition to subject and object it may require other arguments. These complimentary parts, such as **good** in the first example, are called *predicatives* or *complements*. The non-complimentary parts, such as the adverbial **hard** in the second example, are called *modifiers*. The terminology from sentence analysis is often used in dependency grammars (Section 3.2.5).

3.1.8.3 Word order

A considerable amount of information in a sentence may be encoded by the order of the words. In most languages, the word order has several functions. First, for each word, it can mark the sentence element in which the word belongs to. At the same time, it can distinguish between different sentence types (declarative, interrogative, exclamatory, or imperative). Outside syntax, it has discourse, thematic, and pragmatic functions.

The order in which subject (S), object (O), and verb (V) occur in simple, declarative sentences determines the basic word order of the language. Of the six theoretical options, the three most common are SOV, SVO, and VSO, in this order. All of the three have subject before object. These word orders are used in over 90% of the world's languages. VOS order is less common than any of the three, and OVS and OSV are very rare.

Only few languages have a free word order, meaning that there is no single dominating word order even in declarative sentences. One such exception is Basque. A more common case is that there is a standard order, but it can change according to discourse and pragmatic functions, as in Finnish. In both cases, the grammatical functions of the words have to be determined by other means, usually by morphological marking.

3.1.8.4 Agreement and government

A typical syntactic phenomenon is that a word changes its form depending on the other words to which it relates. This is called *agreement* or *concord*. Common agreements are grammatical person (**I *am*** versus **he *is***) grammatical number (**one *book*** versus **pile of *books***), grammatical gender (**Mary blamed *herself*** versus **John blamed *himself***), and grammatical case (***I*** **met *her*** versus ***she*** **met *me***).

In *government* or *rection*, a head word affects the selection of grammatical features for a set of dependent words. A typical example is *case government*, in which the case inflections in a noun or adjective phrase are determined by the governing verb. In English, government shows in how a certain verb requires a certain preposition. For instance, the Finnish verb **puhua** requires the elative case, while the English **speak** requires either the preposition **about** or the preposition **of**:

Finnish: **he puhuivat autois*ta***

[7] In Finnish, a sentence may have only a predicate, as in **"Sataa."** ("It rains.").

English: **they spoke *about* cars** or **they spoke *of* cars**

The government is inherited by derived words (**puhe autoista**, **speech about cars**). It can affect also several words in the dependent phrase: **"he puhuivat vanhoi*sta* autoi*sta* ja juni*sta*"** ("they spoke about old cars and trains").

3.1.8.5 Long-distance dependencies

There are various syntactic phenomena that result in direct dependencies between words that occur far apart in a sentence. A simple example is the combination of a noun phrase that includes long modifiers, such as a relative clause, and agreement between subject and verb: ***The man* who talked to the police officers *was* blaming *himself*.** That is, **the man**, **was**, and **himself** match in grammatical number, person, and gender.

Also free word order can easily induce long-distance dependencies. In the Finnish sentence **"He *puhuivat* monta päivää autoi*sta*."** ("They spoke for many days about cars."), the adverbial phrase **monta päivää** (for many days) separates the governing verb from the affected noun. In **"Töi*stä* koti*in* hän *juoksee* joka päivä."** ("From work to home he runs every day."), the non-standard word order for the arguments of the verb emphasizes the direction or path of the run. At the same, objects of the sentence have moved far away from the predicate.

Wh-movement locates the object of an interrogative sentence to the beginning of the sentences, while the verb on which it is dependent is in the end. For example, in the declarative sentence **"He read a book"**, the object follows the predicate, while in the question **"What book did he read?"** the object comes first.

Long-distance dependencies are problematic especially for models that assume that only nearby words are dependent, such as the n-gram models discussed in Section 4.2. If the word w has a strong dependency on the word v (i.e., $p(w \mid v) \gg p(w)$ or $p(w \mid v) \ll p(w)$), such a model cannot use the dependency if the observed sequence $v\boldsymbol{h}w$ has an arbitrary long sequence of units $\boldsymbol{h}$ between v and w.

3.1.9 Semantics and pragmatics

When building a model based solely on text data, there is no direct semantic information available: there are the signifiers (forms) but not the signified (meanings). However, semantics evidently has a strong effect on the distributions of the forms.

Traditionally, semantics studies meaning of language "abstracted away from the users" (Saeed, 1997): it considers the meaning of the sentence, phrase, or word as an abstract unit, not the meaning that the particular speaker or writer intended for it. The part of meaning that depends on a specific listener or reader is then considered in pragmatics. Pragmatics also considers how linguistic contexts longer than one sentence affect the meaning, thus including discourse and text analysis.

If we start by considering a meaning of single words or morphemes, it is obvious that the grammatical categories of the words reflect their meaning. Nouns refer to individual or groups of objects, verbs to actions, and adjectives to properties of objects or actions. This kind of meaning that exists without reference to any sentence is called *lexical meaning*. In contrast, for many closed category words, such as prepositions or conjunctions, meaning is harder to describe with-

out the context of the occurrence. The same applies to many inflectional and derivational bound morphemes. They have only a *grammatical meaning*. Along this division, morphemes (and words) are called *lexical* or *content* morphemes (words) and *grammatical* or *function* morphemes (words).

3.1.9.1 Lexical semantics

Lexical semantics considers the meaning of individual words, or more specifically, lexemes. The knowledge about a certain lexeme is sometimes separated into *linguistic knowledge* and *encyclopedic* or *world knowledge*. Linguistic knowledge provides whatever is needed to use the word in a sentence, whereas encyclopedic knowledge is about the way the world is. For example, a person may say **"I saw whales yesterday"** without knowing whether a whale is a fish or a mammal, and another person can understand the sentence even if his world knowledge on such issues differs from the speaker.

One approach to linguistic knowledge is to consider what the *lexical relations* between words are. As we have already seen (Section 3.1.4), a single word form can have multiple senses. In *homonymy*, there are two unrelated senses of the same word (e.g., **bank** account versus **bank** of river). In *polysemy*, the word has multiple related senses (e.g., **branch** of tree and **branch** of a company). The distinction is not always clear, and sometimes the term polysemy is used to refer to both. *Lexical field* is an abstract categorization for a group of lexemes which belong to a specific area of knowledge. For example, BANK_1 can belong to the field of economy together with LOAN, and BANK_2 to the field of geography together with SWAMP. The task of identifying the correct sense of the word instance based on its context is called word sense disambiguation.

Other commonly considered lexical relations include *synonymy*, where different words have an identical or almost identical meaning (e.g., **buy** and **purchase**), and *antonymy*, where two words have opposite meanings, *hyponymy* and *hypernymy* that are relations of inclusion (e.g., **sparrow** is a hyponym of **bird**; equivalently **bird** is a *hypernym* of **sparrow**), and *meronymy*, which means part-whole relationship (e.g., **wheel** is a meronym of **car**). The lexical relations are commonly used in various ontologies and lexical databases. One of the most popular ones is WordNet (Miller, 1995; Fellbaum, 1998), which collects glosses, synonyms and antonyms, as well as hyponymy and meronymy relations for English words. Because the relations indicate similarity, the databases provide means for direct evaluation of representation learning (Section 5.3).

3.1.9.2 Reference, sense, and concepts

The previous subsection considered the kind of knowledge of words that is required to write a dictionary. From the cognitive science point of view, the central question of semantics is what *is* a meaning. This question has been approached from two directions.

In the first approach, called *reference* or *denotation*, the word names individuals (objects or persons), sets of individuals, actions, or their properties in the real world. While this is intuitive enough for many words, it has problems explaining for example function words, fictional and non-existing objects, abstract terms, as well as expressions that denote the same thing while having a clear difference in meaning.

The other dimension of the meaning is called *sense*. A typical assumption

is that the sense of a word is its mental representation in the speakers mind. The representations are called conceptual elements or simply *concepts*. Concepts that correspond to a single word are called *lexicalized* concepts. The traditional, logic-based approach for concepts is to define them by *necessary and sufficient conditions*. For example, one may define the concept WOMAN by listing a set of attributes that are necessary and sufficient for some x being a woman; for example x is human, x is adult, and x is female. In practice, such definitions are often problematic, and speakers are likely to neither agree on the set of conditions, nor even know all characteristics of the objects they are speaking of. A modern, influential theory on concepts is the *prototype theory* based on the work by Rosch (1973, 1975). Her psycholinguistic experiments showed that people tend to give graded categorization for objects. For example, most people consider CHAIR to be a better example of FURNITURE than BED let alone PIANO, and SPARROW to be a better example of BIRD than PENGUIN is. The central members of each category are called its prototypes.

The attachment of a symbol to a certain meaning (either by reference or by concept formation) is called *symbol grounding*. Without solving the problem of symbol grounding, a machine can only learn relations between words, not the relations between the words and the world.

3.1.9.3 Compositional semantics

Compositional semantics study how the meaning of single linguistic units are combined, from two-word terms to phrases and complete sentences and even larger units.

The traditional approach to compositional semantics is to infer *propositions* from sentences. Propositions are descriptions of states of affairs. Two different sentences can indicate the same proposition:

(1) **Shakespeare is the author of Hamlet.**
(2) **Shakespeare wrote Hamlet.**

Such two sentences can be considered synonymous. Other types of sentence relations include entailment, presupposition, tautology and contradiction.

Sentence meanings are considered mostly in *formal semantics* (see Saeed, 1997, Ch. 10). Formal semantics uses logical languages, such as lambda calculus, to represent the meanings. Computational approaches for formal semantics are described by Jurafsky and Martin (2008, Ch. 17–18). There are two main problems in formal semantics: *non-compositionality of language* and *non-literal use of language*.

The problem of non-compositionality refers to that the meaning of a linguistic unit is often not compositional: the meaning of the whole cannot be strictly predicted from the meaning of its parts (Manning and Schütze, 1999, p. 110). When a fixed sequence of words (such as **strong coffee**) is dominantly utilized to convey a certain meaning instead of semantically close alternatives (?**powerful coffee**), the sequence is called a *collocation*. Collocations are mostly compositional in meaning. However, there is often an additional semantic component that cannot be predicted. For example, **white** refers to very different colors in **white paper**, **white skin**, and **white wine** (cf. Gärdenfors, 2000). *Idioms* are expressions or phrases for which the meaning is completely non-compositional, such as **kick the bucket** as an informal euphemism for dying. Individual words of the idiom may also cease to have independent meanings, as **kith** in **kith and**

kin. Non-compositionality is one of the main reasons to use lexical units that encompass multiple words in NLP (Section 6.2.5).

Apart from being non-compositional, many idioms are examples of use of non-literal or figurative meaning. When someone is told in Finnish **"Voit heittää sillä vesilintua"** ("you can throw it at a waterbird"), the object in concern is considered useless, not that it is useful for waterfowl hunting. Figurative expressions are common also outside idioms. Instead of **"I'm hungry"**, one can say **"I'm starving"** or **"I could eat a horse"** (Saeed, 1997, p. 15), and neither of the latter are taken literally. A common source of non-literal expressions, including idioms, are *metaphors* (Saeed, 1997, Ch. 11). A metaphor uses a concept from one domain (source domain) to describe something in another domain (target domain) by analogy. For example, a recent economy news article stated that **"if the euro fails, bank lending would freeze worldwide, stock markets would likely crash and Europe's economies would crater"**. Here the stock markets and economies are seen as flying objects and bank lending as a stream of water. Lakoff and Johnson (2003) demonstrate the pervasiveness of metaphors in ordinary language and argue that conceptual metaphors are an important factor in organization of human experience. Thus dealing with metaphors is a crucial challenge for any task that involves natural language understanding.

3.1.9.4 Pragmatics and discourse

The field of pragmatics studies how the speakers' knowledge of the world interacts with the use of language. However, the division between pragmatics and semantics is often vague and there is a diversity of possible definitions (cf. Levinson, 1983, Ch. 1). Especially non-literal meanings, short-hands, metaphors, metonymy (concept is called by a name of something associated to it), and synecdoches (the name of a part is used to refer to the whole) are on the border of semantics and pragmatics. While traditionally left to pragmatics, they are emphasized in the theories of cognitive semantics.

Another area of study in pragmatics is discourse. Practical problems of discourse that have been studied in language processing include discourse segmentation and reference resolution (see Jurafsky and Martin, 2008, Ch. 21). In the former, a document is separated into a sequence of sections or topics. Finding the changes of topics within a document is relevant, for example, in information retrieval (Chapter 5) and language model adaptation (Section 4.3.4). In the latter, the task is either finding expressions that refer to the same entity (coreference resolution) or simply the single preceding entity that a pronoun refers to (anaphora resolution). Reference resolution is relevant for dialogue systems and applications such as question answering and text summarization.

3.2 On theories of grammar

While the number of utterances in one language is practically infinite, there are strong restrictions on how the units of the language can be combined. For example, consider the sentence **"The _ was well written"** from Figure 3.2(c) (page 72). Two types of restrictions can be identified for the second word. Neither the word **compelling** nor **books** fit into the place because the preceding article and the following verb require that the word is a singular noun. The word **fireplace** does

not fit into the place because the sentence makes sense only if the first word is something that can be written. The former restriction is syntactic, while the latter is semantic. Moreover, there are similar restrictions for morphological relations. The list of stems in Figure 3.2(b) cannot include **wish** due to phonological restrictions (**wishes**, not *__wishs__), and the suffixes cannot include **-ly** due to morphotactic restrictions. There are also morphosyntactic restrictions, in which morphology and syntax interact. For example, in **"He walk_ home"**, **walk** has to be followed by either suffix **s** or **ed**.

A large part of linguistic theories concentrate on explaining these restrictions. The speakers of the language have to know them, and the knowledge is at least partially unconscious. Depending on the linguistic theory, this mental knowledge, *grammar*, is thought to be encoded as rules, constraints, patterns, or constructions. Usually also the lexicon of words or morphemes is considered to be part of the grammar, and part of the knowledge may be stored within the lexicon.

3.2.1 Views on the scope of grammar

The structuralist tradition has excluded semantics from the grammar. The reason is quite obvious: to explain what is the difference between **fireplace** and **book** requires information on the real-world objects that they refer to, whereas the difference between **books** and **book** can be explained with one attribute (number of objects). For instance, Bloomfield (1935) considered that the study of meanings is best left to sciences other than linguistics. In addition to semantics, also phonetics and pragmatics are often excluded from grammar, which is then used to refer to the set of syntactic, morphosyntactic, and morphotactic rules.

Chomsky (1965) made a fundamental distinction between the knowledge of language, called *competence*, and the actual use of language in concrete situations, called *performance*. In this view, competence is tied with the notion of *grammaticalness*: a sentence is called grammatical if the generative grammar of the language can produce it. A grammatical sentence may not make sense semantically—as in the famous example **"colourless green ideas sleep furiously"** (Chomsky, 1957, p. 15)—or be easily interpretable—as in **"the man who the boy who the students recognized pointed out is a friend of mine"** (Chomsky, 1965, p. 11). For separating whether sentences are likely to be really used or not, Chomsky (1965) uses the term *acceptability*, and considers it to belong to the study of performance.

Chomsky also considered syntax as the core area of grammar. While he proposed that there are three main components in the grammar—syntactic, phonological, and semantic—he argued that the other two rely on the syntactic component (Chomsky, 1965, p. 16):

> Consequently, the syntactic component of a grammar must specify, for each sentence, a *deep structure* that determines its semantic interpretation and a *surface structure* that determines its phonetic interpretation. The first of these is interpreted by the semantic component; the second, by the phonological component.

Such a view that identifies intermediate layers of representation in the language processing is called *multistratal* (multilayered, derivational) opposed to *monostratal* (non-derivational).

The structuralist and Chomskyan views have been challenged by *cognitive lin-*

guists. The cognitive theories of grammar differ from the traditional structuralist and generative approaches in multiple ways (Lakoff, 1987; Croft and Cruse, 2004; Goldberg, 2006):

- **Competence and performance**: Cognitive grammars are often *usage-based*: The properties of the use of grammatical expressions, in particular their meaning and frequency, affect the representation of the expressions. Thus competence and performance cannot be separated.
- **Grammar and cognition**: Grammar is not a separate module independent from the rest of the cognition.
- **Rules and lexicon:** Grammatical constructions are not epiphenomena arising from the use of generative rules but have a real cognitive status. Grammar and lexicon are not separate but form a continuum. Linguistic processing is monostratal.
- **Role of semantics**: Semantics is a core part of the grammar, inseparable from syntax and morphology. Cognitive semantics, including mental spaces and metaphoric models, are required to describe the meanings of grammatical constructions. Grammars use prototype-based categorization. The concept of motivation is needed to account for the regularities of grammar. Many syntactic properties of grammatical constructions are consequences of their meanings. Syntactic categories are neither autonomous nor completely predictable from semantic considerations, but their central subcategories are predictable, and the non-central subcategories are motivated extensions of the central subcategories. The meanings of grammatical constructions cannot be computed from the meanings of their constituents, but they are motivated by the meanings of the constituents.

Construction grammars, described in Section 3.2.7, provide an example of cognitive theories.

3.2.2 Poverty of the stimulus and universal grammar

Children's ability to learn their first language in an effortless manner, based only on the utterances that they hear and the related situational contexts, is a central question of interest for linguists. One specific question has been whether humans have an innate language-acquisition device that allows for quick language learning (Harley, 1995, Ch. 10).

The main argument for the nativist view is called the *poverty of the stimulus*. The argument is that given only the limited data available to children, the grammar of the language is unlearnable without some innate linguistic capacity. Specifically, the data is limited in the sense that there is only *positive evidence*, that is sentences that are grammatically correct, and no *negative evidence*, that is sentences that are not grammatical. The theoretical ground for the argument was provided by Gold (1967), who proved that any formal language that includes infinite recursion is unlearnable from positive evidence alone.

The innate knowledge of language was formulated by Chomsky as a *universal grammar*. The universal grammar includes the rules of the grammar that all possible natural languages have, in some manner "hard-wired" into the brain. In terms of machine learning, the universal grammar can be thought of as a parametric model of language. The parameters are learned from the positive

evidence that the child acquires. When the parameters are learned, one point hypothesis is picked up from the hypothesis class of the universal grammar, which is the set of all potential human languages.

Obviously, empiricists also agree that there are *some* limitations on the kind of language that humans can learn or even that the language ability is innate. The disagreement is on whether the innateness or the limitations are due to a language-specific devise or the general cognitive apparatus of humans.

The soundness of the poverty of the stimulus has been questioned from several viewpoints. For the premises of the argument, it is not clear that humans are capable of infinite recursion even without considering limitations in the working memory. Such constraints are certainly present in data: Karlsson (2007) found the maximal degree of center-embedding (see Section 3.2.4) in seven European languages to be three. Moreover, indirect negative evidence may be available as forms of absence of some language patterns that would be considered probable if they were grammatical. On the issue of learnability, recent work on machine learning has shown that some context-free languages *can* be learned in an unsupervised manner (Clark, 2001; Clark et al., 2008, 2010). Finally, there are theoretical results showing that learning from positive evidence alone is possible given a probabilistic reformulation of the learning problem (Hsu et al., 2011).

3.2.3 Models of morphology

For a grammatical description of morphology, there are two types of morphemic approaches that are frequently contrasted: item-and-arrangement (IA) and item-and-process (IP) (Hockett, 1954; Matthews, 1991).

In its simplest form, item-and-arrangement morphology considers the words to consists of a set of morphs and their arrangement. The arrangement is determined by morphotactic constructions that tell which items may occur together.

The first problem in this approach is that the phonological rules of the language may produce different morphs for the same morpheme, discussed in Section 3.1.7. To include allomorphy, the items of the IA model can be considered as morphemes that have each their own set of surface forms (allomorphs). Thus the words consist of sequences of arranged morphemes. The surface form of each morpheme is determined by morphophonemic rules of the language based on the context in which it occurs.

The second problem of IA, which also applies to the revised model, are inflections such as **take**–**took**. In this case, there is no suffix corresponding to the morpheme +PAST. Possible solutions include at least the following (Hockett, 1954):

1. Analyze **took** as a single morpheme TOOK.
2. Analyze **took** as a *portmanteau morph* of two morpheme sequence TAKE and +PAST.
3. Analyze **took** as one allomorphs of TAKE, and introduce a *zero* or *null morph* as one allomorph of +PAST.
4. Analyze **took** as a discontinuous allomorph /t...k/ (**t-k-**) of TAKE plus infixed allomorph /ʊ/ (**-oo-**) of +PAST.
5. Analyze **took** as **take** plus a *replacive morph* /ʊ/←/eɪ/.

Hockett (1954) considers only the fourth option acceptable. He finds the first option unacceptable as it controverts the parallelism between **took** and other past tense forms, the second option arbitrary as the similarity between **take** and **took** is no different to that between **bake** and **baked**, the third option arbitrary as one could as well define **took** as an allomorph of +PAST and the null morph as allomorph of TAKE, and the fifth option as inconsistent with that morphs should be composed of phonemic material.

As the above example demonstrates, the IA morphology is most suited to agglutinative languages. Fusion and other processes that break the one-to-one correspondence between morphs and morphemes make the description of the model more complicated. Most work on the unsupervised learning of morphology, including the publications of this thesis, is based on segmentation and thus implicitly assume an IA type of a model.

The item-and-process approach is more general than IA and can deal with both fusion and transfixation. In the IA models, all types of morphemes are of the same standing, whereas in the IP models, the first item of a word is always a free morpheme that is the root of the word. Instead of constructions, new word forms are formed by inflectional, derivational, and compounding *processes*. The processes are, in principle, operations that take one or several items (words or morphemes) as input, and return a new item (word form). The representation of the items is a list of features. For example, the representation for **bird** could be:

$$\left[\begin{array}{l} \textit{Singular} \\ \text{BIRD} \end{array}\right]_{\textit{Noun}}$$

The lexicon stores the phonemic or orthographic representations for lexemes. Substituting the orthographic[8] root for the lexeme above gives:

$$\left[\begin{array}{l} \textit{Singular} \\ \text{bird} \end{array}\right]_{\textit{Noun}}$$

An example of a derivational process for English is

$$[X]_{\textit{Verb}} \rightarrow [X + \text{er}]_{\textit{Noun}},$$

which takes a verb such as **walk** as X and returns the noun **walker**. The suffix **er** is called the *marker* of the process. Similarly, one type of compounding process can be defined by

$$[X]_{\textit{Adj}} + [Y]_{\textit{Noun}} \rightarrow [X + Y]_{\textit{Noun}}.$$

For example, $X =$ **black** and $Y =$ **bird** yields **blackbird**.

Regular inflection to plural adds **s** as a suffix:

$$\left[\begin{array}{l} \textit{Plural} \\ X \end{array}\right]_{\textit{Noun}} \rightarrow X + \text{s}$$

The processes can have arbitrary restrictions on the forms. For the phonological change in **take**–**took**, we might have

$$\left[\begin{array}{l} \textit{Past tense} \\ \text{E} \\ C_1/\text{eɪ}/C_2 \end{array}\right]_{\textit{Verb}} \rightarrow C_1/\text{ʊ}/C_2$$

[8] Since the topic of this chapter is written language, orthographic representations are applied whenever applicable.

where C_1 and C_2 are arbitrary consonants and E is the class of words that can be inflected with the process, including TAKE and SHAKE. See Matthews (1991) for a complete description and more examples.

More recently, the models of morphology have been further divided into *lexical* or *inferential* models, depending on whether morphosyntactic features are associated with morphemes or rules, and *incremental* or *realizational* models, depending on whether morphemes or rules always add information when they are applied or the forms are licensed by morphosyntactic features (Stump, 2001; Roark and Sproat, 2007). IA is an example of lexical–incremental and IP of inferential–realizational approach, and they are still considered the extremes of the categorization. Roark and Sproat (2007, Ch. 3) show that for the computational perspective, IA and IP are actually equivalent: both can be implemented by finite-state transducers. To be specific, they are different refactorizations of the same FST.

A different formalism developed for phonology and morphology is optimality theory (OT) by Prince and Smolensky (1993). In OT, the selection of the surface form of a word given its morphemes is solved as a constraint satisfaction problem. Karttunen (1998) shows that also the OT constraints can be modeled by finite-state transducers; further discussion from the computational viewpoint is again provided by Roark and Sproat (2007, Ch. 4).

3.2.4 Phrase structure grammar

The notion of constituency has played an important role in the development of the modern grammars. The most straightforward application of immediate-constituent analysis to syntax is phrase structure grammar.

Consider the IC analysis of the sentence **"The book I read yesterday was well written"** given in Section 3.1.6. It can also be represented by a tree graph such as the one in Figure 3.3. A parent node and its child nodes, such as *NP* with *Det* and *Noun* in the left, form a grammatical construction with child nodes as constituents.

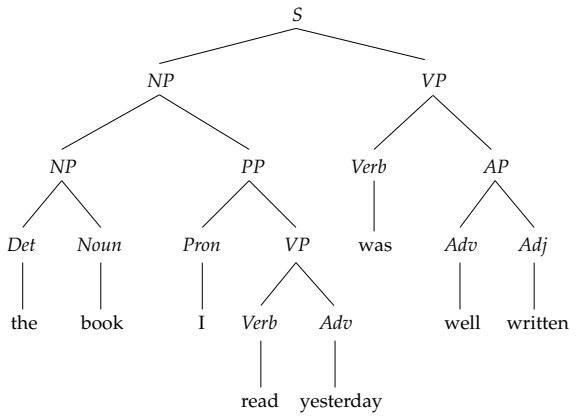


Figure 3.3. Parse tree for the sentence "The book I read yesterday was well written".

Alternatively, the relation between the three nodes can be explained by a *rewrite* rule [*NP* → *Det Noun*], meaning that the symbol *NP* may generate the pair of symbols *Det* and *Noun*. To include the generation of words, rules such as [*Noun* → book] can be included. Grammars that are made up entirely of re-

Table 3.1. A phrase structure grammar that can generate the parse tree in Figure 3.3. The vertical bar indicates optional rewrite rules for a single input symbol.

S	→	*NP VP*	*Adj*	→	written | good | ...
NP	→	*Det Noun* | *NP PP*	*Adv*	→	yesterday | well | really | ...
PP	→	*Pron VP*	*Det*	→	the | a | an
VP	→	*Verb Adv* | *Verb AP*	*Noun*	→	book | article | ...
AP	→	*Adv Adj*	*Pron*	→	I | you | she | ...
			Verb	→	read | was | took | ...

write rules of phrase and part-of-speech categories are called *phrase structure grammars*. The set of rules (grammar) needed for the analysis in Figure 3.3 are shown in Table 3.1. The symbols that occur only on the right side of the rules (i.e. words) are the *terminals* of the grammar, and all other symbols are *non-terminals*. If the left side of each rule may contain only a single non-terminal symbol, the grammar is a *context-free grammar* (CFG).

An important property of phrase structure grammars and many other grammars based on rewrite rules is *recursivity*. A rule such as [$NP \rightarrow NP\ PP$] that has the same non-terminal both in the input and output, can be applied arbitrarily many times. The same applies to any combination of two rules. For example, alternating between [$NP \rightarrow NP\ PP$] and [$PP \rightarrow Prep\ NP$] gives sequence *NP Prep NP Prep NP ... Prep NP*. A few recursions are possible even in practice: consider, for example, **[contracts for [the employees at [the marketing department of [the company]$_{NP}$]$_{NP}$]$_{NP}$]$_{NP}$**. This recursive *center-embedding* introduces long-distance dependencies between the words. For example, the decision of using **was** or **were** after the top-level *NP* (subject-verb agreement) depends not on the previous noun (**company**) but the first one (**contracts**).

Given a CFG, there are two ways to use it. First, it can be used for *deriving* new sentences. The derivation starts from the initial symbol *S* and each rule is read as "rewrite the symbol on the left with the string of symbols on the right". For example, derivation of the sentence **"an article was really good"** with the CFG in Table 3.1 is as follows:

S → *NP VP*
→ *Det Noun VP*
→ *Det Noun Verb AP*
→ *Det Noun Verb Adv Adj*
→ an article was really good

The sentences that can be derived using a CFG are referred to as grammatical, and all the other sentences are referred to as ungrammatical according to the grammar. Thus CFG defines a formal language that is the set of its grammatical sentences.

Second, CFG can be used to find a parse tree for a given grammatical sentence. This is called *syntactic parsing*. The search can be done either top-down, starting with the symbol *S* and applying such rules that can lead to the target words, or bottom-up, applying the rules backwards to the target words and the resulting symbols, trying to reach a single *S* symbol. The algorithms for automatic parsing, such as Cocke-Younger-Kasami (CYK) (Kasami, 1965; Younger, 1967), use dynamic programming techniques (see Jurafsky and Martin, 2008, Ch. 13).

CFGs can be automatically extracted from a *treebank*: a corpus, in which each sentence is manually annotated with a parse tree. Treebanks have been impor-

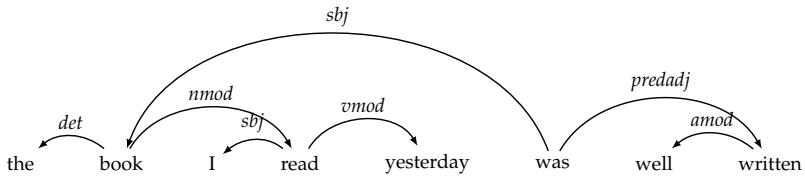


Figure 3.4. Dependency relations for the sentence "The book I read yesterday was well written". Arrows point from heads to dependents. The labels of the relations are suggestive.

tant to the development of CFGs within NLP especially because unsupervised learning of CFGs has proven to be a difficult problem. A major problem is that there are infinitely many grammars that can generate a set of sentences. While there has been progress in learning some restricted types of generative grammars (see Clark et al., 2008, 2010), they have not yet been applied to real natural language data sets.

For statistical language modeling, a relevant extension of CFGs are probabilistic context-free grammars (PCFGs) (see Jurafsky and Martin, 2008, Ch. 14). PCFGs introduce a probability for each rewrite rule $X \rightarrow \boldsymbol{Y}$ so that $\sum_{\boldsymbol{Y} \in \Sigma^*} p(X \rightarrow \boldsymbol{Y}) = 1$, where Σ is the combined set of terminals and non-terminals. PCFGs are discussed more in the context of statistical language models in Section 4.3.1.

Languages with relatively free word order are problematic for phrase structure grammars. A pair of rule such as [$S \rightarrow NP\ VP$] and [$VP \rightarrow Verb\ NP$] assume the SVO word order: the subject occurs before the verb and the object after the verb, as in **"dog chased cat"**. In Finnish, the object is determined by morphological change in the word (e.g. partitive case suffix **a** in **kissa+a**). Accordingly, the OSV order (**"kissaa koira jahtasi"** = **"cat+[PTV] dog chased"**) is equally grammatical to SVO (**"koira jahtasi kissaa"** = **"dog chased cat+[PTV]"**), although the OSV order adds emphasis on the object compared with the default SVO order.

3.2.5 Dependency grammars

Dependency grammars show an alternative way to look at the organization of sentences. Instead of studying constituency and phrase structure rules, they study syntactic *dependency relations* between words. A dependency relation considers two words: one is *dependent* and the other is *head* of the relation. The relations may either be unnamed or labeled. In the latter case, the terms from traditional sentence analysis, such as subjects, objects and predicates are applied, but again, different tagsets can be applied.

The main verb of a sentence is assumed to be the *root* of the sentence. That is, it is not dependent on anything else. Subject and objects are then dependent on the main verb, adjectives and determiners on the nouns that they modify, and so on. Figure 3.4 shows a typed dependency parse for our example sentence. In this graphical representation, the arrows point from heads to dependents; sometimes they are drawn in the opposite direction. The dependency parse can also be drawn as a tree, starting with the root in the top. Every node in the tree except for the root node corresponds to one word in the sentence.

Dependency grammars have two advantages over CFGs. First, the dependents also have a strong statistical dependence on their heads: knowing the verbs help deciding both how many subjects and objects there are and which nouns there are likely to be. Second, the grammars can easily handle languages

with free word order. Dependency grammars can be extracted from dependency treebanks, but also from CFG treebanks, or directly using a CFG or some other suitable grammar. Kübler et al. (2009) give a recent overview on dependency grammars and their parsing algorithms.

3.2.6 Context-sensitive grammars

Phrase structure grammar is restricted to context-free rules, which is a problem if the syntax of natural languages is context-sensitive. While the assumption that the natural languages are not context-free is popular—one widely accepted proof is provided by Shieber (1985)—the question is still controversial (Pullum and Gazdar, 1982; Pullum and Rawlins, 2007).

The first widely influential attempt to increase the descriptive power of phrase structure grammars was the *transformational grammar* by Chomsky (1965, 1957). Consider the following pairs of sentences (Karlsson, 2008, p. 181):

(1a) The State of California executed Caryl Chessman.

(1b) Caryl Chessman was executed by the State of California.

(2a) That the hunters shot made me sick.

(2b) The shooting of the hunters made me sick.

The sentences (1a) and (1b), as well as (2a) and (2b) have approximately the same meaning. In the first case, the sentence has been transformed from active to passive. In the second case, the subordinate clause has been transformed to a noun phrase. The basic idea in transformational grammar is to identify one core sentence from a set of related sentences, and describe the other sentences by *transformation rules* or *transformations* from the core sentence. The transformations are context-sensitive and may delete, substitute and insert the elements of a phrase structure. The language is then context-sensitive even if the core sentences were derived by a CFG.

Unification grammars extend context-free grammars by adding feature structures to the constituents. The features can, for example, specify the grammatical person and number for the word. The operation of unification tries to match the feature structures of different constituents. The possibility of failure of the unification operation imposes constraints on the derivation or parsing.

In *lexicalized grammars*, phrase structure rules are replaced by a small number of combination rules. Instead, the information on how to combine constituents are encoded in lexical categories. For example, *tree-adjoining grammars* (TAGs) have only two rules, substitution and adjunction, but the combined elements are trees instead of symbols.

Both unification grammars and lexicalized grammars have computational implementations for parsing. For details, see Roark and Sproat (2007, Ch. 9) or Jurafsky and Martin (2008, Ch. 15).

3.2.7 Construction grammars

Construction grammars is a family of cognitive theories of grammar that share a number of key ideas. Important contributions have been made by Lakoff (1987), Langacker (1987, 1991), Fillmore et al. (1988), Goldberg (1995, 2006), and Croft (2001). A short overview on construction grammars is given by Goldberg (2003).

While construction grammars are generative in the sense that they try to ac-

count for all grammatical expressions of language, they differ from Chomskyan generative grammars in many other aspects. Similar to most theories of cognitive linguistics, construction grammars posit no language-specific modules in the brain and assume that language acquisition can be explained by general cognitive mechanisms. Instead of separating lexicon and syntax (such as symbols and rules in phrase structure grammar), construction grammars describe all aspects of language by form-meaning pairs (signs), which are called constructions. All the linguistic knowledge of a user of the language is stored in her *construction lexicon* (sometimes called *constructicon*). The construction lexicon is often considered to be a network that stores the relations of the constructions (such as inheritance) in addition to their forms and meanings.

As the meanings and forms are intertwined in the constructions, they are not independent as in traditional generative grammars. The meaning component of a construction can have semantic, pragmatic, or discourse function. The form component can be anything from a morpheme to an abstract sentence construction (see Table 1.1, page 21).

The different versions of construction grammars have some variation regarding whether a construction has to be (1) complex pattern (i.e., not a morpheme), and (2) whether there has to be some aspect of the form or function of the construction that is not strictly predictable from its component parts or from other constructions recognized to exist (Schönefeld, 2006). Using terminology from Kohonen et al. (2009b), the first criterion divides the constructions into *compound* and *non-compound* constructions, and the second criterion divides the construction lexicon into *minimal* and *redundant* constructions.

There are several reasons why construction grammars are interesting both from the view of machine learning and NLP. For example, they predict that all levels of language, from morphology to discourse, can be learned by the same general learning mechanisms of cognition. No intermediate representation is assumed, but learning can be seen as direct mapping from forms to meanings. The task of grammar inference can be simply set as finding a *sufficient construction lexicon* that contains at least all the minimal constructions in the data, but can also include redundant constructions (Kohonen et al., 2009b).

By discarding pragmatics and concentrating on syntax, Chomskyan generative grammars are commonly limited to the description of frequent, central patterns of language. Given such a grammar, finding non-grammatical sentences from any large corpus is rather a rule than an exception—as Sapir (1921) put it, *"all grammars leak"*. In contrast, constructionist studies are often interested in the periphery of the language (Fried and Östman, 2004), making them relevant for anyone who wants to cover as much of the linguistic data as possible. Moreover, most of the construction grammars are usage-based theories, and thus compatible with probabilistic account of language.

However, as criticized by Bod (2009a), the current constructionist theories leave open all details for a computational implementation: How the constructions should be encoded, what kind of operations should be used to combine different constructions, and how the constructions are acquired from the observed utterances? Nevertheless, construction grammars have both inspired and are compatible with several computational models of language learning (e.g., Steels, 2004; Borensztajn et al., 2009). Within this thesis, Publication XI presents a MDL-based method for learning phrasal constructions. The publication and related work are discussed in Chapter 6.

4. Statistical language modeling

This chapter considers the problem of statistical language modeling, highlighted in the overview diagram of Figure 4.1. After a short introduction to the problem, Section 4.1 summarizes the typical applications and evaluation methods for language modeling. The next two sections give an overview on the modeling techniques: Section 4.2 on n-gram models and Section 4.3 on other types of models. The last two sections address the contributions of this thesis related to language modeling techniques. Section 4.4 considers the specific problem of building variable-order n-gram models with Kneser-Ney smoothing. New methods for this problem have been developed and evaluated in Publication I. Section 4.5 discusses the new clustering algorithm for n-gram models developed in Publication II.

Statistical language modeling refers to density estimation of the distribution $p(S)$ over some fragments of text S. The fragments are usually either words, utterances, sentences, or documents. Defining such a probability distribution is an essential part of many applications where the output of the system is text. A prototypical example is automatic speech recognition. Given acoustic speech signal $\boldsymbol{a}$, the goal of the system is to transfer it to text $\boldsymbol{s}$. Using the noisy channel model (see Section 2.4), the original message $\boldsymbol{s}$ is encoded into speech $\boldsymbol{a}$, which we try to decode. To minimize the probability of errors, such an estimate $\hat{\boldsymbol{s}}$ should be selected that

$$\hat{\boldsymbol{s}} = \arg\max_{\boldsymbol{s}} p(S = \boldsymbol{s} \mid A = \boldsymbol{a}) = \arg\max_{\boldsymbol{s}} p(S = \boldsymbol{s}) p(A = \boldsymbol{a} \mid S = \boldsymbol{s}). \tag{4.1}$$

The conditional probability $p(A \mid S)$ is estimated by an acoustic model and the probability $p(S)$ by a statistical language model. Similarly in statistical machine translation, $p(A \mid S)$ is estimated from a translation model.

Although this chapter will concentrate on generative language models that define the distribution $p(S)$, it is not the only option. Instead of using the noisy channel model, it is possible to directly estimate the conditional distribution $p(S \mid A)$—or just unnormalized scores for the candidate sentences given the input utterance. This kind of *discriminative language models* has been studied, for example, by Roark et al. (2004, 2007). The discriminative models are typically linear or log-linear and their predictions are based on features selected from the pairs $(\boldsymbol{s}, \boldsymbol{a})$. In practice, the utterance $\boldsymbol{a}$ is defined by a lattice or n-best list generated by a baseline recognizer, and the first feature is the log-likelihood given by the baseline recognizer (Roark et al., 2007). The rest of the features are based solely on the sentence $\boldsymbol{s}$. The benefit of the discriminative modeling is that arbitrary features can be applied. A supervised training algorithm (e.g. perceptron) can select only a small set of relevant features from the huge number of potential

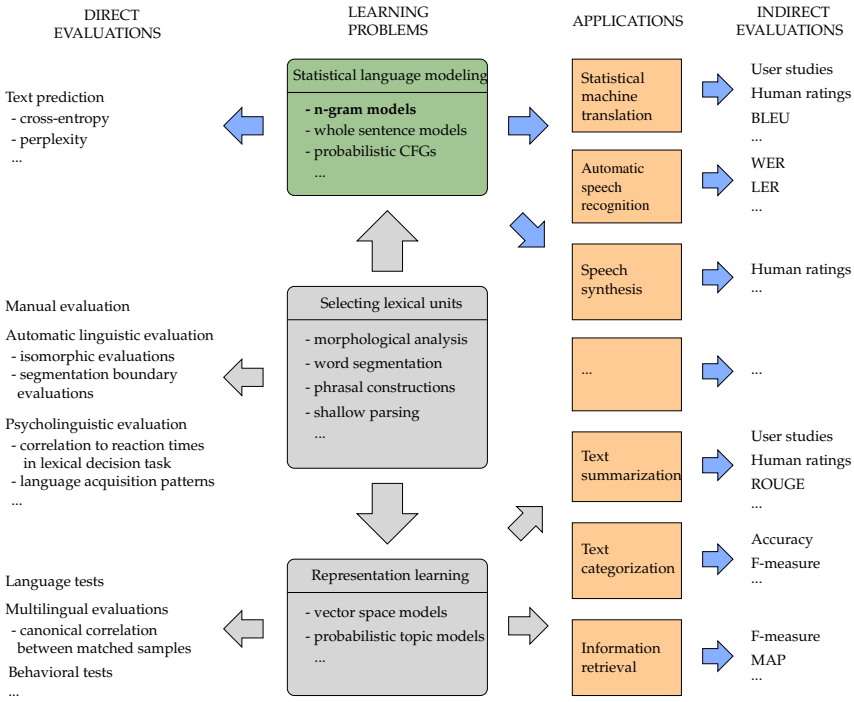


Figure 4.1. Statistical language modeling in the overview diagram.

features to select the best candidate. A drawback is that a discriminative model is dependent on the baseline recognizer (or some other application-specific decoder) and the acoustic and language models utilized by it.

Estimating the density $p(S)$ from given samples $\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_N$ is an unsupervised learning task. However, as the samples $\boldsymbol{s}$ are sequences of discrete units (e.g., characters or words) $w \in \Sigma$, the problem can be reformulated as a set of supervised learning tasks by using the chain rule of probabilities for the sequences:

$$p(\boldsymbol{s}) = \prod_{i=1}^{|\boldsymbol{s}|} p(w_i \,|\, \boldsymbol{w}_1^{i-1}), \tag{4.2}$$

where $\boldsymbol{w}_i^j$ $(i \leq j)$ is a shorthand notation for $(w_i, w_{i+1}, \ldots, w_j)$. Now, for each point i, the task is to predict the units w_i given the *history* $\boldsymbol{h} = \boldsymbol{w}_1^{i-1}$. Given training data for each history, this is a standard classification task $p(W = w_i \,|\, H = \boldsymbol{w}_1^{i-1})$. As a matter of fact, such conditional probabilities are required by the decoders of ASR or SMT systems: generating whole sentences based on the acoustic or translation model before using the language model would be a waste of computational resources. Models that cannot provide conditional probabilities are typically used only for re-ranking the N-best list or lattice of sentences generated by the decoder.

The main problem in statistical language modeling is that the language data is sparse. While it is simple enough to estimate the probability of the first unit, $p(w_1)$, for large enough i, a particular $\boldsymbol{w}_1^{i-1}$ is unlikely to be present in the training data and thus $p(w_i \,|\, \boldsymbol{w}_1^{i-1})$ cannot be estimated directly. The various types of language models differ on how they approach the problem of sparsity.

4.1 Evaluation methods and applications

Statistical language models are usually evaluated directly by calculating empirical cross-entropy or perplexity on test data (Equations 2.37 and 2.38, page 44). Goodman (2001b) shows that the decrease in cross-entropy generally reflects more accurately the potential decrease in error rate in speech recognition than perplexity. In contrast to the likelihood of the test data, entropy and perplexity values are normalized by the size of the data set. The normalization is typically based on the number of the lexical units w in the test data. However, when comparing models based on different units such as words and morphemes, the normalization factor has to be the same (Hirsimäki et al., 2006). For example, word-based cross-entropy can be defined as

$$\tilde{H}_q(\boldsymbol{D}) = -\frac{1}{W_D} \log p(\boldsymbol{D} \mid \boldsymbol{\theta}, \mathcal{M}), \tag{4.3}$$

where W_D is the number of words in $\boldsymbol{D}$, regardless of which kind of units are applied when determining $p(\boldsymbol{D} \mid \boldsymbol{\theta}, \mathcal{M})$.

The most common indirect evaluation for language models is via automatic speech recognition, which is their oldest application. The output of ASR systems is typically evaluated by word error rate (WER). WER is the Levenshtein edit distance between the reference transcription $\boldsymbol{t}$ and the output of the recognizer $\boldsymbol{r}$ normalized by the number of words in the reference. The edit distance calculates the total number of insertion, deletion and substitution operations required to convert $\boldsymbol{r}$ to $\boldsymbol{t}$ (or vice versa). Because WER does not differentiate between partially and fully equal word forms, letter error rate (LER) is sometimes used for morphologically complex languages.

Another common application is statistical machine translation. While WER can also be used to evaluate machine translation output, is not very suitable for the task. Because there is not a single correct translation, the evaluation measure should consider several reference translations whenever available, and not penalize for rephrasing the sentence. For example, edit distance between **"After you are ready, visit my office"** and **"Visit my office after you are ready"** is six (if punctuation and upper-case letters are ignored) although the semantic content is (almost) the same. One of the most popular MT evaluation scores is BLEU (Papineni et al., 2002), which counts how many n-grams (usually for $n = 1, \ldots, 4$) the proposed translation has in common with the reference translations and calculates a similarity score based on this. Although the BLEU scores may sometimes differ greatly from human evaluation (Culy and Riehemann, 2003; Callison-Burch et al., 2006), they are still considered to be good enough for comparison of similar types of machine translation systems as well as discriminative training of the translation models.

Other applications for statistical language models include optical character recognition, text correction (Kukich, 1992), and language identification (Beesley, 1988; Dunning, 1994). Some language models—unigram and topic models—are also used in IR and text categorization tasks (Section 5.2).

4.2 N-gram models

N-gram models try to solve the problem of data sparsity by making an assumption that the probability of the i^{th} unit w_i depends only on the previous $n-1$

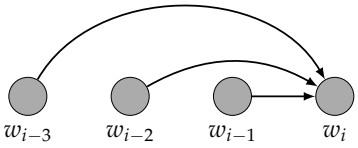


Figure 4.2. Graphical representation for a 4-gram model. Only those vertices that w_i depends on and the corresponding edges are drawn in the figure.

units. In other words, all histories that have the same $n-1$ most recent units are combined into one equivalence class:

$$p(w_i \mid \boldsymbol{w}_1^{i-1}) \approx p(w_i \mid \boldsymbol{w}_{i-n+1}^{i-1}). \tag{4.4}$$

For a relatively small n, this alleviates the data sparsity essentially. The sequences $\boldsymbol{w}_{i-n+1}^{i}$ of n units are called n-grams, or for specific values of n unigrams ($n=1$), bigrams ($n=2$), and trigrams ($n=3$). An n-gram model is also an $(n-1)^{\text{th}}$ order Markov model. Figure 4.2 illustrates a model that is based on the 4-gram assumption.

Using the product rule, the conditional n-gram probability is:

$$p(w_i \mid \boldsymbol{w}_{i-n+1}^{i-1}) = \frac{p(\boldsymbol{w}_{i-n+1}^{i})}{p(\boldsymbol{w}_{i-n+1}^{i-1})}. \tag{4.5}$$

Let $c(\boldsymbol{w}_i^j)$ be the number of occurrences of the sequence $\boldsymbol{w}_i^j$ in the training data of size N. The ML estimate for $p(w_i \mid \boldsymbol{w}_{i-n+1}^{i-1})$ is then

$$p_{\text{ML}}(w_i \mid \boldsymbol{w}_{i-n+1}^{i-1}) = \frac{c(\boldsymbol{w}_{i-n+1}^{i})/(N-n+1)}{c(\boldsymbol{w}_{i-n+1}^{i-1})/(N-n)} \approx \frac{c(\boldsymbol{w}_{i-n+1}^{i})}{c(\boldsymbol{w}_{i-n+1}^{i-1})} \tag{4.6}$$

for $N \gg n$.

The ML estimate has two problems. First, a non-zero probability cannot by assigned to any n-gram that does not exist in the training data. This applies also for any individual unit w in the set of out-of-vocabulary units $\Sigma_{\text{oov}} = \{w \in \Sigma : c(w) = 0\}$. The proportion of unseen n-grams is significant even for a large corpus because linguistic units occur according to a power-law distribution (Section 3.1.1) and the number of possible n-grams grow exponentially in n.

Second, when we estimate the ML probability for a certain rare n-gram using only a small number of occurrences, it is likely that the probability is overestimated. This can be reasoned as follows: Consider a single text corpus of size N. As noted above, all possible n-grams of the language cannot occur in the corpus. For those n-grams that have a small probability (e.g. $p = \frac{1}{2N}$), some do occur in the corpus ($p_{\text{ML}} \geq \frac{1}{N}$) and some do not ($p_{\text{ML}} = 0$). Thus the probability mass of those that do not occur in the corpus is accumulated to those that do.

There are two ways to alleviate the two problems: smoothing the maximum-likelihood distribution and combining n-gram estimators of different lengths n. The term *smoothing technique* is sometimes used to refer to the combination of the two. An extensive overview of the techniques is provided by Chen and Goodman (1999).

4.2.1 Smoothing

The basic idea of smoothing is to move probability mass from n-grams that occurred only few times to those that did not occur at all. As the probabilities of

the seen n-grams are decreased, another name for this is *discounting*.

A common way to describe smoothing methods is to define *adjusted counts* $c^*(\cdot)$ to use for the numerator of the ML estimate:

$$p_{\text{smoothed}}(w_i \mid \boldsymbol{w}_{i-n+1}^{i-1}) = \frac{c^*(\boldsymbol{w}_{i-n+1}^{i})}{c(\boldsymbol{w}_{i-n+1}^{i-1})} \tag{4.7}$$

Another way is to determine a discount $d(\boldsymbol{w}_{i-n+1}^{i}) = c^*(\boldsymbol{w}_{i-n+1}^{i})/c(\boldsymbol{w}_{i-n+1}^{i})$ so that $p_{\text{smoothed}}(w_i \mid \boldsymbol{w}_{i-n+1}^{i-1}) = d(\boldsymbol{w}_{i-n+1}^{i}) \times p_{\text{ML}}(w_i \mid \boldsymbol{w}_{i-n+1}^{i-1})$.

Let $\boldsymbol{h} = \boldsymbol{w}_{i-n+1}^{i-1}$. If $c^*(\boldsymbol{h}w) = 0$ for any $c(\boldsymbol{h}w) = 0$, the probability mass left over after smoothing is

$$\gamma(\boldsymbol{h}) = \frac{\sum_{w:c(\boldsymbol{h}w)>0} \big(c(\boldsymbol{h}w) - c^*(\boldsymbol{h}w)\big)}{c(\boldsymbol{h})} = \frac{\sum_{w:c(\boldsymbol{h}w)>0} c(\boldsymbol{h}w)\big(1 - d(\boldsymbol{h}w)\big)}{c(\boldsymbol{h})}. \tag{4.8}$$

This *missing probability mass* can be evenly distributed for the unseen units $\{w \in \Sigma : c(\boldsymbol{h}w) = 0\}$ so that $\sum_{w\in\Sigma} p_{\text{smoothed}}(w \mid \boldsymbol{h}) = 1$.

One of the oldest smoothing methods is Laplace's law, also referred to as *adding one*. It makes the assumption that every seen or unseen event is already seen once. Thus the smoothed n-gram probability is

$$p_{\text{add-1}}(w_i \mid \boldsymbol{w}_{i-n+1}^{i-1}) = \frac{c(\boldsymbol{w}_{i-n+1}^{i}) + 1}{c(\boldsymbol{w}_{i-n+1}^{i-1}) + V}, \tag{4.9}$$

where $V = |\Sigma|$ is the size of the vocabulary. Manning and Schütze (1999, p. 203) demonstrate that Laplace's law reduces the probability of frequent n-grams too much. An extension is *additive smoothing* (or Lidstone's law), in which a constant $0 < \lambda \leq 1$ is added to the frequencies:

$$p_{\text{add}}(w_i \mid w_{i-n+1}^{i-1}) = \frac{c(\boldsymbol{w}_{i-n+1}^{i}) + \lambda}{c(\boldsymbol{w}_{i-n+1}^{i-1}) + \lambda V}. \tag{4.10}$$

Witten-Bell smoothing (Witten and Bell, 1991) uses one discount for each history $\boldsymbol{h}$. Then $d(\boldsymbol{h}) = 1 - \gamma(\boldsymbol{h})$. The idea is to find such $\gamma(\boldsymbol{h})$ that it estimates the probability that a previously unseen unit w occurs after $\boldsymbol{h}$. There are $t_{1+}(\boldsymbol{h}\bullet) = |\{w \in \Sigma : c(\boldsymbol{h}w) \geq 1\}|$ unique unit types that follow $\boldsymbol{h}$ in the training data. The probability of a new type is approximated by normalizing $t_{1+}(\boldsymbol{h}\bullet)$ by the sum of types $t_{1+}(\boldsymbol{h}\bullet)$ and tokens $c(\boldsymbol{h})$. Accordingly, the discount is set to

$$d_{\text{WB}}(\boldsymbol{h}) = 1 - \frac{t_{1+}(\boldsymbol{h}\bullet)}{t_{1+}(\boldsymbol{h}\bullet) + c(\boldsymbol{h})} = \frac{c(\boldsymbol{h})}{t_{1+}(\boldsymbol{h}\bullet) + c(\boldsymbol{h})}. \tag{4.11}$$

Good-Turing (GT) estimate, proposed by Good (1953) for estimating population frequencies of species for a random sample, replaces the observed frequencies r with modified frequencies R:

$$R_n(r) = (r+1)\frac{N_{n,r+1}}{N_{n,r}}. \tag{4.12}$$

$N_{n,r} = |\{\boldsymbol{w} \in \Sigma^n : c(\boldsymbol{w}) = r\}|$ is the number of n-grams that occurred r times (i.e., frequency of frequency r). The Good-Turing formula is based on the assumption that the frequencies of the sample are binomially distributed. For non-zero counts, the adjusted count is simply

$$c^*(\boldsymbol{w}_{i-n+1}^{i}) := R_n(c(\boldsymbol{w}_{i-n+1}^{i})) \tag{4.13}$$

and the missing mass for items which did not occur in the training data is $\gamma(\boldsymbol{w}_{i-n+1}^{i-1}) = N_{n,1}/N$. In practice, $R_n(r)$ cannot be applied to large counts. Given the power-law distributions, $N_{n,r}$ is sparse for large frequencies r, and thus $N_{n,r+1}$ may easily be zero. This can be circumvented by using curve fitting to estimate the values of $N_{n,r}$ (Gale and Sampson, 1995), but as the MLE is anyway reliable for large frequencies, it is simpler to use smoothing only if $r < k$ for some constant k.

Absolute discounting, first proposed by Ney et al. (1994), subtracts a fixed discount $D > 0$ from the counts that are larger than D:

$$c^*(\boldsymbol{w}_{i-n+1}^{i}) := c(\boldsymbol{w}_{i-n+1}^{i}) - D \quad \text{if } c(\boldsymbol{w}_{i-n+1}^{i}) > D. \tag{4.14}$$

The missing probability mass for history $\boldsymbol{h}$ is then

$$\gamma(\boldsymbol{h}) = \frac{|\{w \in \Sigma : c(\boldsymbol{h}w) > D\}| \times D}{c(\boldsymbol{h})}. \tag{4.15}$$

Absolute discounting may be extended so that the discount depends on the original count $r = c(w\boldsymbol{h})$ and the n-gram order. A common setup proposed by Chen and Goodman (1999) is to have separate discounts for $r = 1$, $r = 2$, and $r \geq 3$:

$$c^*(\boldsymbol{w}_{i-n+1}^{i}) := \begin{cases} c(\boldsymbol{w}_{i-n+1}^{i}) - D_1 & \text{if } c(\boldsymbol{w}_{i-n+1}^{i}) = 1 \\ c(\boldsymbol{w}_{i-n+1}^{i}) - D_2 & \text{if } c(\boldsymbol{w}_{i-n+1}^{i}) = 2 \\ c(\boldsymbol{w}_{i-n+1}^{i}) - D_{3+} & \text{if } c(\boldsymbol{w}_{i-n+1}^{i}) \geq 3 \end{cases} \tag{4.16}$$

and

$$\gamma(\boldsymbol{h}) = \frac{t_1(\boldsymbol{h}\bullet) \times D_1 + t_2(\boldsymbol{h}\bullet) \times D_2 + t_{3+}(\boldsymbol{h}\bullet) \times D_{3+}}{c(\boldsymbol{h})}, \tag{4.17}$$

where

$$t_r(\boldsymbol{h}\bullet) = |\{w \in \Sigma : c(\boldsymbol{h}w) = r\}| \tag{4.18}$$

$$t_{r+}(\boldsymbol{h}\bullet) = |\{w \in \Sigma : c(\boldsymbol{h}w) \geq r\}| \tag{4.19}$$

give how many types of n-grams of a certain frequency follow the given history. The optimal discounts can be estimated analytically from the training data via deleted estimation (see, e.g., Chen and Goodman, 1999) or numerically from the development data.

4.2.2 Back-off and interpolation

Consider the smoothed n-gram distribution $p_{\text{smoothed}}(w \mid \boldsymbol{w}_{i-n+1}^{i-1})$. Using the missing probability mass $\gamma(\boldsymbol{w}_{i-n+1}^{i-1})$ for giving equal probabilities to all unseen w is sensible only if there is no better information. If $n > 1$, it makes sense to use a smoothed *lower-order distribution* $p_{\text{smoothed}}(w \mid \boldsymbol{w}_{i-n+2}^{i-1})$ instead.

There are two basic ways to combine the distributions. In *back-off* models, a lower order estimator is applied only if the current order cannot provide an estimate. If $\boldsymbol{h} = \boldsymbol{w}_{i-n+1}^{i-1}$ with $n > 1$, let us denote a backed-off history $\boldsymbol{w}_{i-n+2}^{i-1}$ by $\bar{\boldsymbol{h}}$. That is, $\boldsymbol{h} = w\bar{\boldsymbol{h}}$ for some $w \in \Sigma$. For unigram distributions, $\bar{\boldsymbol{h}}$ may also be an empty sequence. A recursive formulation for a back-off model is:

$$p_{\text{bo}}(w \mid \boldsymbol{h}) = \begin{cases} \frac{c^*(\boldsymbol{h}w)}{c(\boldsymbol{h})} & \text{if } c(\boldsymbol{h}w) > 0 \\ \gamma(\boldsymbol{h}) \times p_{\text{bo}}(w \mid \bar{\boldsymbol{h}}) & \text{otherwise.} \end{cases} \tag{4.20}$$

In contrast, *interpolated* models use linear interpolation of all estimators of different order:

$$p_{\text{int}}(w \mid \boldsymbol{h}) = \frac{c^*(\boldsymbol{h}w)}{c(\boldsymbol{h})} + \gamma(\boldsymbol{h}) \times p_{\text{int}}(w \mid \bar{\boldsymbol{h}}). \tag{4.21}$$

Sometimes the back-off or interpolation include a *zero-gram* distribution, which is a uniform distribution over the known units: $p_0(w) = 1/V$.

In practice, the models are usually stored in the back-off format, as they are more efficient to use. Any interpolated model can be represented as a back-off model (Jurafsky and Martin, 2008), but the converse is not always possible.

In addition to combining n-gram estimators of different order, linear interpolation is commonly used to combine different types of models. Given two models p_{a} and p_{b} and interpolation weight $0 < \lambda < 1$, the interpolated probability is:

$$p_{\text{combined}}(w \mid \boldsymbol{h}) = \lambda \times p_{\text{a}}(w \mid \boldsymbol{h}) + (1 - \lambda) \times p_{\text{b}}(w \mid \boldsymbol{h}). \tag{4.22}$$

The interpolation weights can be estimated using the EM algorithm or selected by maximizing likelihood of some held-out data. A more flexible way of combining models is *general linear interpolation*, in which the interpolation weights depend on the history $\boldsymbol{h}$ (Manning and Schütze, 1999). However, this drastically increases the number of free parameters.

4.2.3 Kneser-Ney smoothing

Back-off and interpolation, as defined above, make the assumption that the probability distribution $p_{\text{smoothed}}(w \mid \bar{\boldsymbol{h}})$ is independent of $p_{\text{smoothed}}(w \mid \boldsymbol{h})$. In some cases, this is clearly wrong. Consider, for example, the bigram **"San Francisco"**. The probability $p_{\text{smoothed}}(\textbf{Francisco} \mid \textbf{San})$ is evidently quite high: estimated from the Google n-gram corpus (Brants and Franz, 2006), the ML estimate $p_{\text{ML}}(\textbf{Francisco} \mid \textbf{San})$ is 1.2×10^{-3}. The unigram probability $p_{\text{ML}}(\textbf{Francisco}) = 4.2 \times 10^{-5}$ is reasonable if the previous word is not known. However, if we know that $w_{i-1} \neq \textbf{San}$, then it is overestimated because it includes the small but significant part of the occurrences where **Francisco** is preceded by **San**.

Kneser-Ney (KN) smoothing improves the estimates of $(n-1)^{\text{th}}$ order distributions by using *type counts* instead of token counts. Kneser and Ney (1995) derived the type-based formulas by constraining the marginal distributions of the model probabilities to their empirical estimates:

$$\sum_v p_{\text{KN}}(v\boldsymbol{h}w) = p_{\text{ML}}(\boldsymbol{h}w) = \frac{c(\boldsymbol{h}w)}{N}. \tag{4.23}$$

In their derivation, Kneser and Ney (1995) used a back-off model and required some approximations. A simpler derivation was shown by Chen and Goodman (1999). They used an interpolated model with absolute discounting:

$$p_{\text{KN}}(w \mid v\boldsymbol{h}) = \frac{\max(c(v\boldsymbol{h}w) - D, 0)}{c(v\boldsymbol{h})} + \gamma(v\boldsymbol{h}) \times p_{\text{KN}}(w \mid \boldsymbol{h}). \tag{4.24}$$

In both cases, Equation 4.23 can be solved for

$$p_{\text{KN}}(w \mid \boldsymbol{h}) = \frac{t(\bullet\boldsymbol{h}w)}{t(\bullet\boldsymbol{h}\bullet)}, \tag{4.25}$$

where $t(\bullet\boldsymbol{h}w) = |\{v \in \Sigma : c(v\boldsymbol{h}w) \geq 1\}|$ gives the number of distinct left contexts in which $\boldsymbol{h}w$ occurs and $t(\bullet\boldsymbol{h}\bullet) = \sum_{w \in \Sigma} t(\bullet\boldsymbol{h}w)$.

The constraint on the marginal distribution of an n-gram model has a theoretical motivation: Goodman (2001b) shows that any smoothing method that does not preserve the marginal distributions cannot be optimal. Moreover, Goodman (2004) notes that Kneser-Ney smoothing is an approximation to a maximum-entropy model with an exponential prior. In fact, maximum-entropy language models, discussed later in Section 4.3.2, satisfy exactly all known marginal constraints.

Informally, the approach can be motivated by the view that only such a unit that occurs in many different contexts is likely to occur in a new context. In the example above, $p_{\mathrm{KN}}(\textbf{Francisco}) = 7.1 \times 10^{-6}$, which is one order of magnitude smaller than the token-based estimate.

While no longer motivated by the marginal constraints, the type-based distributions are applied also for the distributions of order $k < n-1$. Let $t^*(\cdot)$ be the type count $t(\cdot)$ adjusted by smoothing analogously to $c(\cdot)$ and $c^*(\cdot)$. For interpolated Kneser-Ney smoothed model of order n,

$$p_{\mathrm{KN}}(w \mid \boldsymbol{h}) = \begin{cases} p_{\mathrm{KN}}(w \mid \bar{\boldsymbol{h}}) & \text{if } |\boldsymbol{h}w| > n. \\ \frac{c^*(\boldsymbol{h}w)}{c(\boldsymbol{h})} + \gamma(\boldsymbol{h}) \times p_{\mathrm{KN}}(w \mid \bar{\boldsymbol{h}}) & \text{if } |\boldsymbol{h}w| = n \\ \frac{t^*(\bullet \boldsymbol{h}w)}{t(\bullet \boldsymbol{h}\bullet)} + \gamma(\boldsymbol{h}) \times p_{\mathrm{KN}}(w \mid \bar{\boldsymbol{h}}) & \text{if } |\boldsymbol{h}w| < n. \end{cases} \tag{4.26}$$

Modified Kneser-Ney interpolation by Chen and Goodman (1999) is often considered as the state-of-the-art smoothing method. It uses Kneser-Ney smoothing, interpolation, and absolute discounting with three discounts (Equations 4.16–4.17).

4.2.3.1 Relation to Pitman-Yor process and power-law discounting

Goldwater et al. (2006, 2011) and Teh (2006) have noticed a correspondence between interpolated Kneser-Ney models and the Pitman-Yor process used in non-parametric Bayesian models (Section 2.7.4, page 61). According to Teh (2006), a Kneser-Ney model can be interpreted as a particular type of approximate inference for their hierarchical Pitman-Yor (HPY) language model. In HPY language models, several Pitman-Yor processes (Equation 2.71, page 62) are combined so that the base distribution for each $p(w \mid \boldsymbol{h})$ is $p(w \mid \bar{\boldsymbol{h}})$:

$$G_{\boldsymbol{h}}(w) \sim \mathrm{PYP}(G_{\bar{\boldsymbol{h}}}(w), d_{|\boldsymbol{h}|}, \theta_{|\boldsymbol{h}|}) \tag{4.27}$$

The base distribution for the empty history is the uniform distribution over the vocabulary: $G_0(w) = 1/V$.

Teh (2006) reports only small and statistically insignificant differences in perplexity for the HPY language models and the interpolated Kneser-Ney models. Huang and Renals (2010a) use a parallel training algorithm in order to use larger amounts of training data, and test the models in a conversational speech recognition task. They obtain small but statistically significant improvements for the HPY language model over n-gram models with modified KN interpolation, both for perplexity and word error rate. However, both Teh (2006) and Huang and Renals (2010a) compare only 3-gram models.

Using the relation to Pitman-Yor process language models, Huang and Renals (2010b) introduce a *power-law discounting*, which is similar to modified Kneser-Ney interpolation, but introduces strength parameter θ and multiplies the absolute discount by a power-law term $c(\boldsymbol{h}, w)^D$. This term approximates $t_w(z_1^i)$,

the number of tables labeled with word w, in Equation 2.71. The power law smoothing is defined by:

$$p_{\text{pow}}(w \mid \boldsymbol{h}) = \frac{\max(c(\boldsymbol{h}w) - Dc(\boldsymbol{h}w)^D, 0)}{\theta + c(\boldsymbol{h})} + \gamma(\boldsymbol{h}) \times p_{\text{pow}}(w \mid \bar{\boldsymbol{h}}); \tag{4.28}$$

$$\gamma(\boldsymbol{h}) = \frac{\theta + D\sum_{v \in \Sigma} c(\boldsymbol{h}v)^D}{\theta + c(\boldsymbol{h})}. \tag{4.29}$$

The lower-order distributions are then set to

$$p_{\text{pow}}(w \mid \bar{\boldsymbol{h}}) = \frac{\sum_{v \in \Sigma} c(v\bar{\boldsymbol{h}}w)^D}{\sum_{w' \in \Sigma}\sum_{v \in \Sigma} c(v\bar{\boldsymbol{h}}w')^D}, \tag{4.30}$$

which again satisfies the marginal constraint. Evidently, the result of Equation 4.30 approaches the type-based estimate when $D \to 0$.

In their experiments, Huang and Renals (2010b) use 3-gram models, $\theta = 0$, and $D = n_1/(n_1 + 2n_2)$, where n_1 and n_2 are the total numbers of n-grams with exactly one or two counts (deleted estimation). The perplexities of the power-law discounted models outperform modified Kneser-Ney interpolation and have no statistically significant differences to perplexities of the Pitman-Yor process language models while being computationally more efficient.

4.2.4 Variable length n-grams

If all observed n-grams are included in the model, the size of the model grows quickly with respect to the model order n and available training data. As limiting either of those often reduces the accuracy of the model, it is common to restrict the number of n-grams included in the model based on some other criteria.

An n-gram model can be stored as a tree where the node for the sequence w_i^j is a child for w_i^{j-1} and the parent–child edge is labeled with the identity of w_j (Kneser, 1996). A *leaf* n-gram is not a prefix of any $(n+1)$-gram in the model.[1] In a full model of order M, all leaf n-grams are at the depth M. If some of the n-grams are excluded from the model, the maximal history length may vary in different contexts. For example, removing $v\boldsymbol{h}w$ for all $v \in \Sigma$ from a $(|\boldsymbol{h}|+2)$-gram model means that the model order is decreased to $(|\boldsymbol{h}|+1)$ in the context $\boldsymbol{h}$. These kinds of models are called *variable length n-gram* or *varigram* models. Figure 4.3 illustrates a varigram model that uses a maximum n-gram length of three.

For an n-gram history $\boldsymbol{h}$, let $\mathrm{A}_{\boldsymbol{h}\bullet} = \{w \in \Sigma : c(\boldsymbol{h}w) > 0\}$ be the set of all observed n-grams, $\mathrm{I}_{\boldsymbol{h}\bullet} \subseteq \mathrm{A}_{\boldsymbol{h}\bullet}$ those that are included in the model, and $\mathrm{E}_{\boldsymbol{h}\bullet} = \mathrm{A}_{\boldsymbol{h}\bullet} \setminus \mathrm{I}_{\boldsymbol{h}\bullet}$ those that are excluded (pruned). All n-grams longer than n can be considered as pruned n-grams. A varigram model that uses adjusted counts $c^*(\cdot)$ and linear interpolation can be defined by

$$p_{\text{pruned}}(w \mid \boldsymbol{h}) = \begin{cases} \frac{c^*(\boldsymbol{h}w)}{c(\boldsymbol{h})} & + \quad \gamma(\boldsymbol{h}) \times p_{\text{pruned}}(w \mid \bar{\boldsymbol{h}}) \quad \text{if } \boldsymbol{h}w \in \mathrm{I}_{\boldsymbol{h}\bullet} \\ 0 & + \quad \gamma(\boldsymbol{h}) \times p_{\text{pruned}}(w \mid \bar{\boldsymbol{h}}) \quad \text{if } \boldsymbol{h}w \in \mathrm{E}_{\boldsymbol{h}\bullet} \end{cases}. \tag{4.31}$$

[1] It is also possible to build the tree in such manner that each parent is the suffix of its children, not prefix. However, prefix trees (*tries*) are more convenient as for each n-gram $\boldsymbol{h}w$, also $\gamma(\boldsymbol{h})$ has to be stored.

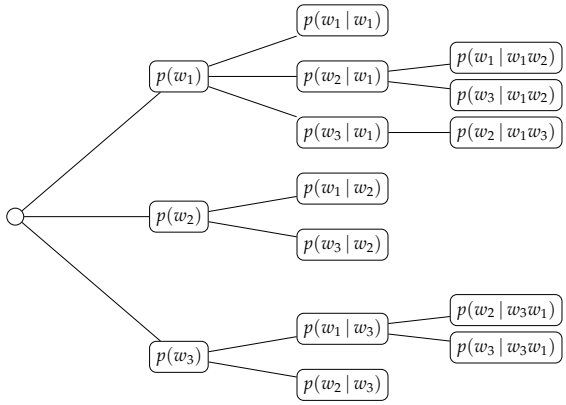


Figure 4.3. Example of a variable length n-gram model arranged in a tree structure.

The interpolation weights $\gamma(\boldsymbol{h})$ are modified to include the probability mass from the removed n-grams:

$$\gamma(\boldsymbol{h}) = \frac{\sum_{\boldsymbol{g} \in \mathrm{I}_{\boldsymbol{h}\bullet}} (c(\boldsymbol{g}) - c^*(\boldsymbol{g})) + \sum_{\boldsymbol{g} \in \mathrm{E}_{\boldsymbol{h}\bullet}} c(\boldsymbol{g})}{c(\boldsymbol{h})}. \tag{4.32}$$

The need to store the interpolation or back-off weights γ limits which n-grams can be pruned. If for any $v \in \Sigma$, n-gram $\boldsymbol{h}wv$ is in the model, then pruning $\boldsymbol{h}w$ is not likely to be useful, as $\gamma(\boldsymbol{h}w)$ has to be stored in any case. Moreover, if the n-grams are stored in the tree structure similar to that of Figure 4.3, $\boldsymbol{h}w$ will automatically have a node in the tree.

There are two basic approaches for selecting the n-grams of the model: *Pruning* refers to starting with a full model that contains all observed n-grams up to a given order, and then removing some according to a pruning criterion. *Growing* refers to starting with a unigram model and adding new n-grams to it, usually one order at a time, again with a suitable criterion. Growing has two advantages over pruning: there is no need to store the full model, which may be significantly larger than the final model, and the model order does not have to be decided beforehand. However, a growing algorithm may never even consider adding some n-grams of the full model that might have been useful. For example, if the tree in Figure 4.3 was grown one order at a time, and $p(w_1 \mid w_3)$ was not added, its children $p(w_2 \mid w_3 w_1)$ and $p(w_3 \mid w_3 w_1)$ would not be considered.

4.2.4.1 Pruning algorithms

The simplest way to prune n-grams is to use *count cutoffs*: all n-grams that occur fewer than T times in the training data are excluded from the model. Formally $\mathrm{I}_{\boldsymbol{h}\bullet} = \{\boldsymbol{g} \in \mathrm{A}_{\boldsymbol{h}\bullet} : c(\boldsymbol{g}) \geq T\}$. Because $c(\boldsymbol{h}) \geq c(\boldsymbol{h}w)$ and $c(\boldsymbol{h}) \geq c(w\boldsymbol{h})$ for any $w \in \Sigma$, cutoffs remove complete subtrees from the model. According to Goodman and Gao (2000), low cutoff thresholds T do not increase cross-entropy of the model significantly. However, if a low T is not enough for the desired model size, more sophisticated methods give better results.

While cutoff pruning is a simple heuristic, the more sophisticated pruning methods try to estimate the difference between the original model (p) and the pruned model (p_{pruned}) when considering whether to prune a single n-gram $\boldsymbol{h}w$. That is, if $p_{\text{pruned}}(w \mid \boldsymbol{h})$ is effectively the same as $p(w \mid \boldsymbol{h})$, little information is lost by pruning $\boldsymbol{h}w$.

One early example of such an pruning method is the *weighted difference* method by Seymore and Rosenfeld (1996). The n-grams $\boldsymbol{h}w$ are ranked according to the criterion

$$d_{\mathrm{WD}}(\boldsymbol{h}w) := c^*(\boldsymbol{h}w) \times \log \frac{p(w \mid \boldsymbol{h})}{p_{\mathrm{pruned}}(w \mid \boldsymbol{h})}, \tag{4.33}$$

where $c^*(\boldsymbol{h}w)$ is the adjusted count of the n-gram and

$$p_{\mathrm{pruned}}(w \mid \boldsymbol{h}) = \gamma(\boldsymbol{h}) p(w \mid \bar{\boldsymbol{h}}) \tag{4.34}$$

is the probability estimate from a model without the n-gram $\boldsymbol{h}w$. After ranking, n-grams are removed in increasing order of d_{WD} until the desired model size is reached. Seymore and Rosenfeld (1996) use the weighted difference pruning for Good-Turing smoothed back-off models.

Kneser (1996) describes a pruning method for Kneser-Ney smoothed models stored in a prefix tree. In this *Kneser pruning*, the cost of pruning a leaf n-gram is

$$d_{\mathrm{KP}}(\boldsymbol{h}w) := p(\boldsymbol{h}w) \times \log \frac{p(w \mid \boldsymbol{h})}{p_{\mathrm{pruned}}(w \mid \boldsymbol{h})}. \tag{4.35}$$

If a non-leaf n-gram $\boldsymbol{h}w$ is pruned, the complete subtree is pruned (i.e., all n-grams that have $\boldsymbol{h}w$ as a prefix are also pruned). For non-leaf n-grams, the cost is the average of $d_{\mathrm{KP}}(\boldsymbol{g})$ over all the n-grams $\boldsymbol{g}$ that have $\boldsymbol{h}w$ as prefix, including the n-gram itself. The costs are first calculated for every n-gram and then the n-grams with the lowest costs are pruned until the desired number of n-grams is reached. Kneser (1996) also defines a back-off distribution that approximately keeps the marginal constraints in the Kneser-Ney smoothing. The relation of Kneser-Ney smoothing and varigram models is considered in more detail in Section 4.4 and Publication I.

The logarithmic ratios in Equations 4.33 and 4.35 resemble that of the relative entropy (Equation 2.35, page 43), but consider only the effect of a single probability $p(W = w \mid \boldsymbol{h})$ instead of the whole distribution $p(W \mid \boldsymbol{h})$. The *entropy-based pruning* by Stolcke (1998) uses a pruning criterion based directly on relative entropy:

$$d_{\mathrm{RE}}(\boldsymbol{h}w) := D(p \,||\, p_{\mathrm{pruned}}) = \sum_{v \in \Sigma} p(\boldsymbol{h}v) \times \log \frac{p(v \mid \boldsymbol{h})}{p_{\mathrm{pruned}}(v \mid \boldsymbol{h})} \tag{4.36}$$

All n-grams that have a lower relative entropy than a threshold ϵ are pruned. Similar to the weighted difference pruning and Kneser pruning, interactions between different n-grams are not considered. Apart from using a theoretically sound measure, an advantage of entropy-based pruning is that it does not need the original counts from the data. Thus it can be used to prune any existing model.

In contrast to the above methods, Bonafonte and Mariño (1996) proposes pruning whole histories instead of individual n-grams. When a history $\boldsymbol{h}$ is pruned, all n-grams $\{\boldsymbol{h}w : w \in \Sigma \wedge c(\boldsymbol{h}w) > 0\}$ are removed and the back-off history $\bar{\boldsymbol{h}}$ is used for prediction. As pruning criterion, they use both count cutoff for $c(\boldsymbol{h})$ and threshold for the relative entropy between the distributions $p(w \mid \boldsymbol{h})$ and $p(w \mid \bar{\boldsymbol{h}})$.

4.2.4.2 Algorithms for growing n-grams

Various algorithms for growing varigram models (e.g. Ristad and Thomas, 1995; Niesler and Woodland, 1996b; Siu and Ostendorf, 2000; Siivola and Pellom, 2005) can be compared at least in three aspects.

The first aspect is whether the model is grown by adding individual n-grams $\boldsymbol{h}w$ or full distributions $\{\boldsymbol{h}w : w \in \Sigma \wedge c(\boldsymbol{h}w) > 0\}$. The former is used by Ristad and Thomas (1995) and Siivola and Pellom (2005), the latter by Niesler and Woodland (1996b, 1999) and Siu and Ostendorf (2000). In the latter case, it is naturally possible to reduce the model further by pruning individual n-grams after the growing phase.

The second aspect is whether the n-grams $\boldsymbol{g}$ are grown forward to $\boldsymbol{g}w$ or backward to $w\boldsymbol{g}$. If the n-grams are stored in a prefix tree such as the one in Figure 4.3, it is natural to grow forward to avoid adding extra n-grams to the tree. This approach is taken by Siivola and Pellom (2005). Niesler and Woodland (1996b, 1999) and Siu and Ostendorf (2000) use an alternative tree structure, in which each node represents a *history* of the model and stores the conditional distribution $p(w \mid \boldsymbol{h})$ as well as $\gamma(\boldsymbol{h})$. The root of the tree represents empty history and stores the unigram distribution. If the parent of $\boldsymbol{h}$ is its suffix $\bar{\boldsymbol{h}}$, the path from a leaf node to the root node encompasses all the nodes that are needed to calculate the probability of a given n-gram.

Finally, as in pruning, a criterion that determines which n-grams or histories to include and which to exclude is required. Including everything that increased the log-likelihood of the training data would result in overlearning. In principle, any of the general model selection methods described in Section 2.6 can be applied. Ristad and Thomas (1995) use two-part MDL and observe the change in the sum of the description length of the model and description length data given the model. A new n-gram is added only if the total code length decreases. Also Siivola and Pellom (2005) use a MDL-based criterion, but with a scheme that follows the practical encoding of the model instead of tighter combinatorial coding such as the one used by Ristad and Thomas (1995). Niesler and Woodland (1996b, 1999) derive a criterion based on the log-likelihoods in leave-one-out cross-validation. Siu and Ostendorf (2000) compare several criteria, including relative entropy, log-likelihood difference, and MDL, and settle on using log-likelihood difference with leave-one-out smoothed estimates.

4.2.5 Cluster n-grams

Making the n-gram assumption is the basic approach for collapsing the histories into equivalence classes. However, it is sometimes combined with other approaches. One popular idea is to cluster the units and make the n-gram predictions on the clusters. For example, for a group of words that have paradigmatic relations, each individual word can be replaced by the index of the group. Trivial examples of groups are similar types of numbers (integers, decimal numbers, dates) and groups of names (days of the week, month names, names of people).

The first cluster n-gram model, proposed by Brown et al. (1992), assumed that the probability of the current unit w_i depends only on its cluster c_i, and c_i depends only on the previous clusters:

$$p_{\text{Brown}}(w_i \mid \boldsymbol{w}_{i-n+1}^{i-1}) = p(w_i \mid c_i) \times p(c_i \mid \boldsymbol{c}_{i-n+1}^{i-1}). \tag{4.37}$$

A graphical model of the Brown clustering for $n = 4$ is shown in Figure 4.4.

Let $\pi : \Sigma \mapsto \Pi$ be a function that maps the words in Σ to the clusters in Π. For selecting π, a direct way is to maximize the likelihood of the data for a simple enough model using a local, greedy algorithm. For example, Brown et al. (1992) derived that maximizing the average mutual information of adjacent classes also maximizes the likelihood for a bigram model, and used agglomerative cluster-

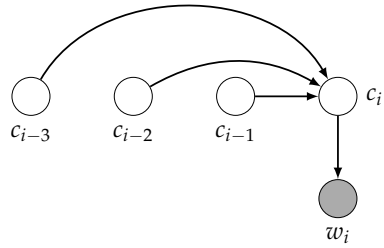


Figure 4.4. Graphical representation for a cluster 4-gram model by Brown et al. (1992). Only the variables that have a direct effect on w_i or c_i are shown.

ing. Niesler et al. (1998) found that a similar approach outperformed clustering based on part-of-speech tags. Gao et al. (2002) were able to get improvements by using different clusters for the same word when (1) the word is in the history and (2) the word is to be predicted.

The predictions from a cluster n-gram model are usually combined with those from a standard n-gram model. Goodman (2001b) studied various other ways of using word clusters. For example, the current word may depend also on the previous clusters $\boldsymbol{c}_{i-n+1}^{i-1}$ or the current cluster on previous words. The best performing technique for trigram models with large training corpus was the "fullibmpredict" model:

$$p_{\text{fullibmpredict}}(w_i \mid \boldsymbol{w}_{i-n+1}^{i-1}) = \big[\lambda p(c_i \mid \boldsymbol{w}_{i-n+1}^{i-1}) + (1-\lambda) p(c_i \mid \boldsymbol{c}_{i-n+1}^{i-1})\big] \times \big[\mu p(w_i \mid \boldsymbol{w}_{i-n+1}^{i-1}, c_i) + (1-\mu) p(w_i \mid \boldsymbol{c}_{i-n+1}^{i})\big]. \quad (4.38)$$

First, the current cluster is predicted using interpolation of two models, and then the current word is predicted, again interpolating two different models.

Publication II of this thesis introduces another type of n-gram model based on clustering. In this model, presented later in Section 4.5, the histories $\boldsymbol{h}$ are clustered instead of single units w.

4.2.6 Back-off graph and skipping

In standard back-off or interpolation models, the probability $p(w_i \mid \boldsymbol{w}_{i-n+1}^{i-1})$ is estimated by combining n-gram estimators that use gradually shorter histories. The sequence

$$\boldsymbol{w}_{i-n+1}^{i-1} \to \boldsymbol{w}_{i-n+2}^{i-1} \to \ldots \to \boldsymbol{w}_{i-2}^{i-1} \to w_{i-1} \to \epsilon,$$

can be called *back-off path* of the model, where ϵ denotes empty history (unigram model). Evidently, there are many other back-off paths that could be used. For example, **"write her a letter"** should be useful for estimating the probability $p(\textbf{letter} \mid \textbf{write Mary a})$, but standard back-off cannot use $w_{i-3} =$ **write** to predict $w_i =$ **letter** without taking both w_{i-2} and w_{i-1} into account.

The different choices of order in which the units can be removed from the history can be illustrated by a *back-off graph* (Bilmes and Kirchhoff, 2003). Figure 4.5 shows a back-off graph for a 4-gram model. The standard back-off path is the leftmost one. Models that use nodes of the graph that are not in the standard path are called *skipping* models, as they skip the dependence of some w_j from the middle of the initial history.

There are a large number of paths that could be applied, and it is not clear which skipping estimators would be useful. In consequence, it makes sense to

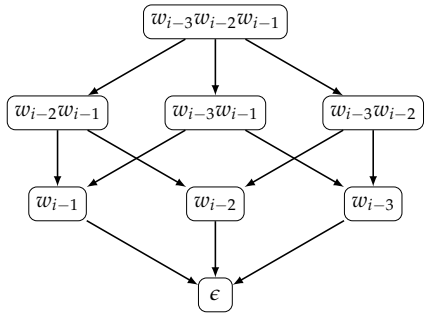


Figure 4.5. The back-off graph of a 4-gram model (Bilmes and Kirchhoff, 2003).

combine a set of different skip models together by interpolation. Such models can improve results if there is only small and intermediate amounts of training data available and longer n-grams cannot be used (Goodman, 2001b).

4.2.7 Factored models

The standard "oldest-first" back-off order is very sensible for models based on variables that are ordered in time. However, already the cluster n-gram models had two variables at each time step: the word w_i and its cluster c_i. In this case, the best back-off order is not evident. For example, Niesler and Woodland (1996a) propose a combined word and category model, which backs off from word-level history to category-level history if there is no estimate available in the word-level.

A general extension to this direction is the *factored n-gram model* (Bilmes and Kirchhoff, 2003; Duh and Kirchhoff, 2004), where each word is equivalent to a set of K factors: $w_i \equiv f_i^1 \dots f_i^K$. The factors of a word may, for example, consist of its stem, root and part-of-speech tag, as well as the word form itself. Given a set of factors for each position in the n-gram history, it is not evident whether, for example, the previous word form should be dropped before or after the part-of-speech of the word before that. For a 3-gram model with three factors, the number of possible back-off paths is $(3 \times 3)! = 362\,880$.

As a solution Bilmes and Kirchhoff (2003) proposes *generalized parallel back-off* that uses multiple paths of the back-off graph simultaneously. The paths can either be defined by hand or learned automatically from the training data (Duh and Kirchhoff, 2004).

4.3 Beyond n-gram models

While n-gram models are the most popular models for various language modeling tasks, they have their obvious drawbacks. This section reviews some models that either do not make the n-gram assumption or that move the discrete problem into a continuous domain.

4.3.1 Grammar-based language models

N-gram models have problems with syntactic long-distance dependencies (see Section 3.1.8) that can be handled with, for example, context-free grammars (Sec-

tion 3.2.4)[2] or dependency grammars (Section 3.2.5). In consequence, there have been various attempts to use grammatical parsers in language modeling.

In principle, any stochastic syntactic parsing method can be used for density estimation. Let $\mathcal{Z}_{\boldsymbol{s}}$ be the set of possible parses for sentence $\boldsymbol{s}$ and $p(\boldsymbol{s}, z)$ the probability of parse $z \in \mathcal{Z}_{\boldsymbol{s}}$. The probability of the sentence is then:

$$p(\boldsymbol{s}) = \sum_{z \in \mathcal{Z}_{\boldsymbol{s}}} p(\boldsymbol{s}, z). \tag{4.39}$$

Of course, the probability of the whole sentence is problematic to use in decoding. For PCFGs, there are also efficient parsing algorithms that calculate prefix probabilities $p(\boldsymbol{w}_1^i)$, where $i < |\boldsymbol{s}|$ (e.g. Stolcke, 1995; Stolcke and Segal, 1994). Using prefix probabilities, it is possible to calculate the conditional probabilities

$$p(w_i \mid \boldsymbol{w}_1^{i-1}) = \frac{p(\boldsymbol{w}_1^i)}{p(\boldsymbol{w}_1^{i-1})} \tag{4.40}$$

that are often required by ASR and SMT decoders. Jurafsky et al. (1995) show that a hand-crafted PCFG used with Stolcke's (1995) algorithm outperforms a bigram model in a medium-vocabulary speech recognition task.

More recent work is often based on *lexicalized* parsers that condition the probabilities of the rules by the lexical content (i.e. words) of the sentence. For example, Roark (2001) uses probabilistic top-down parsing to directly define the conditional probabilities of words. Another example is the *structured language model* by Chelba and Jelinek (2000). This model resembles n-gram models in that it defines a conditional probability of a word given a fixed number of words in the context. However, the context words are not the two immediate predecessors, but previous headwords defined by the parser.

Charniak (2001) uses an immediate-head parsing method that cannot derive prefix probabilities, as probability of events below a constituent depend on its head that may occur after the constituent. While this complicates the usage of the model, the perplexities of the model are lower than those of Chelba and Jelinek (2000) and Roark (2001).

A major problem of grammar-based models is that parsing is a computationally heavy operation. This hinders the applicability of the models, as decoders need probability estimates for a large number of different sentences to decide on each single output. Another limitation is that annotated corpora are required for training the parsers.

4.3.2 Maximum-entropy language models

Using clusters, factors, or skips in n-gram models can be considered as having different groupings of the language model histories $\boldsymbol{h}$. As discussed above, these models are often combined with linear interpolation to improve the results. The advantages of linear interpolation are that it is very general, easy to implement, and guaranteed to be no worse than any of its components. However, as the interpolation weights are optimized globally, it makes suboptimal use of the component models and violates their consistency (see Rosenfeld, 1996).

Maximum-entropy (ME) language models provide an alternative way to combine different knowledge sources into a single model. As described in Section 2.2.2, maximum-entropy models are log-linear models based on feature

[2] Although either highly granular non-terminals or lexicalized rules are required to model, for example, grammatical agreement; see Jurafsky and Martin (2008, Ch. 14).

functions $f_j(\mathbf{x})$. In the standard language modeling approach, the goal is to define the conditional distribution $p(w \mid \boldsymbol{h})$. This leads to a *conditional ME model* of the following form (Rosenfeld et al., 2001):

$$p_{\mathrm{ME}}(w \mid \boldsymbol{h}) = \frac{1}{Z(\boldsymbol{h})} p_0(w \mid \boldsymbol{h}) \exp\Big(\sum_{j=1}^{N} \lambda_j f_j(\boldsymbol{h}, w)\Big), \tag{4.41}$$

where $p_0(S)$ is an arbitrary initial distribution. The consistency of the model is ensured by constraining the expected values of the features f_j to their empirical expectations in the training data:

$$\sum_{\boldsymbol{h},w} p_{\mathrm{ME}}(\boldsymbol{h}w) f_j(\boldsymbol{h}, w) = \frac{1}{N} \sum_{i=1}^{N} f_j(\boldsymbol{h}_i, w_i). \tag{4.42}$$

The parameters are optimized to minimize the relative entropy $D(p \,||\, p_0)$, which is equivalent to maximum-entropy models if $p_0(w \mid \boldsymbol{h})$ is uniform. The models can be trained with the generalized iterative scaling algorithm by Darroch and Ratcliff (1972).

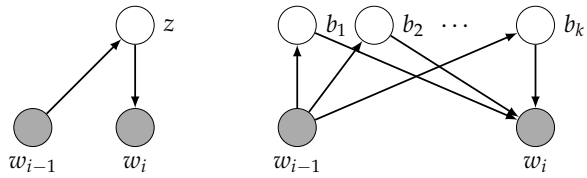
Rosenfeld (1996) uses the ME framework to combine n-gram features ($n = 2$, $n = 3$), skip n-gram features at distance two, and long-distance *trigger* features that predict $p(w_a \mid w_b \in \boldsymbol{h})$ for w_b in any position in the current document. Significant perplexity improvements over standard 3-gram models are obtained at the expense of computational burden of the training algorithm. One reason for the high computational burden of the conditional ME models is that the partition function $Z(\boldsymbol{h})$ has to be computed separately for each $\boldsymbol{h}$ by looping over the vocabulary. Techniques that speed up the training process include hierarchical factorization of the model by clustering the vocabulary (Goodman, 2001a) and using the nested structure of n-grams (Wu and Khudanpur, 2002). Alumäe and Kurimo (2010) provide an implementation that applies some of the recent speed-ups and show that they make it possible to use large training corpora with reasonable computation time and memory requirements.

Similar to n-gram models, the empirical expectations of rare features are often overestimated. Chen and Rosenfeld (2000) study different smoothing methods for conditional ME models. A simple and efficient smoothing is obtained by setting zero-mean Gaussian priors for the weights λ_i. However, Goodman (2004) shows that an exponential prior, which leads to smoothing by absolute discounting, is better motivated and improves the results also in practice.

Another maximum-entropy approach to statistical language modeling is the *whole-sentence ME model* by Rosenfeld et al. (2001):

$$p_{\mathrm{ME}}(\boldsymbol{s}) = \frac{1}{Z} p_0(\boldsymbol{s}) \exp\Big(\sum_{j=1}^{N} \lambda_j f_j(\boldsymbol{s})\Big). \tag{4.43}$$

An advantage of the whole-sentence ME model is the possibility to incorporate global sentence features. Moreover, unlike in the conditional models, the partition function Z is a constant. While the value of Z is infeasible to calculate, the model can be applied without normalization for selecting the most likely $\boldsymbol{s}$ or re-scoring N-best lists or lattices. A more serious problem is that the feature expectations over all possible sentences are required to train the model. While it is not feasible to calculate the exact values, the problem can be circumvented by sampling from $p(S)$ and using sample estimates. However, also the sampling is non-trivial and computationally intensive.



(a) AMM (b) DMM

Figure 4.6. (a) Aggregate Markov model by Saul and Pereira (1997). (b) Distributed Markov model by Blitzer et al. (2005).

4.3.3 Continuous-space language models

The models discussed so far are based on discrete representations of words: clusters, factors, or the words themselves. As mentioned in Section 3.1.3—and discussed further in Chapter 5—it is also possible to obtain continuous representations for linguistic units and their collections. The main advantage of continuous representations is that generalization and smoothing can simply be based on similarity measures.

Continuous-space models frequently use the n-gram assumption, which limits the history $\boldsymbol{h}$ to $n-1$ previous words. The essential part of the models is that the history is mapped to a continuous space using *shared* representations of words, meaning that each occurrence of the same word has the same representation regardless of the position of the word or its context. While the cluster n-gram models discussed in Section 4.2.5 also use shared representations, there each representation was a single discrete symbol selected in a deterministic manner.

4.3.3.1 Latent variable models

The first model that can be considered as a continuous-space language model is the *aggregate Markov model* (AMM) by Saul and Pereira (1997). It simply extends the hard clustering in the cluster bigram model in Brown et al. (1992) to soft clustering:

$$p(w_i \mid w_{i-1}) = \sum_{z \in \mathcal{Z}} p(w_i \mid z) p(z \mid w_{i-1}). \tag{4.44}$$

While the hidden variable z is still discrete, the distributions $p(z \mid w_{i-1})$ can be interpreted as continuous $|\mathcal{Z}|$-dimensional representations for the words w_i. The model, illustrated in Figure 4.6(a), is in essence the same as the PLSA model (Hofmann, 1999a) discussed later in Section 5.2. However, it predicts the joint distribution of the preceding word and the next word instead of the joint distribution of the current document and the next word. The parameters of the AMM can be estimated with the EM algorithm. In order to use longer histories, Saul and Pereira (1997) combine similar models with different skips with general linear interpolation.

Blitzer et al. (2005) extend the aggregate Markov model to *vectors* of latent variables $\mathbf{b} = (b_1, b_2, \ldots, b_m)$. The single variables b_i are assumed to be binary and conditionally independent given $\boldsymbol{h}$. The probabilities $p(w \mid \mathbf{b})$ are defined by a conditional maximum-entropy model and the bigram distribution is

$$p(w \mid w_{i-1}) = \sum_{\mathbf{b} \in \mathcal{B}} \left[\prod_{i=1}^{m} p(b_i \mid w_{i-1}) \times \frac{1}{Z(\mathbf{b})} \exp\Big(\sum_{i=1}^{m} \psi(b_i, w) \Big) \right], \tag{4.45}$$

where $\psi(b_i, w)$ is an arbitrary real-valued function. A graphical representation of the model, called *distributed Markov model* (DMM), is shown in Figure 4.6(b). As a mixture of $|\mathcal{B}| = 2^m$ components, the model requires only $O(m)$ parameters, while AMM requires $O(m)$ parameters for m components. In addition, Blitzer et al. (2005) extend the model to longer histories by having a new latent vector for each word in the history. They show that a 3-gram DMM outperforms a Kneser-Ney smoothed 3-gram model.

4.3.3.2 Neural network models

The number of free parameters in an n-gram model, V^n, grows exponentially in n unless further independence assumptions are made. The largest part of the parameters, V^{n-1}, are due to the different combinations of the n-gram history $\boldsymbol{h}$. Many continuous-space models use shared representations of words in order to have only a linear growth. Let the representation of word w_j be $\boldsymbol{r}_j \in \mathbb{R}^m$. To represent a history $\boldsymbol{h}_i = \boldsymbol{w}_{n-i+1}^{i-1}$, the distributed vectors are concatenated into an $(n-1)m$-dimensional vector:

$$\mathbf{x}_i = \left(\mathbf{r}_{n-i+1}^{\mathrm{T}} \mathbf{r}_{n-i+1}^{\mathrm{T}} \dots \mathbf{r}_{i-1}^{\mathrm{T}}\right)^{\mathrm{T}} \tag{4.46}$$

The context vector $\mathbf{x}_i$ is often projected into K dimensions using a matrix $\mathbf{W} = \left(\mathbf{W}_1^{\mathrm{T}} \dots \mathbf{W}_{n-1}^{\mathrm{T}}\right)^{\mathrm{T}} \in \mathbb{R}^{(n-1)m \times K}$. Some equations below will use the notation

$$\mathbf{W}^{\mathrm{T}} \mathbf{x}_i = \sum_{j=n-i+1}^{i-1} \mathbf{W}_j^{\mathrm{T}} \mathbf{R}^{\mathrm{T}} \mathbf{v}_j, \tag{4.47}$$

where $\mathbf{R}$ is a $V \times m$ matrix containing the shared vectors and $\mathbf{v}_j \in \{0,1\}^V$ is an index vector that has one in the w_j^{th} position and zeros elsewhere.

Various approaches have been proposed to obtain the probabilities $p(w \mid \mathbf{x}_i)$. So far the most popular approach has been the *neural language model* by Bengio et al. (2001, 2003). It uses a feedforward MLP network to first project $\mathbf{x}$ into a K-dimensional space using the hyperbolic tangent activation function:

$$\mathbf{z} = \tanh\left(\mathbf{W}^{\mathrm{T}} \mathbf{x} + \mathbf{b}_h\right), \tag{4.48}$$

where $\mathbf{b}_h \in \mathbb{R}^K$ is a bias vector. Next, $\mathbf{z}$ is projected into a V-dimensional vector by $\mathbf{y} = \mathbf{U}^{\mathrm{T}} \mathbf{z} + \mathbf{b}_u$, where $\mathbf{U} \in \mathbb{R}^{K \times V}$ and $\mathbf{b}_u \in \mathbb{R}^V$. The probabilities p_i of the following words w_i are given by the *softmax* normalization:

$$p_i = \frac{\exp(y_i)}{\sum_{j=1}^{V} \exp(y_j)}. \tag{4.49}$$

The network is trained with the standard back-propagation algorithm for MLPs. The computational complexity of calculating a single probability estimate is $O(K(nm + V))$, dominated by the size of the vocabulary V. To make it feasible to train the model with millions of examples and use it in real applications, many optimization techniques are required (cf. Bengio et al., 2003; Schwenk, 2007).

The neural language model has been applied in large vocabulary ASR systems by Schwenk and Gauvain (2002) and Schwenk (2007). It outperforms standard Kneser-Ney smoothed n-gram models both in terms of perplexity and word error rate. Further improvements are obtained by linear interpolation of the models. Neural language models have also been combined with other language

modeling techniques. For example, Emami et al. (2003) use an MLP with structured language models and Alexandrescu and Kirchhoff (2006) extend the neural language model with factored word representations.

Mnih and Hinton (2007) propose a similar model that is based on *restricted Boltzmann machine* (RBM) instead of an MLP network. RBMs are undirected graphical models. They have a set of observed variables and a set of binary hidden variables, and there are connections only between the observed and the hidden variables. The energy function (see Section 2.2.2) is

$$E(w_i, \mathbf{z}; \boldsymbol{w}_{i-n+1}^{i-1}) = \Big(\sum_{j=i-n+1}^{i} \mathbf{v}_j^{\mathrm{T}} \mathbf{R} \mathbf{W}_j \Big) \mathbf{z} + \mathbf{b}_h^{\mathrm{T}} \mathbf{z} + \mathbf{b}_r^{\mathrm{T}} \mathbf{R} \mathbf{v}_i + \mathbf{b}_v^{\mathrm{T}} \mathbf{v}_i. \tag{4.50}$$

The vector $\mathbf{z} \in \{0,1\}^K$ contains the configuration of the hidden units and $\mathbf{b}_h$, $\mathbf{b}_r$, and $\mathbf{b}_v$ are bias vectors. Similarly to ME language models, the partition function Z is conditional on the history $\boldsymbol{w}_{i-n+1}^{i-1}$, and normalization requires summing over V terms. Mnih and Hinton (2007) report that the mixture of a standard 5-gram model and a factored RBM language model with $n = 6$ obtains lower perplexity than the mixture of a 5-gram model and the neural language model by Bengio et al. (2003) with $n = 6$.

4.3.3.3 Log-bilinear model

Log-bilinear language model is another continuous-space model by Mnih and Hinton (2007). It is an undirected model similar to the RBM language model, but without the hidden variables. Instead, the model predicts the feature vector of the next word by computing a linear function of the feature vectors of the context words. The model can also be considered as an undirected analogue of the standard n-gram model in Figure 4.2. The energy function is defined by:

$$E(w_i; \boldsymbol{w}_{i-n+1}^{i-1}) = \Big(\sum_{j=i-n+1}^{i-1} \mathbf{v}_j^{\mathrm{T}} \mathbf{R} \mathbf{W}_j \Big) \mathbf{R}^{\mathrm{T}} \mathbf{v}_i + \mathbf{b}_r^{\mathrm{T}} \mathbf{R}^{\mathrm{T}} \mathbf{v}_i + \mathbf{b}_v^{\mathrm{T}} \mathbf{v}_i, \tag{4.51}$$

where $\mathbf{W}_i$ have weights and $\mathbf{b}_r$ and $\mathbf{b}_v$ bias terms. A similar model was earlier proposed by Bengio et al. (2003), but with an MLP network instead of the bilinear energy function. With respect to training, the log-bilinear model is much simpler than the RBM model. In the experiments of Mnih and Hinton (2007), it outperformed both the neural language model and the RBM model. Mnih and Hinton (2008) speed up the training using the hierarchical clustering technique proposed by Goodman (2001a) for ME models and Morin and Bengio (2005) for neural language models. Mnih et al. (2009) consider extensions with different types of non-linear interactions between the context words. Their best models significantly outperform the original log-bilinear model in terms of perplexity.

4.3.3.4 State-space models

Increasing the model order n in a neural or RBM language model increases the number of parameters linearly. This is a clear improvement to the exponential growth in n-gram models. However, in neural network models, it is also possible to add dependencies from the previous hidden states to the following states. This kind of *state-space models* can potentially take advantage of indefinitely long contexts.

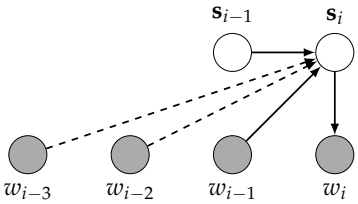


Figure 4.7. Directed state-space model by Siivola and Honkela (2003) and Mikolov et al. (2010). Mikolov et al. (2010) use only the previous state and word for predicting the next state (solid lines); Siivola and Honkela (2003) use the previous state and the $n-1$ previous words (dashed lines, here $n = 4$). The temporal factored RBM model by Mnih and Hinton (2007) is an undirected model with a similar structure.

Siivola and Honkela (2003) study a direct graphical model that predicts the next state $\mathbf{s}_i$ from the previous state $\mathbf{s}_{i-1}$ and distributed representations of the words in the observed history $\boldsymbol{h}$. A graphical representation of the model is shown in Figure 4.7. The word w_i is predicted from $\mathbf{s}_i$ using the softmax function after a linear transform. The model is trained with an EM-like algorithm with numerical approximations. It is tested only for letter prediction and it does not outperform standard n-gram models in this task.

Recently, Mikolov et al. (2010) have proposed a very similar state-space model called *recurrent neural network* (RNN) language model. The main differences are that Mikolov et al. (2010) apply a sigmoid activation function when determining $\mathbf{s}_i$, use only the previous word for predicting the new state (i.e., $n = 2$), and train the model with the back-propagation algorithm. In addition to static models whose parameters stay fixed after the training phase, they consider a dynamic model that is updated also when processing the test data. The perplexity and speech recognition experiments of Mikolov et al. (2010) show that both the static and dynamic RNN models outperform Kneser-Ney smoothed 5-gram model, and linear interpolation of the three models improve the results further. Using a mixture of several RNNs that differ in the initialization of weights and the size of the hidden layer yields additional improvements. The extensive comparison of advanced language modeling techniques by Mikolov et al. (2011) show that a mixture of twenty RNN language models outperform many other state-of-the-art models—including the structured language model, ME language model, feedforward neural network model, and log-linear language model.

Promising results for state-space modeling were obtained also by Mnih and Hinton (2007). Their *temporal factored RBM model* is an extension of the RBM language model above. It includes direct connections between the previous state of the hidden vectors $\mathbf{z}_{i-1}$ and the next state $\mathbf{z}_i$. The training of the model is demanding, but it is shown to outperform the standard RBM 2-gram model, as well as a log-bilinear 5-gram model when both are interpolated with a KN smoothed 6-gram model.[3]

4.3.4 Models for domain adaptation

Grammar-based language models, maximum-entropy models, as well as continuous-space language models can improve on n-gram models by taking into account long-distance dependencies between the words in the same sentence.

[3] The temporal factored RBM model is not included in the comparison by Mikolov et al. (2011), but as its results seem to be very close to those of the log-bilinear model, it is not likely to outperform the RNN model.

However, with the exception of the dynamic state-space model by Mikolov et al. (2010), they assume that the data is stationary: the probability distribution $p(S)$ is not expected to change over time. In practice, of course, different corpora as well as parts of the same corpus have different authors, topics, genres, styles, and so on. Their configurations are called *domains*, and approaches to deal with the changes in domain is called domain adaptation.

Rosenfeld (1996) identifies three types of adaptation. In *within-domain adaptation*, a heterogeneous source of language is considered as a mixture of multiple topics or domains. On-line adaptation is needed for dealing with the changes in the text. In *cross-domain adaptation*, test data comes from a different domain than the training data. The latter is called source domain and the former is called target domain. As there is no target domain data available before training, the model has to be adapted on-line also in this case.

Finally, in *limited-domain adaptation*, there is a limited amount of data available from the target domain for training in addition to a large source domain corpus. There are three basic approaches for this setup: combining the data sets and training a single model on the pooled data, training separate models for source and target domains and interpolating between them, or merging the source and target domain counts at the level of individual n-grams. In all cases, target and source domains can be weighted using optimal weights for held-out data. Bacchiani and Roark (2003) show that interpolating and count merging can be interpreted as MAP estimates with different Dirichlet priors on the categorical distribution of the model. They found that count merging slightly outperforms interpolation in speech recognition experiments.

Maximum-entropy language models can be adapted by setting a prior based on the source domain corpus for the parameters of the target domain model, either in a heuristic (Chelba and Acero, 2004; Daumé III, 2007) or more principled (Finkel and Manning, 2009) manner. Alumäe and Kurimo (2010) show that the hierarchical Bayesian adaptation by Finkel and Manning (2009) outperforms linear interpolation of domain-specific ME language models both in terms of perplexity and error rate in speech recognition.

Considering specific models for within-domain and cross-domain adaptation, there are two popular model types: *cache models* and *topic models*. In most cases, they are combined with standard models trained on the source domain data. In addition to linear interpolation, common combination methods include log-linear interpolation and unigram rescaling (see e.g. Broman and Kurimo, 2005).

Cache language models (Kuhn and De Mori, 1990) are based on the observation that recently occurred words are likely to re-occur. The models store the data observed in the current document so far, and use n-gram models estimated on this data for new predictions. The cache models are usually unigram models. If the occurrences of the K previous words are stored as $c_{\text{cache}}(w_i)$, $p_{\text{cache}}(w_i \mid \boldsymbol{h}) \approx c_{\text{cache}}(w_i)/K$. Clarkson and Robinson (1997) propose using exponential decays to decrease the effect of the words observed further away in the history. Frequent words, which are likely to be function words, can be excluded from the cache using count cut-offs (Rosenfeld, 1994; Iyer and Ostendorf, 1999). Rosenfeld (1994, 1996) has incorporated cache modeling in the maximum-entropy language models.

Topic language models are trained on document collections that can be divided into several topics (or domains). The model has one component for pre-

dicting words for each topic t_j:

$$p(w_i \mid \boldsymbol{h}) = \sum_{j=1}^{K} p(t_j \mid \boldsymbol{h}) p(w_i \mid t_j, \boldsymbol{h}) \tag{4.52}$$

The first topic models were mixtures of standard n-gram models such that each model was trained on a subset of the training corpus (Clarkson and Robinson, 1997; Iyer and Ostendorf, 1999). Weights of the models $p(t_j \mid \boldsymbol{h})$ were assigned based on the observed history $\boldsymbol{h}$ using the EM algorithm. More recent approaches apply the vector space models and probabilistic topic models used in IR and text categorization tasks (see Chapter 5). Usually $p(w_i \mid t_j, \boldsymbol{h})$ is set to be a unigram model and thus independent on $\boldsymbol{h}$. Accordingly, the model concentrates on the domain-level information and does not incorporate sentence-level dependencies. Thus it is always interpolated with a standard n-gram model. Methods used for topic language models include LSA (Coccaro and Jurafsky, 1998; Bellegarda, 2000), PLSA (Gildea and Hofmann, 1999), non-negative matrix factorization (Novak and Mammone, 2001), SOM (Lagus and Kurimo, 2002), and LDA (Tam and Schultz, 2005).

Cache models and topic models can be combined for better performance in adaptation tasks. Apart from standard combining techniques such as linear interpolation (Clarkson and Robinson, 1997), one can use topic-specific caches (Iyer and Ostendorf, 1999) or develop models that incorporate the main ideas of both approaches (Chueh and Chien, 2010).

4.4 Kneser-Ney smoothing and pruned models

Compared with the grammar-based language models, maximum-entropy language models, continuous-space language models, as well as Bayesian Pitman-Yor language models, the standard n-gram models with Kneser-Ney smoothing still offer an excellent trade-off between model performance and simple and quick training and prediction. Kneser-Ney smoothed models have been trained with data sets up to 10^{11} words (Brants et al., 2007), an amount much larger than what is easily available for most languages. The only major problem, exponential growth of model parameters with respect to the n-gram length n, can be alleviated by pruning and growing techniques.

However, the use of model pruning and growing for the type-based distributions used in Kneser-Ney (KN) smoothing (Kneser and Ney, 1995) raises some non-trivial problems. Recall from Section 4.2.3 that lower-order distributions of the KN model $p(w \mid \bar{\boldsymbol{h}})$, $|\bar{\boldsymbol{h}}| < n$, are based on the assumption that the higher-order distributions $p(w \mid \boldsymbol{h})$ could not predict the word, that is, $c(\boldsymbol{h}w) = 0$. Now if $c(\boldsymbol{h}w) > 0$ but the n-gram $\boldsymbol{h}w$ is pruned, this assumption does not hold. To use the example from Section 4.2.3, if **San Francisco** is pruned, then $p(\textbf{Francisco})$ *should* be affected by the count $c(\textbf{San Francisco})$, not only by $t(\bullet\textbf{Francisco})$, that is, the number of distinct contexts **Francisco** has appeared in.

While the first known solution to this issue was introduced already by Kneser (1996), it was not systematically studied until Publication I of this thesis. The *revised Kneser pruning* proposed in Publication I shows clear experimental improvements over previous approaches, including the original Kneser pruning. Also a third version of Kneser-Ney smoothing for varigram models will be discussed. It was applied—but not adequately described—in Publication II.

In addition to presenting the Kneser-Ney formulas, this section will summarize the pruning and growing algorithms and cross-entropy and speech recognition experiments of Publication I.

4.4.1 Kneser-Ney distributions for varigram models

Let $\mathrm{A}_{\bullet \boldsymbol{h}w} = \{v\boldsymbol{h}w : v \in \Sigma \wedge c(v\boldsymbol{h}w) > 0\}$ be the observed n-grams that have $\boldsymbol{h}w$ as a suffix, $\mathrm{E}_{\bullet \boldsymbol{h}w} \subseteq \mathrm{A}_{\bullet \boldsymbol{h}w}$ be the subset of the n-grams that are pruned, and $\mathrm{I}_{\bullet \boldsymbol{h}w} = \mathrm{A}_{\bullet \boldsymbol{h}w} \setminus \mathrm{E}_{\bullet \boldsymbol{h}w}$ be the subset of the n-grams that are included in the model. Now consider $p_{\mathrm{KN}}(w \mid \bar{\boldsymbol{h}})$ in the case that $\bar{\boldsymbol{h}}w$ is not pruned. Neglecting interpolation and smoothing, the probability of an interpolated Kneser-Ney model (Equation 4.26) can be written as

$$p_{\mathrm{KN}}(w \mid \bar{\boldsymbol{h}}) \propto \begin{cases} c(\bar{\boldsymbol{h}}w) = \sum_{\substack{v \in \Sigma: \\ c(v\bar{\boldsymbol{h}}w) > 0}} c(v\bar{\boldsymbol{h}}w), & \text{if } \mathrm{I}_{\bullet \bar{\boldsymbol{h}}w} = \emptyset \\ t(\bullet\bar{\boldsymbol{h}}w) = \sum_{\substack{v \in \Sigma: \\ c(v\bar{\boldsymbol{h}}w) > 0}} 1, & \text{if } \mathrm{E}_{\bullet \bar{\boldsymbol{h}}w} = \emptyset \end{cases} .$$

For proper subsets $\mathrm{I}_{\bullet \boldsymbol{h}w} \subset \mathrm{A}_{\bullet \boldsymbol{h}w}$, the probability should evidently be somewhere between these two extremes of the token-based estimate and the type-based estimate.

Kneser (1996) derives modified counts for a pruned Kneser-Ney smoothed model using the marginal constraint in Equation 4.23. In Kneser pruning (KP), the lower-order distribution is defined by

$$\begin{aligned} p_{\mathrm{KP}}(w \mid \bar{\boldsymbol{h}}) \;&\propto\; c(\bar{\boldsymbol{h}}w) - \sum_{\substack{v \in \Sigma: \\ v\bar{\boldsymbol{h}}w \in \mathrm{I}_{v\bar{\boldsymbol{h}}\bullet}}} \left[c(v\bar{\boldsymbol{h}}w) - D_{|v\bar{\boldsymbol{h}}|} \right] \\ &= \sum_{\substack{v \in \Sigma: \\ v\bar{\boldsymbol{h}}w \in \mathrm{E}_{\bullet \bar{\boldsymbol{h}}w}}} c(v\bar{\boldsymbol{h}}w) + D_{|v\bar{\boldsymbol{h}}|} \times \sum_{\substack{v \in \Sigma: \\ v\bar{\boldsymbol{h}}w \in \mathrm{I}_{\bullet \bar{\boldsymbol{h}}w}}} 1. \end{aligned} \tag{4.53}$$

A practical problem in this formula is that it includes the discounting constant: the lower-order distribution is dependent on the discount $D_{|v\bar{\boldsymbol{h}}|}$ of the higher order distribution. Having multiple discounts per order would complicate the dependency issues further. Moreover, it is not evident which one of the multiple discounts should be applied in Equation 4.53.

In the revised Kneser pruning (RKP) presented in Publication I, the probability $p_{\mathrm{KN}}(w \mid \bar{\boldsymbol{h}})$ is set proportional to the sum

$$p_{\mathrm{RKP}}(w \mid \bar{\boldsymbol{h}}) \propto \sum_{\substack{v \in \Sigma: \\ v\bar{\boldsymbol{h}}w \in \mathrm{E}_{\bullet \bar{\boldsymbol{h}}w}}} c(v\bar{\boldsymbol{h}}w) + \sum_{\substack{v \in \Sigma: \\ v\bar{\boldsymbol{h}}w \in \mathrm{I}_{\bullet \bar{\boldsymbol{h}}w}}} 1. \tag{4.54}$$

That is, each included n-gram $v\bar{\boldsymbol{h}}w$ increases the lower-order count by one and each pruned n-gram increases the lower-order count by the count of the n-gram. The only difference between KP and RKP is that the former weights the second term of the sum by the discount $D_{|v\boldsymbol{h}|}$. Usually $D < 1$ and thus type counts have a larger effect on the probability in RKP than in KP.

A slightly different approach for combining the token and type-based estimates was used in Publication II. Here, it will be called *weighted Kneser pruning* (WKP). The formula is derived from the observation that backing off from $\boldsymbol{h}$ to $\bar{\boldsymbol{h}}$ means that the longer context $\boldsymbol{h}$ was either not observed ($\boldsymbol{h} \notin \mathrm{A}_{\bullet \bar{\boldsymbol{h}}}$) or it was observed but not included in the model ($\boldsymbol{h} \notin \mathrm{I}_{\bullet \bar{\boldsymbol{h}}}$). By simply using the rules of

probability calculus,

$$\begin{aligned} p_{\text{WKP}}(w \mid \bar{h}; \boldsymbol{h} \notin \text{I}) &= p(w; \boldsymbol{h} \notin \text{A} \mid \bar{h}; \boldsymbol{h} \notin \text{I}) + p(w; \boldsymbol{h} \in \text{A} \mid \bar{h}; \boldsymbol{h} \notin \text{I}) \\ &= p(\boldsymbol{h} \notin \text{A} \mid \bar{h}; \boldsymbol{h} \notin \text{I}) \times p(w \mid \bar{h}; \boldsymbol{h} \notin \text{I} \wedge \boldsymbol{h} \notin \text{A}) + \\ &\quad p(\boldsymbol{h} \in \text{A} \mid \bar{h}; \boldsymbol{h} \notin \text{I}) \times p(w \mid \bar{h}; \boldsymbol{h} \notin \text{I} \wedge \boldsymbol{h} \in \text{A}) \\ &= p(\boldsymbol{h} \notin \text{A} \mid \bar{h}; \boldsymbol{h} \notin \text{I}) \times p(w \mid \bar{h}; \boldsymbol{h} \notin \text{A}) + \\ &\quad p(\boldsymbol{h} \in \text{A} \mid \bar{h}; \boldsymbol{h} \notin \text{I}) \times p(w \mid \bar{h}; \boldsymbol{h} \in \text{E}) \end{aligned} \tag{4.55}$$

To simplify the notation, the subscripts of I, E, and A have been excluded. The first term $p(w \mid \bar{h}; \boldsymbol{h} \notin \text{A})$ can be approximated by going through the training data and observing how many times $\bar{h}$ occurs in a new context:

$$p(w \mid \bar{h}; \boldsymbol{h} \notin \text{A}) = \frac{|\text{E}|}{|\text{E}| + \sum_{v: v\bar{h} \in \text{E}} c(v\bar{h})} \tag{4.56}$$

A similar type of estimate is used in the Witten-Bell smoothing (Equation 4.11). Of course, if E is empty, $p(\boldsymbol{h} \notin \text{A} \mid \bar{h}; \boldsymbol{h} \notin \text{I}) = 1$. The second term $p(w \mid \bar{h}; \boldsymbol{h} \notin \text{A})$ is simply the type-based Kneser-Ney estimate (Equation 4.25). The third term $p(w \mid \bar{h}; \boldsymbol{h} \in \text{A}) = 1 - p(w \mid \bar{h}; \boldsymbol{h} \notin \text{A})$. Finally, $p(w \mid \bar{h}; \boldsymbol{h} \in \text{E})$ means that w is generated by one of the pruned histories. Accordingly,

$$p(w \mid \bar{h}; \boldsymbol{h} \in \text{E}) = \frac{\sum_{v: v\bar{h} \in \text{E}} c(v\bar{h}w)}{\sum_{v: v\bar{h} \in \text{E}} c(v\bar{h})}. \tag{4.57}$$

Substituting the terms in Equation 4.55 for the estimates above,

$$p_{\text{WKP}}(w \mid \bar{h}; \boldsymbol{h} \notin \text{I}) = \frac{|\text{E}|}{|\text{E}| + \sum_{v: v\bar{h} \in \text{E}} c(v\bar{h})} \times \frac{t(\bullet\bar{h}w)}{t(\bullet\bar{h}\bullet)} + \frac{\sum_{v: v\bar{h} \in \text{E}} c(v\bar{h}w)}{|\text{E}| + \sum_{v: v\bar{h} \in \text{E}} c(v\bar{h})}. \tag{4.58}$$

This is a weighted sum of the type and token based probabilities. The weight of the type-based estimate varies between 0.5 (if $c(v\bar{h}) = 1 \, \forall v\bar{h} \in \text{E}$) and one. The essential difference to KP and RKP is that WKP considers only pruning of complete histories $v\bar{h}$, not individual n-grams $v\bar{h}w$.

4.4.2 Pruning and growing algorithms

Using the modified counts $b(\bar{h}w)$, the full Kneser-Ney / RKP model can be written as:

$$p_{\text{RKP}}(w \mid \boldsymbol{h}) = \begin{cases} \frac{b(\boldsymbol{h}w) - D_{|\boldsymbol{h}|}}{s(\boldsymbol{h})} & + \quad \gamma(\boldsymbol{h}) \times p_{\text{RKP}}(w \mid \bar{h}), \quad \text{if } \boldsymbol{h}w \in \text{I}_{\boldsymbol{h}\bullet} \\ 0 & + \quad \gamma(\boldsymbol{h}) \times p_{\text{RKP}}(w \mid \bar{h}), \quad \text{if } \boldsymbol{h}w \in \text{E}_{\boldsymbol{h}\bullet} \end{cases}, \tag{4.59}$$

where

$$b(\bar{h}w) = \sum_{\substack{v \in \Sigma: \\ v\bar{h}w \in \text{E}_{\bullet\bar{h}w}}} c(v\bar{h}w) + \sum_{\substack{v \in \Sigma: \\ v\bar{h}w \in \text{I}_{\bullet\bar{h}w}}} 1 \tag{4.60}$$

$$s(\boldsymbol{h}) = \sum_{v \in \Sigma} b(\boldsymbol{h}v) \tag{4.61}$$

$$\gamma(\boldsymbol{h}) = \frac{|\text{I}_{\boldsymbol{h}\bullet}| D_{|\boldsymbol{h}|} + l(\boldsymbol{h})}{s(\boldsymbol{h})} \tag{4.62}$$

$$l(\boldsymbol{h}) = \sum_{g \in \text{E}_{\boldsymbol{h}\bullet}} c(\boldsymbol{g}). \tag{4.63}$$

ADDGRAM($\boldsymbol{h}w$)
1 $b(\boldsymbol{h}w) \leftarrow c(\boldsymbol{h}w)$
2 $s(\boldsymbol{h}) \leftarrow s(\boldsymbol{h}) + c(\boldsymbol{h}w)$
3 **if** $b(\bar{\boldsymbol{h}}w) > 0$
4 $\quad b(\bar{\boldsymbol{h}}w) \leftarrow b(\bar{\boldsymbol{h}}w) - c(\boldsymbol{h}w) + 1$
5 $\quad s(\bar{\boldsymbol{h}}) \leftarrow s(\bar{\boldsymbol{h}}) - c(\boldsymbol{h}w) + 1$

PRUNEGRAM($\boldsymbol{h}w$)
1 $l(\boldsymbol{h}) \leftarrow l(\boldsymbol{h}) + b(\boldsymbol{h}w)$
2 **if** $b(\bar{\boldsymbol{h}}w) > 0$
3 $\quad b(\bar{\boldsymbol{h}}w) \leftarrow b(\bar{\boldsymbol{h}}w) + b(\boldsymbol{h}w) - 1$
4 $\quad s(\bar{\boldsymbol{h}}) \leftarrow s(\bar{\boldsymbol{h}}) + b(\boldsymbol{h}w) - 1$
5 $b(\boldsymbol{h}w) \leftarrow 0$

Figure 4.8. Algorithms for adding and pruning n-grams for Kneser-Ney smoothed models used by revised Kneser pruning and Kneser-Ney growing.

The main issue in the implementation is to keep track of the modified count $b(\boldsymbol{h}w)$ in Equation 4.60, their sums $s(\boldsymbol{h})$ in Equation 4.61, and the sum of the counts of the pruned n-grams $l(\boldsymbol{h})$ in Equation 4.63. The necessary updates when an individual n-gram $\boldsymbol{h}w$ is added or pruned are shown by the ADDGRAM and PRUNEGRAM algorithms in Figure 4.8.

The proposed pruning algorithm processes the higher order n-grams before lower order n-grams and considers a single n-gram at a time. Thus in contrast to the pruning algorithms in Section 4.2.4, the decision on a certain n-gram may affect the later decisions. The algorithm for pruning one order, PRUNEORDER(k, ϵ), is shown in Figure 4.9. The pruning criterion is based only the log-probability $p_{\text{RKP}}(w \mid \boldsymbol{h})$ of the pruned n-gram $\boldsymbol{h}w$, similarly to weighted difference pruning and Kneser pruning. The probability is weighted by the actual count $c(\boldsymbol{h}w)$. If the decrease in the log-probability is smaller then a chosen constant ϵ, the n-gram is pruned.

The proposed growing algorithm (GROWORDER(k, δ) in Figure 4.9) is called *Kneser-Ney growing*. It adds one distribution $\{\boldsymbol{h}w : w \in \Sigma \wedge c(\boldsymbol{h}w) > 0\}$ at a time. The histories are grown forward: $\boldsymbol{h}$ is considered only if $b(\boldsymbol{h}) > 0$. There are two reasons for this: First, the n-grams are stored in a prefix tree (see Section 4.2.4) to allow later pruning of single n-grams. Second, growing backward has a problem noted already by Ristad and Thomas (1995): If the distributions $p(w \mid \boldsymbol{h})$ and $p(w \mid v\boldsymbol{h})$ are very similar, the latter is unlikely to be added to the model. Then the distribution $p(w \mid uv\boldsymbol{h})$ is never considered even if it might change the estimates remarkably.[4] When the model is grown forward, $p(w \mid uv\boldsymbol{h})$ is more likely to be added, because also $p(w \mid u)$, $p(w \mid uv)$, and so forth are likely to be useful for the model.

The growing criterion of the algorithm is based on two-part MDL and follows the coding scheme proposed by Siivola and Pellom (2005). Assuming the prefix tree structure for n-grams and a fixed number of bits, α, to store the parameters of a single n-gram, the change in model cost is

$$\Delta l(\boldsymbol{\theta}) = \alpha(N_{\text{new}} - N_{\text{old}}) + N_{\text{new}} \log_2 N_{\text{new}} - N_{\text{old}} \log_2 N_{\text{old}}, \tag{4.64}$$

where N_{old} and N_{new} are the numbers of n-grams in the model before and after growing, respectively. The model coding is obviously defective, but practical, as it follows the actual implementation of the model (cf. Section 2.6.8). If $\delta \times \Delta l(\boldsymbol{\theta})$,

[4] Consider, for example, $u =$ **government**, $v =$ **did**, $\boldsymbol{h} =$ **not confirm**.

```
BUILDVARIGRAM(δ, ε)
1  k ← 1
2  while new n-grams are added do
3          GROWORDER(k, δ)
4          k ← k + 1
5  while k > 1 do
6          k ← k − 1
7          PRUNEORDER(k, ε)
8   re-estimate all discount parameters D_i

GROWORDER(k, δ)
 1  for {h : |h| = k − 1 ∧ b(h) > 0} do
 2      size_0 ← |{g : b(g) > 0}|
 3      logprob_0 ← Σ_{w∈Σ:c(hw)>0} c(hw) log_2 p_RKP(w | h)
 4      for w ∈ Σ : c(hw) > 0 do
 5          ADDGRAM(hw)
 6      size_1 ← |{g : b(g) > 0}|
 7      logprob_1 ← Σ_{w∈Σ:c(hw)>0} c(hw) log_2 p_RKP(w | h)
 8      sizecost ← α · (size_1 − size_0) + size_1 log(size_1) − size_0 log(size_0)
 9      if logprob_1 − logprob_0 ≤ δ · sizecost
10          undo previous ADDGRAM(hw) for each w
11   re-estimate all discount parameters D_i

PRUNEORDER(k, ε)
1  for {hw : |hw| = k ∧ b(hw) > 0} do
2      logprob_0 ← c(hw) log_2 p_RKP(w | h)
3      PRUNEGRAM(hw)
4      logprob_1 ← c(hw) log_2 p_RKP(w | h)
5      if logprob_1 < logprob_0 − ε
6          undo previous PRUNEGRAM
```

Figure 4.9. Kneser-Ney growing and revised Kneser pruning algorithms for building varigram models.

where $\delta > 0$ is a constant, is smaller than the increase in the log-likelihood of the training data, the candidate distribution is added to the model.

Figure 4.9 also shows a simple way to combine the growing and pruning: the algorithm BUILDVARIGRAM(δ, ϵ) first grows the model one order at a time, starting from the unigram distribution. When the model stops growing, the algorithm prunes single n-grams starting from the highest order and stopping at unigrams. Optimally, the growing parameter δ should be as large as the available memory and computation time allows, and the pruning parameter ϵ should be selected so that the desired model size is reached after pruning.

4.4.3 Experiments

The algorithms of Publication I were tested on cross-entropy evaluations and in a Finnish speech recognition task. Because of the synthetic and agglutinative morphology of Finnish, Morfessor Baseline (Creutz and Lagus, 2005b) was used to split the words into morphs prior to language model training. The 150 million word training data had 460 million morph tokens and 8 428 morph types. A part consisting of 110 000 morph tokens was taken as a held-out set and 510 000 tokens as a test set. The audio data consisted of clean speech by adult speakers. There were 26 h of data from 207 speakers for training, 1 h of data from 20 new

Table 4.1. N-gram and varigram models compared in Publication I.

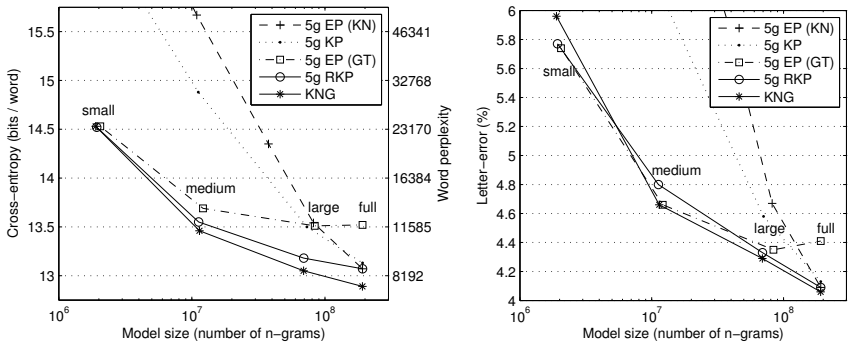
Abbreviation	Smoothing	Growing / max. n	Pruning
EP (GT)	Good-Turing	5	entropy-based
EP (KN)	modified Kneser-Ney	5	entropy-based
KP	Kneser-Ney	5	Kneser
RKP	modified Kneser-Ney	5	revised Kneser
KNG	modified Kneser-Ney	Kneser-Ney growing	revised Kneser

speakers for development, and 1.5 h of data from 31 new speakers for testing. The applied speech recognizer is described by Kurimo et al. (2006b).

The two proposed algorithms—revised Kneser pruning (RKP) and Kneser-Ney growing (KNG)—were compared with three other approaches that used combinations of Kneser-Ney and Good-Turing smoothing and entropy-based and Kneser pruning. The approaches are summarized in Table 4.1. The KNG model was grown to the same size as the full 5-gram models prior to pruning. Then all models were pruned to three different sizes (large, medium, small). The sizes were measured by the number of n-grams in the models. Pairwise one-sided signed-rank Wilcoxon test was performed to evaluate the statistical significance ($p < 0.01$) of the differences.

The cross-entropies of the models, normalized to bits per word, are shown in Figure 4.10(a). The results show that (1) KN smoothing is better than GT for the full model, and (2) neither EP nor KP works well for pruning the KN smoothed models. In consequence, both KNG and RKP outperformed the other models for large and medium model sizes. Moreover, KNG outperformed RKP (and all other models) except for the smallest model size.

The speech recognition results are shown in Figure 4.10(b). Again, KN smoothing is the best option for the full model and neither EP (KN) nor KP works well. However, in this case there was no statistically significant difference between the results of EP (GT), KNG and RKP except for the full models, for which the Good-Turing smoothed model was significantly worse than the others.



(a) (b)

Figure 4.10. Comparison of the entropy-based pruning (EP), Kneser pruning (KP), revised Kneser pruning (RKP), and Kneser-Ney growing (KNG) in Publication I: (a) cross-entropy and perplexity on a Finnish corpus and (b) LER in a Finnish speech recognition task. EP has been run both for Kneser-Ney smoothed (KN) and Good-Turing smoothed (GT) models.

4.4.4 Discussion

While the lower order distribution for Kneser pruning (Equation 4.53) is very similar to that of the revised Kneser pruning (Equation 4.54), the experiments show a significant difference for the results of the two algorithms. The main reason is that Kneser pruning, similarly to entropy-based pruning, makes all pruning decisions independently and at the same time. In contrast, revised Kneser pruning makes one decision at a time and updates the lower-order statistics every time an n-gram is pruned. Another difference is that RKP prunes only leaf n-grams or complete subtrees, while preliminary experiments indicated that excluding this restriction improves the results. Preliminary experiments also showed that if the RKP pruning algorithm was used with the lower order distribution defined in KP, the results were close to RKP, but not any better. Thus, while satisfying the marginal constraint is theoretically motivated, it is sometimes restricts the n-gram models too much.

The benefit of the Kneser-Ney growing compared with the revised Kneser pruning is illustrated by Figure 4.11 that shows how the n-grams included in the models are distributed on different n-gram orders. For small and medium models, KNG and RKP have similar distributions, but for larger sizes, KNG uses considerably longer histories. For the largest models trained with RKP, most of the n-grams are 5-grams, which clearly indicates too low a model order. In contrast, the n-gram distributions obtained with KNG are very similar to those reported by Mochihashi and Sumita (2008), who use a non-parametric Bayesian approach to select the n-gram lengths for hierarchical Pitman-Yor language models.

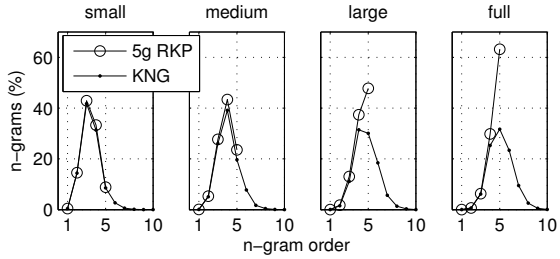


Figure 4.11. Distribution of the n-gram lengths in pruned 5-gram models (RKP) and grown models (KNG) with increasing model size (Publication I).

4.5 Clustering of n-gram histories

In addition to varigram models, the size of the n-gram models can be reduced by clustering (e.g., Brown et al., 1992; Gao et al., 2002). The combination of both the extensions is shown to be useful, for example, by Niesler and Woodland (1999) and Blasig (1999). The standard cluster n-grams are, however, easily applicable only to models that have a large vocabulary to begin with. This contrasts with varigram models that are able to determine the model order from the data and thus can be based on very short units, such as morphemes, syllables or characters. Context-independent clustering of a very small number of lexical units is not a promising approach.

An alternative clustering approach, studied in Publication II, is to cluster the

n-gram histories $\boldsymbol{h}$ instead of the individual units w. This is feasible for any type of units. The approach can be motivated by that paradigmatic relations exist not only between single units of the same type, but between many sequences of different types and lengths. For example, the following distributions are likely to be similar:

- $p(W \mid \textbf{U.S. military officials said that})$ and $p(W \mid \textbf{Pentagon said that})$
- $p(W \mid \textbf{in spite of})$ and $p(W \mid \textbf{regardless of})$
- $p(W \mid \textbf{un finish ed <w>})$ and $p(W \mid \textbf{not <w> finish ed <w>})$
- $p(W \mid \textbf{s h o r t e})$ and $p(W \mid \textbf{l o n g e})$

In the third example, the model is based on morphs, and **<w>** is a word boundary token. In the last example, the model is based on characters.

4.5.1 Context cluster model

For an n-gram model of order n, let $\Omega \subseteq \epsilon \cup \Sigma \cup \Sigma^2 \cup \ldots \cup \Sigma^{n-1}$ be the set of histories $\boldsymbol{h}$ in the model, including the empty history ϵ. The model proposed in Publication II assigns the histories $\boldsymbol{h} \in \Omega$ to clusters $\Pi = \{c_1, \ldots, c_K\}$. Assuming that the next unit w depends only on the cluster of the history, not on the history itself, and that each history is assigned to belong to exactly one cluster by the surjection $\pi : \Omega \mapsto \Pi$, the conditional probability is

$$p(w \mid \boldsymbol{h}) \approx p(w \mid \pi(\boldsymbol{h})). \tag{4.65}$$

This probability can be estimated simply by accumulating the observed counts within the cluster. With absolute discounting and interpolation between different context lengths,

$$p_{\mathrm{ACC}}(w \mid \pi(\boldsymbol{h})) = \begin{cases} \frac{\max(b(\pi(\boldsymbol{h}),w)-D,0)}{s(\pi(\boldsymbol{h}))} + \gamma(\pi(\boldsymbol{h})) \times p_{\mathrm{ACC}}(w \mid \pi(\bar{\boldsymbol{h}})), & \text{if } \boldsymbol{h} \in \Omega \\ p_{\mathrm{ACC}}(w \mid \pi(\bar{\boldsymbol{h}})), & \text{if } \boldsymbol{h} \notin \Omega \end{cases} \tag{4.66}$$

where

$$b(c, w) = \sum_{\boldsymbol{g} \in c} c(\boldsymbol{g}w), \tag{4.67}$$

$$s(c) = \sum_{w \in \Sigma} b(c, w), \tag{4.68}$$

$$\gamma(c) = \frac{D \times v(c)}{s(c)}, \tag{4.69}$$

and $v(c) = |\{w \in \Sigma : b(c, w) > 0\}|$ is the number of non-zero estimates in a cluster. In this thesis, this approach will be called *accumulative context clustering* (ACC). In theory, the ACC model is able to (1) reduce the number of parameters by storing only one distribution instead of all distributions of the contexts in the cluster, (2) increase the number of training samples per each distribution, and (3) generalize the predictive distributions.

Using Kneser-Ney smoothing for the ACC model during the training phase is problematic, as adding a new history would affect the distributions of many clusters. However, the distributions can also be modified afterwards. In Publication II, absolute discounting of the standard ML estimates (Equation 4.66)

was used during training. Then the weighted Kneser pruning described in Section 4.4.1 was applied to each history to obtain $p_{\mathrm{WKP}}(w \mid \boldsymbol{g})$ (Equation 4.58), and the counts $b(c, w)$ of the cluster were set to pseudo-counts

$$b^*(c, w) = \sum_{\boldsymbol{g} \in c} c(\boldsymbol{g}) p_{\mathrm{WKP}}(w \mid \boldsymbol{g}). \tag{4.70}$$

Regardless of the changes of the individual predictive distributions within each cluster, this *ad-hoc* smoothing still improved the cross-entropy of the model.

The problems of using the Kneser-Ney smoothing during model training can be circumvented if the histories of order n are clustered only when it is known which $(n+1)$-grams are included in the model. This approach was tested later together with the revised Kneser pruning, and as mentioned in Publication I, was found to improve the results further.

4.5.2 Clustering algorithm

If the number of histories in the model is M and the number of clusters is K, there are K^{M-K} ways to divide the histories into clusters assuming at least one history per cluster. Evidently, brute force search is not possible. The algorithm proposed in Publication II uses a greedy, incremental search. It resembles the varigram growing algorithms that add one distribution at a time to the model (Niesler and Woodland, 1999; Siu and Ostendorf, 2000). The difference is that there are now more options for each new $\boldsymbol{h}$: leave it out, create a new cluster for it, or merge it with one of the existing clusters.

The selection criterion is based on maximizing the posterior probability $p(\boldsymbol{\theta} \mid \boldsymbol{D})$ $\propto p(\boldsymbol{\theta}) p(\boldsymbol{D} \mid \boldsymbol{\theta})$. Similarly to the pruning and growing algorithms, the change in likelihood is approximated by including only the likelihood of the directly modified distributions. These are all distributions in the candidate cluster c and the new $\boldsymbol{h}$. Their part of the log-likelihood is

$$\mathit{logprob}_{\boldsymbol{h},c} = \sum_w c(\boldsymbol{h}w) \log p_{\mathrm{ACC}}(w \mid \pi(\boldsymbol{h})) + \sum_{\boldsymbol{g} \in c} \sum_w c(\boldsymbol{g}w) \log p_{\mathrm{ACC}}(w \mid \pi(\boldsymbol{g})). \tag{4.71}$$

The prior $p(\boldsymbol{\theta})$ should give a smaller probability for more complex models. The exact prior is described in the next section.

For a large number of clusters Π, testing each possible target cluster is too slow, even though it is done only once per history. For example, the log-likelihood above would ultimately be calculated over all histories in the model. This problem is avoided by testing only the cluster $c_{\min}$ that has the most similar maximum-likelihood distribution $p_{\mathrm{ML}}(W \mid c)$ to the observed distribution $p_{\mathrm{ML}}(W \mid \boldsymbol{h})$. The similarity is measured by the information radius (Equation 2.36, page 43).

The training algorithm is outlined in Figure 4.12. The parameter $\alpha \geq 0$ modifies the effect of the prior: larger models will be built with smaller α. The function MERGECLUSTERS(c_1, c_2) simply moves all histories in c_1 to c_2, modifies the counts $b(c_2, w)$ accordingly, and removes c_1. For lexicon size V and M added histories, the time complexity of the algorithm is $O(M^2 V)$. However, using the information radius to preselect the candidate cluster improves the practical efficiency. As the distributions are sparse and the summation over V can be stopped whenever the current minimum IRad is reached, V calculations are rarely necessary.

BUILDCLUSTEREDVARIGRAM(α)
1 $k \leftarrow 1$
2 **while** new histories are added **do**
3 GROWANDCLUSTERORDER(k, α)
4 $k \leftarrow k + 1$
5 re-estimate cluster distributions and discount parameters

GROWANDCLUSTERORDER(k, α)
1 **for** $\{\boldsymbol{h} : |\boldsymbol{h}| = k - 1 \wedge \exists\, \boldsymbol{g} \in \Omega, v \in \Sigma : \boldsymbol{h} = \boldsymbol{g}v \vee \boldsymbol{h} = v\boldsymbol{g}\}$ **do**
2 $c_{\min} \leftarrow \arg\min_{c \in \Pi} \mathrm{IRad}\left(p_{\mathrm{ML}}(W \mid \boldsymbol{h}) \,||\, p_{\mathrm{ML}}(W \mid c)\right)$
3 $logprob_0 = \alpha \log_2 p(\boldsymbol{\theta}) + \sum_w c(\boldsymbol{h}w) \log_2 p_{\mathrm{ACC}}(w \mid \pi(\boldsymbol{h}))$
4 $+ \sum_{\boldsymbol{g} \in c_{\min}} \sum_w c(\boldsymbol{g}w) \log_2 p_{\mathrm{ACC}}(w \mid \pi(\boldsymbol{g}))$
5 $c_{\mathrm{new}} \leftarrow$ new cluster
6 $\pi(\boldsymbol{h}) \leftarrow c_{\mathrm{new}}$
7 $logprob_{\mathrm{new}} = \alpha \log_2 p(\boldsymbol{\theta}) + \sum_w c(\boldsymbol{h}w) \log_2 p_{\mathrm{ACC}}(w \mid \pi(\boldsymbol{h}))$
8 $+ \sum_{\boldsymbol{g} \in c_{\min}} \sum_w c(\boldsymbol{g}w) \log_2 p_{\mathrm{ACC}}(w \mid \pi(\boldsymbol{g}))$
9 MERGECLUSTERS($c_{\mathrm{new}}, c_{\min}$)
10 $logprob_{\mathrm{merge}} = \alpha \log_2 p(\boldsymbol{\theta}) + \sum_{\boldsymbol{g} \in c_{\min}} \sum_w c(\boldsymbol{g}w) \log_2 p_{\mathrm{ACC}}(w \mid \pi(\boldsymbol{g}))$
11 **if** $logprob_{\mathrm{merge}} < logprob_{\mathrm{new}}$
12 undo MERGECLUSTERS($c_{\mathrm{new}}, c_{\min}$)
13 **if** $logprob_{\mathrm{new}} < logprob_0$
14 remove c_{new}
15 **else**
16 **if** $logprob_{\mathrm{merge}} < logprob_0$
17 undo MERGECLUSTERS($c_{\mathrm{new}}, c_{\min}$)
18 remove c_{new}

Figure 4.12. Algorithm of Publication II for training an ACC model.

The alternative approach that first grows an order and then clusters the histories of the previous order, is outlined in Figure 4.13. GROWORDER is shown in Figure 4.9. Of course, *sizecost* on line 8 should now be estimated from the model prior $p(\boldsymbol{\theta})$. CLUSTERORDER is similar to GROWANDCLUSTERORDER but does not have the option to remove the history. The algorithm also includes pruning of individual n-grams from the cluster distributions $p(W \mid \mathrm{c})$ (PRUNECLUSTER). Pruning affects only the cluster distributions. That is, the counts of individual histories are not updated.

4.5.3 Model prior

Publication II does not provide the details of the model prior $p(\boldsymbol{\theta})$. This section describes the exact form of the prior that was applied in the experiments.

The parameters $\boldsymbol{\theta}$ of the model include the histories Ω, clusters Π, and frequency distributions of the clusters $\mathbf{B}$, where $b_{ij} = b(\mathrm{c}_i, w_j)$. Let $V = |\Sigma|$, $M = |\Omega|$, $K = |\Pi|$, and N be the number of units in the training data. The prior for the parameters is defined using an MDL-inspired coding scheme. First, the number of histories is encoded using the universal prior by Rissanen (1983):

$$p(M) \approx 2^{-\log_2 2.865 - \log_2 M - \log_2 \log_2 M - \ldots}. \tag{4.72}$$

Each history is one of the existing histories preceded or followed (two options) by a single unit (V options). Starting from the shortest history and proceeding

```
BUILDCLUSTEREDVARIGRAM(α, ε)
 1  k ← 1
 2  while new histories are added do
 3        GROWORDER(k, α)
 4        if k > 1
 5           CLUSTERORDER(k − 1, α)
 6        k ← k + 1
 7  CLUSTERORDER(k, α)
 8  for c in model do
 9     PRUNECLUSTER(c, ε)
10  re-estimate discount parameters
```

Figure 4.13. Outline of an algorithm for growing, clustering, and pruning an ACC model.

to longer ones,

$$p(\Omega \mid M) = \prod_{i=1}^{M} \frac{1}{2Vi}. \tag{4.73}$$

The number of clusters is limited by the number of histories. Thus $p(K \mid M) = 1/M$. The histories are divided into K clusters, so there are K^M possible combinations. Giving equal probability to each combination,

$$p(\Pi \mid M, K) = K^{-M}. \tag{4.74}$$

Finally, a prior for $\mathbf{B}$ is has to be specified. The frequency distribution of each cluster is encoded independently and only non-zero entries are stored. First, $s(c)$ and $v(c)$ are selected uniformly from N and V options, respectively. There are $\frac{V!}{(V-v(c))!}$ ways to select the units with non-zero counts, and $v(c)$ positive integers that sum up to $s(c)$ can combined in $\binom{s(c)-1}{v(c)-1}$ ways. Thus

$$P(\mathbf{B} \mid \Omega, \Pi) = \prod_{c \in \Pi} \left[NV \frac{V!}{(V - v(c))!} \binom{s(c) - 1}{v(c) - 1} \right]^{-1}. \tag{4.75}$$

Evidently, the described encoding of the parameters is not the best possible. For example, the constraints that each cluster should have at least one history or that the counts in $\mathbf{B}$ should sum up to N are not applied. However, the defectiveness of the prior is not a problem, as its goal is only to prevent overfitting, to not provide an optimal description length. The prior is in any case modified by the weight parameter α.

4.5.4 Experiments

The proposed ACC model was tested on Finnish data using both cross-entropy evaluations and in a speech recognition task. The training data contained several books and magazines, 8 600 000 words in total. A morph lexicon of 2 113 units was estimated from the training data using Morfessor Baseline (Creutz and Lagus, 2005b). A few language models were trained on a larger set of 150 million words (mostly newspaper text) for reference. The speech recognizer is described by Kurimo et al. (2006b). The ASR test data was an audio book read by one female speaker. Of the total 13 h of speech, an acoustic model was trained using the first 11 hours, and 2 hours were used as test data. The cross-entropy evaluations were performed on text data from the same book (50 000 words) and

another data set including one year of articles from a tabloid magazine (100 000 words).

The proposed ACC model was compared with two baselines: n-gram models ($n = 1, 2, 3$) trained with modified Kneser-Ney interpolation ("baseline"), and varigram models built with the growing algorithm by Siivola and Pellom (2005) ("growing"). Pruning was not applied to any of the models. Cross-entropies and word error rates were compared with the sizes of the models. The sizes were approximated by the number of the n-grams (for baseline and growing) or sum of the number of histories and numbers of the units predicted directly by each cluster (for ACC).

The results are shown in Figure 4.14. The morph-based cross-entropies, in Figure 4.14(a-b), are very similar for the growing and ACC ("clustered" in the figures) models of similar size. Baseline 2-gram and 3-gram models are also close to the varigram models of the similar size. The larger data set actually gives larger entropies for the growing model than the smaller data set—the domain of the smaller data set is evidently more similar to the test sets. The ASR results in Figure 4.14(c) show that for the models of about 200 000 parameters, both varigram models yield much smaller error rates than the baseline 2-gram (over 10% absolute). For both 200 000 and 5 000 000 parameters, the ACC model outperforms the standard varigram model by about 2% absolute. Statistical significances of the differences were not studied.

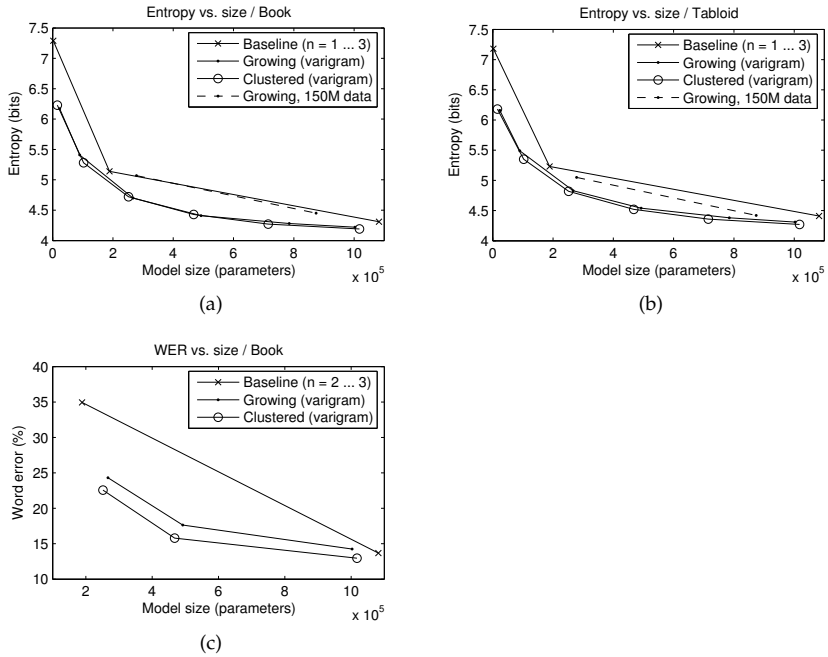


Figure 4.14. Comparison of the baseline n-grams, growing algorithm by Siivola and Pellom (2005), and incremental clustering algorithm in Publication II in (a) cross-entropy on a Finnish book data, (b) cross-entropy on a Finnish tabloid data, and (b) word error rate in a Finnish speech recognition task.

In the preliminary experiments mentioned in Publication I, revised Kneser pruning was applied to the ACC model. The ACC models trained with the new "first-grow-then-cluster" algorithm (Figure 4.13) were compared with the original ACC model, standard Kneser-Ney smoothed n-grams, and varigram

models trained with the Kneser-Ney growing algorithm of Publication I. This time, models up to 3 500 000 parameters were trained, although the ACC models were limited to two million parameters due to long training times. Also pruning was applied: RKP for the varigram models and entropy-based pruning (EP) for the baseline models. The results on the tabloid test data are shown in Figure 4.15. The alternative algorithm for the ACC models outperforms the original one even without pruning (solid lines). With pruning (dashed lines), the difference is much larger. For the smallest models, ACC has a slightly lower entropy than KNG, but the difference may not be statistically significant. Moreover, the differences between ACC and KNG even out already with two million parameters.

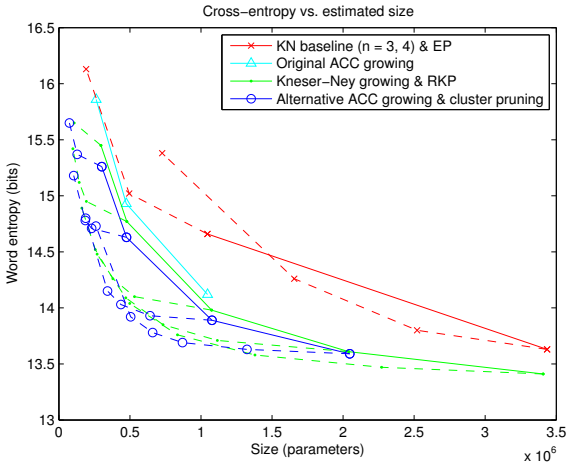


Figure 4.15. Comparison of the ACC models, baseline 3-gram and 4-gram models, and KNG models. The points connected with solid lines indicate the unpruned models with varying model order or growing threshold. Each of them was pruned with varying pruning threshold, as indicated by the dashed lines. For example, the baseline 3-gram and 4-gram models are connected with a solid red line, and dashed red lines show how EP pruning affects the two models.

4.5.5 Discussion

There are only few prior studies for the n-gram context clustering approach of Publication II. The closest related work is probably that of Siu and Ostendorf (2000), who consider a conversational speech recognition task with varigram models. As mentioned in Section 4.2.4, Siu and Ostendorf (2000) use a suffix tree to store the history nodes. They consider two types of restricted cases of context clustering. In the first case, the node $\boldsymbol{h}$ may be combined with $\boldsymbol{h}_{-j}$, where the j^{th} word of $\boldsymbol{h}$ is removed (skipped). This helps in modeling the fillers (e.g. **uh**, **um**) and repetitions in conversational speech. In the second case, the sibling nodes $v\boldsymbol{h}$ and $w\boldsymbol{h}$ may be combined. Here, in addition to combining the predictive distributions, all subtrees of the nodes are merged. For example, merging **sort of** and **kind of** will also merge the histories **was sort of** and **was kind of**. For the context clustering, they obtain 8% reductions in parameter size with insignificant increase in perplexity and word error rate.

Another approach that resembles the n-gram context clustering is decision tree (DT) language modeling (Bahl et al., 1989; Potamianos and Jelinek, 1998; Xu and

Jelinek, 2004). For the language modeling task, decision trees cluster the histories into equivalence classes by making questions on the elements in the history. In contrast to the above methods, clustering is divisive. The questions usually consider identity of the word v_i in the i^{th} position of $\boldsymbol{h}$. For binary trees, there are up to $2^V - 1$ possible way to split the histories for each i, which means that only greedy search is possible. While divisive clustering increases the computational complexity of the task, DT models are more flexible than the n-gram models. To prevent overlearning, Xu and Jelinek (2004) uses partially random splits and train a collection of decision trees, called *random forest*. They also use Kneser-Ney smoothing and back-off distributions for the distributions in the leaf nodes. While a single 3-gram DTs still have higher entropy than standard KN smoothed 3-gram, Xu and Jelinek (2004) show that the linear interpolation over the trees in the random forest yields considerable improvements both in terms of perplexity and word error rate of an ASR task.

While not as flexible as the DT models in terms of forming the equivalence classes, the accumulative context clustering proposed in Publication II is still a very general approach: it can be applied to any model that collects predictive distributions $p(W \mid \boldsymbol{h})$ for a large number of contexts $\boldsymbol{h}$, including skip n-gram models and standard word cluster models. In fact, word clustering should make ACC easier to apply, as the lexicon size is reduced. Different context clusterings could be used in different components of a model. For example,

$$\begin{aligned} p(w \mid \boldsymbol{h}) &= \sum_t p(t \mid \boldsymbol{h}) p(w \mid t, \boldsymbol{h}) \\ &\approx \sum_t p(t \mid \pi_1(\boldsymbol{h})) p(w \mid t, \pi_2(\boldsymbol{h})) \end{aligned} \tag{4.76}$$

applies one ACC model to predict the next word cluster t and another to predict the word among the words in t.

The results of the experiments show that the ACC model can improve both perplexity and word error rate over non-clustered varigram models, without interpolation with standard n-gram models. However, the advantage seems to hold only for very small model sizes. Moreover, computational complexity of the algorithm makes it hard to use large training data sets.

A possible extension for the ACC model would be to use soft clustering of the histories. With K clusters, the the conditional probability would be

$$p(w \mid \boldsymbol{h}) = \sum_{i=1}^{K} p(w \mid c_i) p(c_i \mid \boldsymbol{h}). \tag{4.77}$$

The corresponding graphical model for $n = 4$ is shown in Figure 4.16. In fact, it is an extension of the aggregate Markov model by Saul and Pereira (1997) (cf. Figure 4.6(a), page 109). While Saul and Pereira (1997) as well as Blitzer et al. (2005) extend the order of their models by having separate hidden variable(s) for every word in the history, the context cluster model predicts a single hidden variable based on all of them.

For any clustering of the histories it is essential that the distributions $p(W \mid \boldsymbol{h})$ are not too sparse on average: otherwise the clustering would not be able to reduce the number of parameters. The sparsity can be defined by $1 - t(\boldsymbol{h}\bullet)/V$. Figure 4.17 shows the proportion of n-grams $\boldsymbol{h}w$ in histories ordered by $t(\boldsymbol{h}\bullet)$ in a morph-based varigram model trained with the Kneser-Ney growing algorithm (Section 4.4.2). The histogram shows that most of the n-grams are indeed in very sparse distributions. The median value of $t(\boldsymbol{h}\bullet)$ is 31. The graph of cumulative

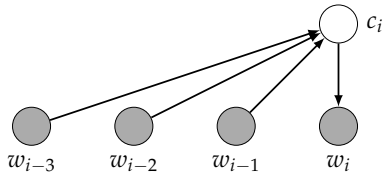


Figure 4.16. Graphical representation for the context clustering model in Publication II.

proportion indicates that about 25% of the n-grams are in distributions with over 200 non-zero entries. This is certainly enough for the context clustering to be feasible—as confirmed by the experiments above—but also indicates that large improvements are not likely even with a more sophisticated clustering method.

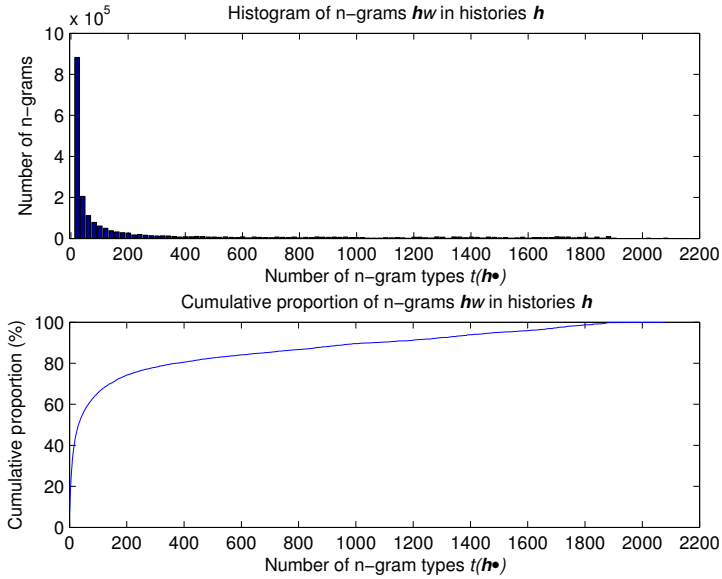


Figure 4.17. The proportion of n-grams in histories of different distribution sparsity for a morph-based varigram model of 2 million n-grams. The vocabulary size is 2080. Top: histogram of the number of n-grams. Bottom: cumulative proportion of the n-grams.

5. Representation learning

In this chapter, the problem of representation learning, highlighted in Figure 5.1, is considered in more detail. The first two sections review the standard methods used for linguistic data: vector space models (Section 5.1) and probabilistic topic models (Section 5.2). Section 5.3 discusses various ways for evaluating representation learning methods. Finally, Section 5.4 describes the main contribution of this thesis for the representation learning problem: the novel evaluation method proposed in Publication III.

Consider the random variable S representing fragments of a language and a set of samples $s_1, s_2, \ldots, s_N$ from S. There are several NLP applications in which the similarities between the given samples are important, but the generation of the individual samples is not relevant. A typical example is an information retrieval (IR) task, where the search engine should return those of the known documents s_i that are closest to the given search query q. If the similarity function $\mathrm{sim}(x, y)$ directly used the string representations of the texts, for example by calculating the string edit distance, the computational complexity would be too high for any large-scale application. In other tasks, the samples may also be words, in which case there is no way to calculate similarities based directly on the strings. Thus there is a need for efficient representations that encode the semantic similarities between the samples.

In information retrieval systems (see, e.g., Manning et al., 2008; Blanken and Hiemstra, 2007), documents are indexed by *terms* that describe the content of the documents. The index may be filled manually by specialists, sometimes from a catalog that is a prescribed list of terms. On large-scale and open domain systems, however, terms are usually defined more or less automatically. Usually they are the words (or stems) that occur in the documents. Function words are often excluded using a manually collected *stop word list*.

The traditional boolean models of IR are based on the logic operators AND, OR, and NOT. Given a query, such as **"learning AND unsupervised AND NOT clustering"**, the system returns the set of all documents for which the boolean expression evaluates true. Often the models are extended with proximity searching of terms that are adjacent or nearby in the document, and wildcards that are used to match the terms only partially. While they are still widely used, the boolean models have problems with such common lexical relations as synonymy and homonymy. Optimally, the system should know that **car** and **automobile** have the same or at least very similar meaning. Representation learning is useful for finding such relations automatically from large text corpora.

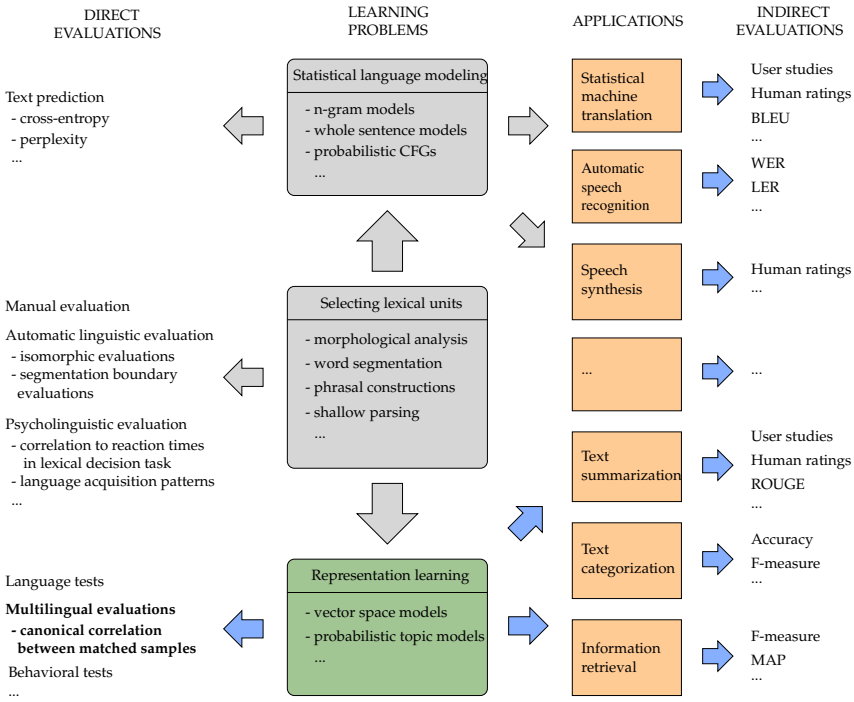


Figure 5.1. Representation learning in the overview diagram.

5.1 Vector space models

Vector space models are a standard way to represent documents or words as numerical vectors of features. Thus they provide a solution to the problem of representing symbolic information in numerical form for computational processing. In a vector space, similar items are close to each other, and the closeness can be measured using vector similarity measures.

Vector space models can be characterized by a feature generator $\mathfrak{F}(\cdot;\cdot)$ and similarity function $\mathrm{sim}(\cdot,\cdot)$. Trained with a data set $\boldsymbol{S}$, the feature generator provides a vector $\mathbf{x}_i = \mathfrak{F}(s_i;\boldsymbol{S})$ for each sample s_i. The similarity function returns a scalar similarity value $\mathrm{sim}(\mathbf{x}_i,\mathbf{x}_j)$ for two vectors $\mathbf{x}_i$ and $\mathbf{x}_j$.

Let $\boldsymbol{S}$ be a set of N documents, V the number of different word types in $\boldsymbol{S}$ and $c(i,j)$ the number of occurrences of word w_i in document s_j. A common vector space model sets the representation of document s_j to

$$\mathbf{x}_j = \mathfrak{F}(s_j;\boldsymbol{S}) = \Big(c(1,j),\, c(2,j),\, \ldots,\, c(V,j)\Big)^{\mathrm{T}} \in \mathbb{R}^V. \tag{5.1}$$

In this kind of representations, the word order information is discarded, and hence they are called *bag-of-words* representations. The bag-of-words approach is trivial to extend to other lexical units, such as morphemes, index terms, or phrases. The bag-of-words and related representations are useful for encoding topics of long documents, but poor for determining meaning of individual sentences, where a single word such as the negation **"not"** may change the meaning to its opposite.

The matrix $\mathbf{X} = \mathfrak{F}(\boldsymbol{S};\boldsymbol{S}) \in \mathbb{R}^{V\times N}$ that consists of the vectors $\mathbf{x}_j$ is called a *word-document matrix*. At the same time as the column vectors of $\mathbf{X}$ describe

documents as bags-of-words, the column vectors of the document-word matrix $\mathbf{X}^T$ describe words in terms of the documents in which they occur. For word representations, it is common to also use smaller contexts than documents, such as sentences or fixed-width windows in running text. Accordingly, $\mathbf{X}$ can more generally be called a *feature-context* matrix.

The word-document matrix contains the occurrences of the words in their contexts and thus represents *first-order similarity*. In contrast, *second-order similarities* can be observed by collecting a word-word matrix, where the values are co-occurrences of words within some contexts, or a document-document matrix, where the values define how many common features the documents have. The second-order matrices can be obtained, for example, by computing $\mathbf{X}\mathbf{X}^T$ for a word-word matrix or $\mathbf{X}^T\mathbf{X}$ for a document-document matrix. Using short context windows, first-order similarities often provide syntagmatic associations and second order similarities paradigmatic associations (Rapp, 2002; see also Section 3.1.3).

The most common similarity function for vector space models is the cosine similarity, which is equal to the cosine of the angle between the vectors:

$$\cos(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x}^T\mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|}. \tag{5.2}$$

Thus cosine similarity does not depend on the lengths of the vectors. If $\|\mathbf{x}\| = \|\mathbf{y}\| = 1$, the cosine similarity has a linear relation to the squared Euclidean distance: $\|\mathbf{x} - \mathbf{y}\|^2 = \|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 - 2\mathbf{x}^T\mathbf{y} = 2(1 - \cos(\mathbf{x}, \mathbf{y}))$.

5.1.1 Weighting

Plain word-document co-occurrence data emphasizes the most frequent words in the document collection. This is in contrast to intuition: a rare word like **ecotourism** should clearly be more informative than a common word like **agreement**, not vice versa. To improve the performance of the models, different weighting schemes are commonly utilized. They aim to give more weight to such index terms that better represent the semantic content of the document.

The schemes can be divided into *global* and *local* weighting schemes. Global weights indicate the overall importance of a term in the collection and they are applied to each term in all the documents, whereas local weights are applied to each term in one document. The final weight is usually the product of the global and local weights.

5.1.1.1 Local weighting

The frequency of the term in a document (*term frequency*, tf) indicates the saliency of a term. However, the effect of a raw count $c(i, j)$ is often too large. One option is simply to discard the frequencies and use binary local weights. However, a more popular approach is to dampen the term frequency with a logarithmic function $f(x) = 1 + \log x\,(x > 0)$ or $f(x) = \log(1 + x)$. This is called logarithmic term frequency (log-tf).

5.1.1.2 Global weighting

Global weighting is used for emphasizing those terms that occur only in few documents and de-emphasizing those that occur in most of the documents.

Table 5.1. Some global weightings for index terms. N is the number of documents in a document collection, $c(i,j)$ is the frequency of term i in document j, $g(i) = \sum_j c(i,j)$ is the global frequency of term i in the whole collection, $d(i) = |\{j : c(i,j) > 0\}|$ is the document frequency for term i, and ${\sigma_i}^2$ is the sample variance for the frequencies of term i.

Weighting	Coefficient for feature i
Logarithmic idf	$\log \frac{N}{d(i)}$
Entropy weighting	$1 - \sum_j \frac{p_{ij} \log p_{ij}}{\log N}$, where $p_{ij} = \frac{c(i,j)}{g(i)}$
Variance normalization	${\sigma_i}^{-1} = (\frac{1}{N-1} \sum_j (c(i,j) - \frac{g(i)}{N})^2)^{-\frac{1}{2}}$

Common global weighting schemes are presented in Table 5.1. The most popular global weighting scheme, *inverse document frequency* (idf), assigns a high weight to terms that occur only in few documents and thus refer to very specific concepts. For term i, idf is the total number of documents N divided by the number of documents the term i occurs in. To dampen the effect of the weight, usually logarithmic idf (log-idf) is applied (Spärck Jones, 1972). *Entropy weighting* assigns the minimum weight to terms for which the distribution over documents is close to uniform and the maximum weight to terms which are concentrated in a few documents (Dumais, 1991). Finally, one can simply normalize variances of the features to one, as it is often done for continuous features.

5.1.2 Length normalization

The length of the obtained vectors varies across the documents. The length depends both on how many words there are in each document and on the applied local and global weightings. The similarities between the documents are often calculated with cosine similarity measure, which neglects the vector lengths (Salton and Buckley, 1988). However, if some other distance measure is applied, the vectors can be explicitly normalized using, for example, ℓ_1 or ℓ_2 norms.

5.1.3 Dimensionality reduction

The dimensionality of a feature-context matrix $\mathbf{X} \in \mathbb{R}^{V \times N}$ may be very high due to a large number of features M (e.g., words or index terms) or a large number of contexts N (e.g., documents, sentences, or neighboring words). To reduce the computational cost of calculating the similarities in the vector space, it is common to use dimensionality reduction (see Section 2.5.2). The methods for reducing the dimensionality can be divided into two families of approaches: feature selection and feature extraction (Schütze et al., 1995; Alpaydin, 2004, Ch. 6). A comprehensive review on different feature selection and extraction methods for vector space models is given by Sebastiani (2002).

5.1.3.1 Feature selection

In feature selection, the task is to choose K dimensions out of the V original dimensions that give as much information as possible. The rest $V - K$ dimensions are discarded. If it is computationally feasible to evaluate a large number of subsets of features, feature selection can be done systematically with subset selection methods (Alpaydin, 2004, p. 106). In vector space models, it is common to apply heuristic preprocessing, such as stemming, exclusion of too frequent and too rare words, or removal of non-alphabet characters, although

more sophisticated methods have also been applied (Sebastiani, 2002). In the global framework discussed in this thesis, feature selection can be considered to be part of the unit selection problem.

5.1.3.2 Feature extraction

In feature extraction, or reparametrization, the task is to find a new set of K dimensions that are combinations of the original V dimensions. Usually, this is accomplished by finding a linear projection $\widehat{\mathbf{X}} = \mathbf{R}\mathbf{X}$, where $\mathbf{R} \in \mathbb{R}^{K \times V}$ is a projection matrix.

A computationally light way to reduce the dimensionality is to project the data with random vectors that are nearly orthogonal. If the randomly selected subspace has a sufficiently high dimension, the distances between the data points are approximately preserved (Johnson and Lindenstrauss, 1984). This approach has been addressed by several names: random projection (Ritter and Kohonen, 1989), random mapping (Kaski, 1998), and random indexing (Kanerva et al., 2000).

Random projection is computationally efficient especially with sparse projection matrices (Achlioptas, 2001; Bingham and Mannila, 2001). However, if also the original data is sparse—as in the case of bag-of-words—a sparse projection matrix will distort the data. This can be prevented by the Fourier transform of the data (Ailon and Chazelle, 2006). A recent paradigm that exploit sparsity and random projection matrices is *compressed sensing* or *sampling* (Donoho, 2006; Candès et al., 2006; Candès and Wakin, 2008). However, it seems that these new findings are not yet tested on text data.

Principal components analysis (PCA, see Section 2.8.2) can be used to find K features that encode the maximal amount of variance in the original data set $\mathbf{X}$. A simple way to calculate PCA is to use singular value decomposition (SVD, Section 2.8.1) on a centered data set. As centering removes the inherent sparsity in text data, it is common to use SVD directly on $\mathbf{X}$. The difference between SVD and PCA is often minor because the mean is in any case close to zero because of the data sparsity. SVD can also be motivated directly by the optimality of the solution in the least-squares sense (see, e.g., Manning and Schütze, 1999, p. 559). The use of SVD on text document data dates back to Benzécri (1973), but was made widely known by the names of *latent semantic analysis* (LSA) and *latent semantic indexing* (LSI) (Deerwester et al., 1990).

SVD, as well as probabilistic topic models discussed later in Section 5.2, exploit second-order statistics and generalize the data besides reducing the dimensionality. For instance, the latent space found for documents using LSA often combines the individual terms into more general topics. The methods can also address the problems of polysemy and synonymy (Deerwester et al., 1990).

Also independent component analysis (ICA) has been used for feature extraction. For example, Isbell and Viola (1999) apply ICA for document representations and Honkela et al. (2010) for word representations. ICA uses higher-order statistics to find components that are as independent as possible. Typically, this provides sparse components (Hyvärinen et al., 2001). Compared with LSA, Honkela et al. (2010) find that the ICA components better reflect linguistic categories and are easier to interpret by humans.

The self-organizing map (SOM) performs non-linear dimensionality reduction to two dimensions. While very precise similarities cannot be stored by using only two dimensions, the result is very suitable for visualization and exploration

of the similarities of words (Honkela et al., 1995) or documents (Kaski et al., 1998; Kohonen et al., 2000; Lagus et al., 2004).

5.2 Probabilistic topic models

An alternative to the heuristics used in vector space models is to create a probabilistic model for generation of the documents. The first approaches to this direction (e.g. Bookstein and Swanson, 1974) were based on the Poisson distribution:

$$p(k \mid \lambda_i) = e^{-\lambda_i} \frac{{\lambda_i}^k}{k!}, \tag{5.3}$$

where k is the number of occurrences for an event and λ_i is the single parameter of the distribution. By setting $\lambda_i = g(i)/N$, the distribution can be used to estimate the probability of k occurrences of a word w_i in a document. The Poisson distribution assumes that occurrences are independent of the documents. In consequence, most function words follow the Poisson distribution quite well, while many content words do not, as they tend to be concentrated on certain documents. This observation has provided some approaches for term weighting (Manning and Schütze, 1999, Sec. 15.3).

More recent probabilistic models for documents are based on the categorical or multinomial distribution. Ponte and Croft (1998) were the first to propose a statistical language modeling approach to IR: Each document d_j is presented by a categorical distribution $p(W = w_i \mid \boldsymbol{\pi}_j)$ (see Section 2.1), also called a *unigram* model. The relevance of the document to the query $\boldsymbol{q}$ is then obtained by the probability that the document model would generate the query:

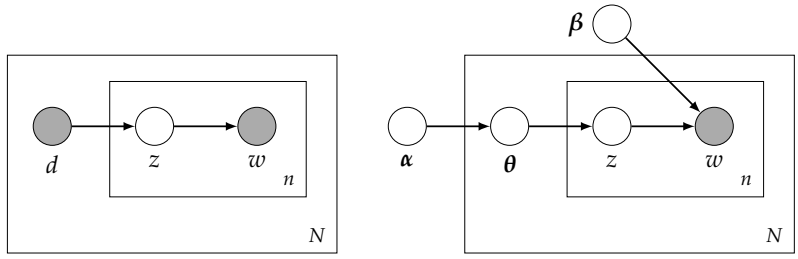
$$p(\boldsymbol{q} \mid \boldsymbol{\pi}_j) = \prod_{w \in \boldsymbol{q}} p(w \mid \boldsymbol{\pi}_j). \tag{5.4}$$

Ponte and Croft (1998) used ML estimates $\pi_{ij} = c(i,j)/\sum_i c(i,j)$ for the document distributions, but replaced zero probabilities with collection-wide estimates $g(i)/\sum_i g(i)$. An alternative way is to interpolate document models with a background model estimated from the collection, resulting in a global weighting effect similar to idf (Westerweld et al., 2007).

Assuming a separate unigram model for each document requires a large number of parameters: $N \times V$ for N documents and V index terms. Evidently, for a large enough collection, the number of topics in the documents is smaller than the number of documents. Thus it is useful to assume that the documents are generated by *latent topics* $z \in \mathrm{Z}$. Assuming a single topic per document gives a *mixture of unigrams* model (Nigam et al., 2000). For a mixture of unigrams, the probability of a document that consists of words $\boldsymbol{w} = w_1 \ldots w_n$ is

$$p(\boldsymbol{w}) = \sum_{z \in \mathrm{Z}} p(z) p(\boldsymbol{w} \mid z) = \sum_{z \in \mathrm{Z}} p(z) \prod_{i=1}^{n} p(w_i \mid z). \tag{5.5}$$

The parameters of the model—topic priors and unigram parameters—can be estimated with the EM algorithm. Nigam et al. (2000) use the model for semi-supervised text classification. In the basic approach, each z is a class label in the labeled training data, and classes of the unlabeled documents are assigned by EM. They also extend the model to the case of multiple latent topics per class.



(a) PLSA (b) LDA

Figure 5.2. Graphical representations for PLSA and LDA topic models. N is the number of documents and n is the number of words generated for a document.

In an unsupervised setting, a reasonable assumption is that each document d is generated by a weighted sum of latent topics z, and documents and words are independent when conditioned on z (Hofmann, 1999a, 2001). The joint distribution of documents and words can then be written as

$$p(d, \boldsymbol{w}) = \sum_{z \in \mathrm{Z}} p(z) p(d \mid z) p(\boldsymbol{w} \mid z). \tag{5.6}$$

Let $K = |\mathrm{Z}|$ and collect the above probabilities to matrices $\mathbf{P} \in \mathbb{R}^{V \times N}$, $\mathbf{U} \in \mathbb{R}^{V \times K}$, $\mathbf{V} \in \mathbb{R}^{N \times K}$, and $\mathbf{D} \in \mathbb{R}^{K \times K}$, such that $p_{ij} = p(w_i, d_j)$, $u_{ik} = p(w_i \mid z_k)$, $v_{jk} = p(d_j \mid z_k)$, and $d_{kl} = \delta_{kl} \times p(z_k)$, where δ_{jk} is the Kronecker delta function (i.e., $\delta_{ij} = 1$ if $i = j$ and otherwise zero). Now the model can be written as a non-negative matrix factorization by $\mathbf{P} = \mathbf{U}\mathbf{D}\mathbf{V}^{\mathrm{T}}$. Due to the evident similarity to LSA, Hofmann (2001) calls this *probabilistic latent semantic analysis* (PLSA). The parameters of the model can be estimated with the EM algorithm. Hofmann (1999b) shows that it outperforms LSA in information retrieval tasks.

PLSA represents each document d_j by the mixing weights $p(z_k \mid d_j)$ of the topics. In consequence, the number of parameters grows linearly with the number of documents. Moreover, while the mixing weights for documents outside the training set can be determined by so-called *folding* (Hofmann, 1999b), it is not clear how to assign a probability to such a document. To improve on these issues, Blei et al. (2003) have proposed a three-level hierarchical Bayesian model called *latent Dirichlet allocation* (LDA). In LDA, a global parameter $\boldsymbol{\beta}$ defines the unigram distributions of the topics z_k. Similarly to PLSA, each word in a document is generated by a randomly chosen topic. In LDA, this topic is drawn from $p(z_k \mid \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is drawn once per document from a Dirichlet distribution (Section 2.1) with parameter $\boldsymbol{\alpha}$. Figure 5.2 shows graphical representations for PLSA and LDA models. Blei et al. (2003) show that both a mixture of unigrams and PLSA suffer from overfitting avoided by LDA.

Both PLSA and LDA models can be used for representation learning by setting the conditional topic distributions $p(z_k \mid d_j)$ or $p(z_k \mid \theta_j)$ as new features for the documents. A benefit compared with LSA is that the features have a clear interpretation, as each z_k corresponds to a word distribution $p(w_i \mid z_k)$.

A drawback of the probabilistic models is the computational complexity of the training. The word-document matrix is a sparse $V \times N$ matrix; let the average number of non-zero entries per column be $c < V$ and the target dimensionality K. According to Papadimitriou et al. (2000), full SVD can be computed in $O(VcN)$ and random projection in $O(KcN)$ operations. However, truncated SVD can be approximated significantly faster (Menon and Elkan, 2011). Accord-

ing to Hofmann (1999b), PLSA requires 40–60 iterations of complexity $O(KcN)$ for K latent topics. Finally, the variational inference algorithm used by Blei et al. (2003) for LDA requires around V iterations of complexity $O(KV)$, which is usually worse than the complexity of the other methods. For example, if the vocabulary size was $V = 10^5$, there were $N = 10^3$ documents that on average have $c = 10^3$ word types, $K = 10^2$, and the number of iterations for PLSA and LDA taken into account, the number of operations would be 10^8 for random projection, from 10^9 to 10^{10} for PLSA, and 10^{12} for LDA. Full SVD would require 10^{11} operations, but Menon and Elkan (2011) suggest that truncated SVD may be closer to the complexity of the random projection, even without approximative methods.

5.3 Evaluation of representation learning

Using the general division in this thesis, the evaluations for representation learning methods include application-oriented indirect evaluations and application independent direct evaluations. The choice of the evaluation depends on the type of linguistic units for which the representations are obtained. Typically the units are either full documents or individual word types, but they can as well be single sentences or even phrases. Application evaluations are more common for document representations, but include some settings for evaluating word representations. In contrast, the direct evaluations proposed so far have been exclusively for word representations.

5.3.1 Application evaluations

The most common evaluation for representation learning of documents is its most common application: information retrieval. The resources required for a typical IR evaluation are a set of documents, set of queries, and the manually determined relevance of each document for each query. Cross-language information retrieval (CLIR) is a multilingual extension to the IR framework: a query can be in a different language than the retrieved document. There are three major evaluation campaigns and conferences for IR and CLIR systems: TREC (first overview by Harman, 1992), NTCIR (Kando et al., 1999), and CLEF (Peters, 2001).

The classical IR task is called *ad-hoc retrieval*: a user enters a query that describes the desired information, and the system returns a list of documents. In *exact matching*, the documents that precisely satisfy the query are returned. In *ranked retrieval*, the documents are returned by their relevance, and the number of documents to return is fixed or selected by the user.

For evaluating exact matching, there are two sets of documents to consider: retrieved documents P and relevant documents R. The intersection of the sets, $\mathrm{P} \cap \mathrm{R}$, tells which of the retrieved documents were correct. Normalizing $|\mathrm{P} \cap \mathrm{R}|$ with the number of retrieved documents gives the *precision* (Pre) and normalizing it with the number of relevant documents gives the *recall* (Rec):

$$\mathrm{Pre} = \frac{|\mathrm{P} \cap \mathrm{R}|}{|\mathrm{P}|}; \quad \mathrm{Rec} = \frac{|\mathrm{P} \cap \mathrm{R}|}{|\mathrm{R}|}, \tag{5.7}$$

That is, precision measures the proportion of the retrieved documents that are correct, whereas recall measures the proportion of the relevant documents are

retrieved. To get a single measure that includes the aspects of both precision and recall, they are often combined using harmonic mean, resulting in *F-score* or *F-measure*:

$$\mathrm{F} = \frac{2}{\frac{1}{\mathrm{Pre}} + \frac{1}{\mathrm{Rec}}} = \frac{2 \times \mathrm{Pre} \times \mathrm{Rec}}{\mathrm{Pre} + \mathrm{Rec}} \tag{5.8}$$

A generalization of the balanced F-score is F_β-score:

$$\mathrm{F}_\beta = \frac{(1 + \beta^2) \times \mathrm{Pre} \times \mathrm{Rec}}{\beta^2 \times \mathrm{Pre} + \mathrm{Rec}}, \tag{5.9}$$

where $\beta > 1$ gives more weight to recall and $\beta < 1$ gives more weight to precision.

In ranked retrieval, the order of the retrieved documents has to be taken into account. A common measure for this is the *average precision* (AP). For calculating AP, the ranked list of retrieved documents $\boldsymbol{P} = (d_1, d_2, \ldots, d_K)$ is compared with the set of relevant documents R. Precision $\mathrm{Pre}(k)$ at rank k is the proportion of relevant documents among the k topmost documents:

$$\mathrm{Pre}(k) = \frac{1}{k} \sum_{i=1}^{k} \mathrm{I}(d_i \in \mathrm{R}). \tag{5.10}$$

AP is then the average of precisions at ranks that have a relevant document:

$$\mathrm{AP} = \frac{1}{|\mathrm{R}|} \sum_{i=1}^{K} \mathrm{I}(d_i \in \mathrm{R}) \mathrm{Pre}(i). \tag{5.11}$$

The mean average precision (MAP) is the mean of this value over a set of queries.

Apart from ad-hoc retrieval, vector representations can be used for *visualization* and *exploration* of document collections, as in the WEBSOM system (Kaski et al., 1998; Kohonen et al., 2000; Lagus et al., 2004). From the evaluation perspective, the evaluation of an explorative task is problematic as it requires user studies (see, e.g., Chen et al., 1998).

In addition, there are IR-related tasks that use annotated training data. In *text categorization* (e.g. Lewis, 1992), the documents are assigned to two or more predefined categories. The obtained categories can naturally be used as a part of an information retrieval system. *Text filtering* (e.g. Lewis and Tong, 1992) and *text routing* (e.g. Schütze et al., 1995) are categorization tasks with only two categories: relevant and non-relevant. Routing provides a ranking of the input documents according to their estimated relevance, while filtering assesses whether each given document is relevant or not. Another type of categorization task is sentiment analysis, where the goal is to determine whether documents (e.g. product reviews) are positive or negative (Turney, 2002; Pang et al., 2002).

A multilingual corpus can be used to evaluate an alignment task, where the goal is to match each text entry with its translation. The entries may be documents (e.g. Besançon and Rajman, 2002) or sentences (e.g. Tripathi et al., 2010). Both tasks have practical use for collecting and processing training data for machine translation.

In addition to document or sentence representations, there are many uses for representations of individual words. Vector space models of words have been used at least in word sense disambiguation (e.g. Schütze, 1992; Florian and Wicentowski, 2002; Lindén and Lagus, 2002), part-of-speech tagging (e.g. Schütze, 1995; Lamar et al., 2010), cross-document co-referencing (Bagga and Baldwin, 1998), bilingual lexicon extraction (Gaussier et al., 2004; Sahlgren and Karlgren, 2005), and phrase-break prediction in speech synthesis (Watts et al., 2011).

5.3.2 Direct evaluations

A direct evaluation of a vector space model is a non-trivial task. In contrast to statistical language modeling, there is no intrinsic measure of success such as cross-entropy. In contrast to statistical parsing or morphological analysis, there is no foreseeable way to manually define "linguistically correct" vectors for direct comparison.

The goal of vector space models is to find and store semantic (or otherwise relevant) similarities between the data samples. Thus any direct evaluation method has to be based on measuring the similarities and dissimilarities between the samples and comparing the results with external data. As the external data, the evaluations often utilize data either intended for human use or collected from humans. Such data sets are available only for representations of single words.

One type of evaluation data is obtained from language tests designed for humans. The language test that have been applied include Test for English as a Foreign Language (TOEFL) (Landauer and Dumais, 1997), test of English as a Second Language (ESL) (Turney, 2001) and the SAT college entrance exam (Turney, 2005). The questions typically ask for synonyms or the semantically closest alternative word for a certain word in a given sentence. Interestingly, the best automatic methods perform better than non-native speakers of English on the TOEFL test (Rapp, 2004). The drawbacks of evaluations based on a language test are that they consider only limited aspects of word similarity (e.g., synonymy) and that the amount of test data is often small.

A second type of external data is available from thesauri. Thesauri typically define synonyms, antonyms, or free association similarities for a list of head terms. Steyvers et al. (2005) has used the University of South Florida free associations by Nelson et al. (1998), and at least Curran and Moens (2002), Väyrynen et al. (2007), and Sahlgren (2006b) have used the Moby synonym data by Ward (2002). Also more structured lexical databases are available, including the currently widely used WordNet (Miller, 1995; Fellbaum, 1998) and other ontologies of different areas and languages. The number of terms in the thesauri range from a few thousand to hundreds of thousands, and thus the evaluation sets are larger than those obtained from language tests. However, availability for particular domains and languages is often limited.

A third way of evaluation is to have human evaluators judge the similarity of the items close to each other in the vector space (Mitchell and Lapata, 2008; Zesch and Gurevych, 2009). Human judgement is a sensible way of evaluation from the cognitive point of view. It can also deal with the variation among the language users given a large enough number of evaluators. However, such extensive use of human labor is expensive to arrange.

Finally, vector space models can be evaluated using data from psycholinguistic experiments of human language processing. For example, Lund and Burgess (1996) show that the semantic distances between words in a vector space correlate with human reaction times in a lexical priming study.

5.4 Vector space model evaluation using CCA

The direct and indirect evaluation methods discussed in the previous section have their own advantages and drawbacks. Direct evaluation methods often fo-

cus on a specific phenomenon, which means that the results may not generalize to real applications: the fact that a vector space encodes a particular type of semantic relation, such as synonymy, does not tell how well it encodes meaning in general (Sahlgren, 2006a). They are usually straightforward and quick to apply, but suffer from limited and expensive evaluation data. Indirect evaluations, in turn, are often time-consuming and require additional components or resources besides the vector space model. However, they evaluate exactly the aspects of VSM that are required for the application, and there are often data sets available in various domains and languages.

Publication III proposes a direct evaluation method that combines the advantages of indirect and direct methods. As a direct method, it is straightforward to apply and does not require any additional components. Although it evaluates the VSM independently from applications, it does not concentrate on a specific linguistic phenomenon and in this aspect resembles indirect evaluation methods. Moreover, compared with other direct methods, it has no serious problems of data availability. The only required resource is a parallel corpus, and such are readily available for several languages and domains.

The essential idea of the evaluation method is as follows. Recall that the standard task for a VSM is to encode the semantic information in a set of documents $\boldsymbol{S}$. Now consider a document-aligned parallel corpus: the parallel documents in two languages $\boldsymbol{S}$ and $\boldsymbol{T}$ can be seen as two views of the same underlying semantics (Mihalcea and Simard, 2005). If the evaluated VSM captures the language independent semantic intention that is common to the aligned documents, the generated features $\mathfrak{F}(s;\boldsymbol{S})$ and $\mathfrak{F}(t;\boldsymbol{T})$ should have a high dependence across the document pairs (s,t). In order to measure the dependence, the proposed evaluation method relies on canonical correlation analysis (CCA). CCA finds the maximally correlated subspaces for the two sets of features (Section 2.8.3). In the simplest case, a correlated feature may be a certain word in the first language and its translation in the second language. However, if the dimensionality of the representation is limited, each feature has to incorporate broader concepts than single words. The higher the cross-correlations are, the better are the features generated by the VSM.

The rest of this section gives an overview of the evaluation method, experiments, and discussion presented in Publication III.

5.4.1 Mathematical foundation

The proposed evaluation method assumes the bilingual document generation model illustrated in Figure 5.3. The documents s and t are generated from semantic features $\mathbf{z}_s \in \mathbb{R}^{D_s}$ and $\mathbf{z}_t \in \mathbb{R}^{D_t}$, respectively. The generating processes, denoted G_s and G_t, are stochastic and unknown but independent of each other. However, the language-dependent semantic features $\mathbf{z}_s$ and $\mathbf{z}_t$ are linearly dependent on $\mathbf{z} \in \mathbb{R}^{D_z}$. Thus, given a global meaning representation $\mathbf{z}$, documents s and t are produced as

$$s = \mathrm{G}_s(\mathbf{z}_s) = \mathrm{G}_s(\mathbf{W}_s\mathbf{z}); \qquad t = \mathrm{G}_t(\mathbf{z}_t) = \mathrm{G}_t(\mathbf{W}_t\mathbf{z}), \tag{5.12}$$

where $\mathbf{W}_s \in \mathbb{R}^{D_s \times D_z}$ and $\mathbf{W}_t \in \mathbb{R}^{D_t \times D_z}$. Moreover, s and t are conditionally independent given the meaning $\mathbf{z}$.

Next, consider a VSM with a feature generator $\mathfrak{F}$. Let us denote the generators trained with document sets $\boldsymbol{S_0}$ and $\boldsymbol{T_0}$ as $\mathfrak{F}_s(s) := \mathfrak{F}(s;\boldsymbol{S_0})$ and $\mathfrak{F}_t(t) := \mathfrak{F}(t;\boldsymbol{T_0})$, respectively. For simplicity, assume $E_S[\mathfrak{F}_s(s)] = E_T[\mathfrak{F}_t(t)] = 0$. Let $(\boldsymbol{S},\boldsymbol{T})$ be a

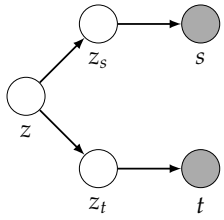


Figure 5.3. The bilingual document generation model assumed by the evaluation method proposed in Publication III.

new, aligned data set of N documents, in which each pair (s_i, t_i) is sampled from $p(s \mid \mathbf{z}_i)$ and $p(t \mid \mathbf{z}_i)$. Feature generators transform the documents s_i and t_i into vectors $\mathbf{x}_i$ and $\mathbf{y}_i$:

$$\mathbf{x}_i := \mathfrak{F}_s(s_i) \in \mathbb{R}^{D_x}; \qquad \mathbf{y}_i := \mathfrak{F}_t(t_i) \in \mathbb{R}^{D_y}. \tag{5.13}$$

Let $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N) = \mathfrak{F}_s(\boldsymbol{S})$, $\mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_N) = \mathfrak{F}_t(\boldsymbol{T})$, and $D = \min(D_x, D_y)$. Applying CCA to $\mathbf{X}$ and $\mathbf{Y}$ will provide matrices $\mathbf{A}$ and $\mathbf{B}$ that project $\mathbf{X}$ and $\mathbf{Y}$ to a common vector space as $\mathbf{U} = \mathbf{A}^{\mathrm{T}}\mathbf{X}$ and $\mathbf{V} = \mathbf{B}^{\mathrm{T}}\mathbf{Y}$, respectively, where $\mathbf{U}, \mathbf{V} \in \mathbb{R}^{D \times N}$ are orthogonal ($\mathbf{U}\mathbf{U}^{\mathrm{T}} = \mathbf{V}\mathbf{V}^{\mathrm{T}} = \mathbf{I}$) and have the highest correlations $\rho_i = \mathrm{corr}(u_i, v_i)$. The empirical correlation estimates are then in the diagonal of $\mathbf{U}\mathbf{V}^{\mathrm{T}}$.

Using the assumed model of document generation with the original semantic document representations $\mathbf{Z} \in \mathbb{R}^{D_z \times N}$,

$$\mathbf{U}\mathbf{V}^{\mathrm{T}} = \mathbf{A}^{\mathrm{T}}\mathfrak{F}_s(\mathrm{G}_s(\mathbf{W}_s\mathbf{Z}))\mathfrak{F}_t(\mathrm{G}_t(\mathbf{W}_t\mathbf{Z}))^{\mathrm{T}}\mathbf{B}. \tag{5.14}$$

Evidently, any feature generated by $\mathfrak{F}$ that does not originate from $\mathbf{Z}$ will decrease the correlations of the row vectors of $\mathbf{U}$ and $\mathbf{V}$. If $\mathbf{Z}$ is non-singular, $D_s = D_t = D_z$, and the matrices $\mathbf{W}_s$ and $\mathbf{W}_t$ are invertible, it is simple to show that the highest possible correlations are given by the feature generator that gives $\mathfrak{F}(\mathrm{G}_s(\mathbf{Z}_s)) = \mathbf{Z}_s$ and $\mathfrak{F}(\mathrm{G}_t(\mathbf{Z}_t)) = \mathbf{Z}_t$ (Appendix A.1). Even if $\mathbf{W}_s$ and $\mathbf{W}_t$ are not square or full rank matrices, CCA provides the optimal linear projections. Thus it is able to give insight on how well the evaluated VSM generates features that correspond to the shared meaning of the document pairs.

5.4.2 Evaluation setup

The evaluation method proposed in Publication III evaluates the feature generator $\mathfrak{F}$ of a vector space model. As most of the feature generators require learning, the training sets ($\boldsymbol{S_0}$ and $\boldsymbol{T_0}$ above) have to be separate from the evaluation data in order to deal with overfitting.

In the present evaluation method, however, the separation of training and evaluation sets is not alone sufficient: also the empirical correlation estimates returned by CCA may be overfitted. Especially if the number of samples N is small compared with the dimensionalities of $\mathbf{x}$ and $\mathbf{y}$, the sample estimates of the covariance matrices $\mathbf{C}_{xx}$, $\mathbf{C}_{yy}$ and $\mathbf{C}_{xy}$ are not robust. In order to get more reliable estimates of the final correlations, the evaluation set is divided into two distinct sets: namely, an evaluation training set (*evaltrain*) and an evaluation test set (*evaltest*). An overview of the evaluation setup is shown in Figure 5.4.

The evaltrain set $(\boldsymbol{S}, \boldsymbol{T})$ is used for training the parameters of CCA as described in Section 2.8.3. As regularization, a small positive value ϵ proportional to the

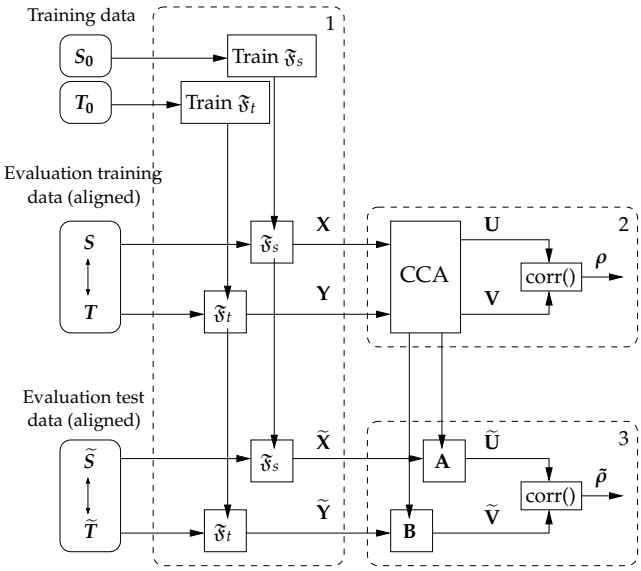


Figure 5.4. Diagram of the evaluation setup proposed in Publication III. (1) The feature generator $\mathfrak{F}$ is trained on monolingual corpora $\boldsymbol{S_0}$ and $\boldsymbol{T_0}$, and then applied to transform the evaluation data sets into vectorial form. (2) CCA is trained on the evaluation training data to find the canonical variates $\mathbf{U}$ and $\mathbf{V}$ and the respective projection matrices $\mathbf{A}$ and $\mathbf{B}$. (3) The evaluation test data is then projected into the same space, and finally the test set correlations $\tilde{\boldsymbol{\rho}}$ are computed.

variances of $\mathbf{X} = \mathfrak{F}_s(\boldsymbol{S})$ and $\mathbf{Y} = \mathfrak{F}_t(\mathrm{T})$ is added to the diagonals of the respective covariance matrices $\mathbf{C}_{xx}$ and $\mathbf{C}_{yy}$.

The evaltest set $(\widetilde{\boldsymbol{S}}, \widetilde{\boldsymbol{T}})$ is used for estimating the correlations for the evaluation score. For zero-mean $\widetilde{\mathbf{X}} = \mathfrak{F}_s(\widetilde{\boldsymbol{S}})$ and $\widetilde{\mathbf{Y}} = \mathfrak{F}_t(\widetilde{\boldsymbol{T}})$, the test set correlations $\tilde{\rho}_i$ are

$$\tilde{\rho}_i = \frac{\mathbf{a}_i^{\mathrm{T}} \widetilde{\mathbf{X}} \widetilde{\mathbf{Y}}^{\mathrm{T}} \mathbf{b}_i}{\sqrt{\mathbf{a}_i^{\mathrm{T}} \widetilde{\mathbf{X}} \widetilde{\mathbf{X}}^{\mathrm{T}} \mathbf{a}_i} \sqrt{\mathbf{b}_i^{\mathrm{T}} \widetilde{\mathbf{Y}} \widetilde{\mathbf{Y}}^{\mathrm{T}} \mathbf{b}_i}}, \tag{5.15}$$

where $\mathbf{a}_i$ and $\mathbf{b}_i$ are the projection vectors for the i^{th} canonical variate.

Instead of a vector of correlations $\tilde{\boldsymbol{\rho}}$, it is often preferable to have a scalar evaluation score. An intuitive score for comparing two feature generators that have the same output dimensionality D is the sum of the correlations:

$$R(\widetilde{\mathbf{X}}, \widetilde{\mathbf{Y}}) = \sum_{i=1}^{D} \tilde{\rho}_i. \tag{5.16}$$

For perfectly correlated sets, $R = D$, and for uncorrelated sets, $R = 0$. The experiments of Publication III show that this works in general better than, for example, assuming normally distributed data and calculating a "Gaussian mutual information" score based on Equation 2.85 (page 64).

In the case that the feature generators return different numbers of features, the comparison of the vector representations is not as straightforward. Uncorrelated canonical variates should be penalized, but even a dimension that has a very small positive correlation can be useful if it is weighted according to the strength of the correlation (Tripathi et al., 2008).

5.4.3 Validation experiments

Publication III validates the proposed vector space evaluation method in three different ways: First, it was shown that the results based on the evaluation method agree generally well with previous findings on VSMs. Second, the results were compared with two indirect evaluations that were based on sentence matching tasks. Third, the results were compared against a quantitative manual evaluation of the factor loadings of the bilingual features found by CCA.

The experiments were performed using Europarl (Koehn, 2005), a parallel corpus consisting of the proceedings of the European Parliament meetings in 11 European languages. The sentences of the corpus were used as documents. Among the language pairs, English–Finnish data was used except for a study of the effect of language similarities that included several language pairs.

The first set of validation experiments considered five different aspects of vector space construction. The hypotheses based the previous research and their agreements with the proposed evaluation method were the following:

1. Dimensionality reduction

   Hypothesis: SVD outperforms simple feature selection methods and random projection if the target dimensionality is relatively low.

   Confirmed by: Deerwester et al. (1990); Kaski (1998); Bingham and Mannila (2001)

   Result: Agreed; SVD showed significantly higher scores than any other tested method for dimensionalities between 20 and 500.

2. Global weighting

   Hypothesis: Among the different global weighting schemes, entropy weighting and logarithmic idf should give the best improvements when using SVD.

   Confirmed by: Dumais (1991); Nakov et al. (2001)

   Result: Agreed; entropy weighting and logarithmic idf outperformed other tested global weighting schemes up to 500-dimensional representations.

3. Amount of data

   Hypothesis: More training data should improve the results whenever a reasonable dimensionality reduction method is applied.

   Confirmed by: Zelikovitz and Hirsh (2001); Yarowsky and Florian (2002)

   Result: Agreed; using SVD as feature extraction, increasing (decreasing) the amount of training data showed significant increase (decrease) in the scores.

4. Phrases as features

   Hypothesis: The results are not improved by including common n-grams from the training data as features, unless the n-grams are selected very carefully.

   Confirmed by: Lewis (1992); Scott and Matwin (1999); Caropreso et al. (2001); Sebastiani (2002); Koster and Seutter (2003); Coenen et al. (2007)

   Result: Not agreed; adding 2-grams that occurred at least twice outperformed bag-of-words significantly for dimensionalities between 50 and 500. How-

ever, increasing the n-gram lengths up to five did not show clear improvements to 2-grams.

5. Language similarity

   Hypothesis: Due to morphological and syntactic similarities, closely related languages should have higher correlations than, for example, languages that belong to different language families.

   Confirmed by: Besançon and Rajman (2002); Chew and Abdelali (2007); Sadeniemi et al. (2008)

   Result: Agreed; among the tested language pairs the highest scores were obtained for Danish–Swedish corpus (same language family, sub-family, and branch), and lowest scores were obtained for Finnish–English and Finnish–German corpora (different language families).

Thus the only case in which the result disagreed with the hypothesis was the use of phrase features. This mismatch may be due to using vector representations of sentences: information on word order, disregarded in bag-of-words and partially encoded in bag-of-phrases, is likely to be more important for sentence representations than for document representations that were evaluated in the previous studies.

In the second validation approach, several different VSMs were evaluated by the proposed method and two bilingual sentences matching tasks. The general idea was to check how well the sentences that are translations of each other could be matched based on the vector space model and the projections learned via CCA. The first task was finding the translation of the sentence among a subset of other sentences. It was similar to the the nearest translation test used by Besançon and Rajman (2002) for VSM evaluation. The second task was one-to-one alignment of sentences and similar to the one studied by Tripathi et al. (2010). The evaluation scores and the matching accuracies were compared using Spearman's rank correlation coefficient (Section 2.5.4). Using a set of twelve different 100-dimensional representations, the correlation between translation accuracies and evaluation scores was 1.0, and the correlation between alignment accuracies and evaluation scores was 0.888. As the correlation between translation and alignment accuracies was also 0.888, indirect evaluations based on two very similar tasks could not predict each other's performance any better than the proposed direct evaluation.

The third validation exploited the canonical factor loadings of the generated features. The factor loadings measure which original variable is involved in which canonical variate and to what extent, and are obtained by

$$l_{x(ij)} = \mathrm{corr}(\mathbf{u}_i, \mathbf{x}_j); \qquad l_{y(ij)} = \mathrm{corr}(\mathbf{v}_i, \mathbf{y}_j). \tag{5.17}$$

For a bag-of-words model, the original variables are words. As an example from a model that used SVD, there was a variate that strongly correlated with English words **vote**, **place**, **take**, **tomorrow**, and **noon** and Finnish words **äänestys** (*vote*), **toimitetaan** (*takes place*), **klo** (*at* [*time*]), **huomenna** (*tomorrow*), and **12.00** (*noon*). While these are not direct translations, it is easy to see that they come from the same sentences in each language. A good feature generator should provide this kind of translation pairs. Thus the correctness of the translations, scored manually for the five most positively and negatively correlated words of each dimension, was compared with the evaluation scores. As a result for a

set of six 100-dimensional representations, the rank correlation between the two scores was 0.977.

5.4.4 Discussion

CCA and its variations had already been utilized in many applications of natural language processing prior to Publication III. The tasks include cross-language information retrieval (e.g. Vinokourov et al., 2003; Li and Shawe-Taylor, 2007; Hardoon and Shawe-Taylor, 2007), text categorization (Minier et al., 2007), bilingual lexicon learning (Haghighi et al., 2008), and sentence alignment (Tripathi et al., 2010). The novel idea of Publication III was to use CCA not as a part of some indirect evaluation but for directly measuring the dependence of the representations of data sets that should be highly dependent.

There are two restrictions in the proposed evaluation method. First, it uses linear dependence calculation, which, although very robust and fast, may not always find the true dependence. A solution would be to use kernel CCA to extend the evaluation approach for non-linear dependences. However, selecting the kernel function and its parameters would require a more complicated evaluation setup. It is also difficult to interpret the canonical variates in the kernel space. Second, the evaluation method works for monolingual feature extraction methods only. The evaluation setup encompasses cross-language feature extraction comparable to Vinokourov et al. (2003) or Zhang et al. (2010). Thus, if the features were extracted from a bilingual corpus, the evaluation scores would only measure how similar the generated features are to those found by CCA.

Finally, the evaluation method can be extended at least in two different directions. One is to consider a situation where the underlying features are not semantic. For example, in a genre identification task, the features are typically statistics on part-of-speech tags, word lengths, and punctuations (Finn and Kushmerick, 2006). Given a data set where paired documents were of the same genre but otherwise different content, a feature extraction that finds genre-related features would get high correlations. Second direction is to consider an asymmetric evaluation. The two views of the data set may have different types of text (e.g., full text and summarized text, professional and layperson language) or even different type of data (e.g., images and their captions), as long as it is mainly the semantics that is shared by the data sets. Moreover, if there already is a feature generator known to perform well for the first view of data, feature generators for the second view can be evaluated simply by measuring the correlations with the features of the first view. For example, consider the case that the first language is isolating and the second is synthetic. If some standard bag-of-words representation was considered good enough for the first language but not for the second, one could evaluate a morphologically-aware feature generator for the synthetic language.

6. Selecting lexical units

In this chapter, the problem of lexical unit selection, highlighted in the middle of Figure 5.1, is considered in more detail. As the problem has rarely been considered as a whole, the concepts of the problem are not well established. Thus this chapter starts by giving an extensive overview of the problem, including terminology, criteria for useful lexical units, and evaluation approaches (Section 6.1). Next, Section 6.2 compares different types of linguistic units regarding the criteria and the common applications of the units, and also gives an overview on the most popular approaches for identifying the units.

Section 6.3 goes deeper in the question of evaluation for one particular unit selection problem: morphological analysis. Among indirect evaluations, this thesis concentrates on statistical machine translation (Publications IV and V), but also speech recognition and information retrieval tasks are considered. For direct evaluation using linguistic resources, a central contribution of this thesis is the introduction and experimental comparison of various evaluation methods for unsupervised morphological analysis (Publication VI). Also direct evaluation with psycholinguistic data is considered. In particular, the contributions of this thesis include experiments for evaluating statistical models of morphology via reaction time data (Publication VII).

The last substantial contributions of this thesis, extensions for the Morfessor method for selecting lexical units, are described in Section 6.4. These extensions include a model for allomorphy (Publication VIII), a weighting method to deal with the effect of varying size of training data (Publication IX), a semi-supervised training algorithm (Publication X), and a method for learning phrasal constructions (Publication XI).

6.1 Problem definition

The low-level representation of a sample of written text, $\boldsymbol{t}$, is a string of a sequence of characters from a character set Σ: $\boldsymbol{t} \in \Sigma^*$. The length of the string may vary from a few characters in a single word to trillions of characters in a large corpus. Using the characters directly as the lexical units may work for some specific tasks. However, in many cases, it is useful to have a set of units that individually refer to particular meanings (cf. Section 3.1.4).

Adopting the basic framework and notation from Goldsmith (2010), the unit selection can be defined as encoding the samples $\boldsymbol{t}$ using a new lexicon $\mathcal{L}$. The units $w \in \mathcal{L}$ may have arbitrary features, but are often represented by finite-length substrings of some subset of Σ^*. For example, $\mathcal{L}$ can contain all lower-

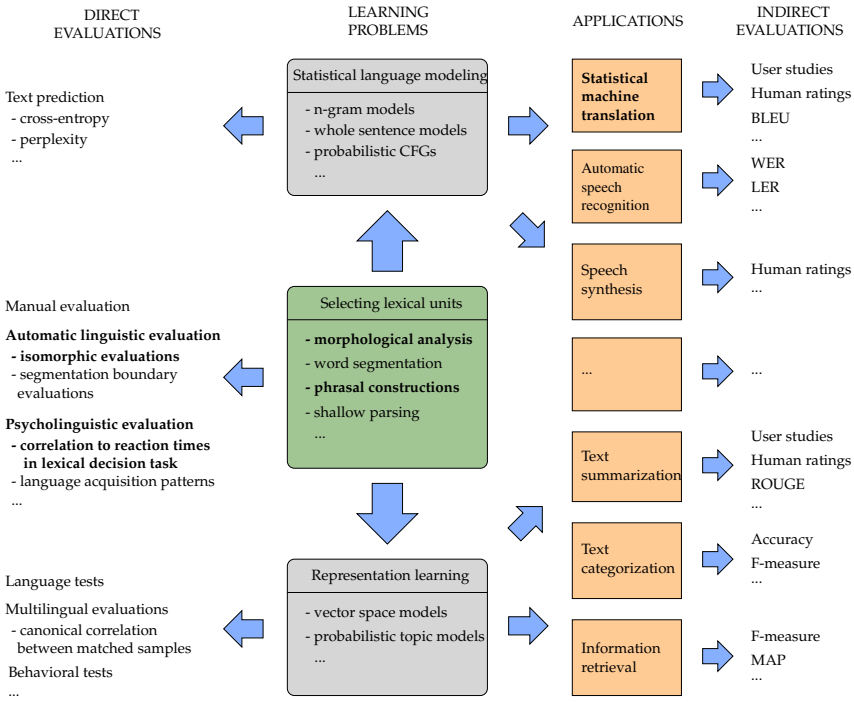


Figure 6.1. Unit selection in the overview diagram.

cased words observed in the samples (thus making uppercase letters as well as whitespace and punctuation characters in Σ unnecessary).

Goldsmith (2010) considers the problem of word segmentation and identifies two subtasks: learning the word lexicon $\mathcal{L}$ from a stripped corpus (i.e., such that there is no marking of word breaks), and finding the original segmentations given the lexicon and a stripped corpus. In the more general unit selection problem considered here, these will be called *unit learning* and *tokenization* tasks. In contrast to the word segmentation problem, no information is removed from the training corpus. If the target units, for example words, are already marked in the text, this makes both problems much easier (but not always trivial). Specifically, if the tokenization problem is easy to solve by devising a small set of manual rules, there is no need for unit learning.

Given a sample $\boldsymbol{t} \in \Sigma^*$, a tokenization function $\phi(\cdot)$ provides a *parse*

$$\boldsymbol{s} := \phi(\boldsymbol{t}) = (w_1, w_2, \ldots, w_n) \in \mathcal{L}^*. \tag{6.1}$$

The units w_i in the parse may overlap in the sample string or over-represent the sample. Examples of such approaches include combined index phrases and words in an IR system (Fagan, 1988) and using both words and morphemes in factored models (Bilmes and Kirchhoff, 2003; Koehn and Hoang, 2007). Using the terminology from Section 3.2.7, if there is a combination operator $\oplus$ and $w \in \mathcal{L}$ such that $w = u \oplus v$ for some $u, v \in \mathcal{L}$, the lexicon is *redundant*. If no such w exists, the lexicon is *minimal*.

Having several possible parses for single $\boldsymbol{t}$ is common and sometimes even desired due to the inherent ambiguity of language. In this case, the output of tokenization is a set of parses:

$$\mathrm{S} := \Phi(\boldsymbol{t}) = \{\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_M\}, \tag{6.2}$$

where $\boldsymbol{s}_i = \phi_i(\boldsymbol{t})$. However, usually the tokenization algorithm should provide a single best parse $\phi_{\text{best}}(\boldsymbol{t}) \in \text{S}$, either using probabilities $p(\boldsymbol{s}_i \mid \boldsymbol{t})$ or in some other well-defined manner.

Sometimes a parse $\boldsymbol{s}$ may have to be transformed back into a string $\boldsymbol{t} \in \Sigma^*$. This is called the *detokenization* problem. In many cases, detokenization is performed via the combination operator $\oplus$ and post-processing function $\psi(\cdot)$: $\boldsymbol{t} = \phi^{-1}(\boldsymbol{s}) = \psi(w_1 \oplus w_2 \oplus \ldots \oplus w_n)$. For example, if the tokenized sample is a sequence of lowercased word forms, combination includes adding word break characters between the words before concatenation, and post-processing includes predicting the correct capitalization and punctuation marks. Detokenization is necessary only for those applications that have to generate text from the tokenized samples. If tokenization is followed by representation learning—as in information retrieval—detokenization is rarely necessary.

6.1.1 Qualitative criteria for unit selection

In contrast to representation learning and statistical language modeling, unit selection does not have specific goals. Instead, the first step in the unit selection is to make the choice of the target units.

There are several questions to consider related to the application: Does it require density estimation or just calculation of similarities of parts of text? Is it enough to study orthographical features, or does the application need to assess semantic similarities? What kind of information is it acceptable to lose? For example, is syntax or morphology relevant for the application?

In addition, the choice of the units depends also on the language. With respect to the language, there are two main questions: What are the graphemes of the language, and what type of morphology does the language have?

Taking both the application and the language into account, there are four main criteria to consider in the unit selection problem:

- *Relevance:* The selected units have to be relevant to the next stage of processing, usually density estimation or representation learning. In most cases, the units should carry meanings that are as fixed and context-independent as possible. Any information relevant for the next stage should not be lost.
- *Coverage:* The set of units has to be able to model as large part of the data as possible. Especially, the number of out-of-vocabulary (OOV) units for any new data should be low. Moreover, units that have low frequency in the training data should be avoided, as their statistical properties are hard to estimate.
- *Compactness:* The number and complexity of the units may have to be limited due to memory and computation time constraints.
- *Ease of extraction:* The easier the tasks of unit learning, tokenization, and detokenization are, the better.

Even for a single target application and language, there rarely is a single choice of units that would clearly be the best for all of these criteria. Instead, one has to make a trade-off between the different aspects.

6.1.2 Evaluating unit selection

The above defined criteria are useful for considering what kind of units should be best suited for different tasks in NLP. However, some automatic way to measure the quality of the results is essential for developing an efficient method for unit selection.

6.1.2.1 Indirect evaluations

Again, usefulness of a method should eventually be evaluated by how it improves the performance of the target application, but the indirect evaluations in applications are often complicated and expensive in terms of time and manual work. Because unit selection is the first step in NLP, there are often additional layers of components (such as models for representation learning or statistical language modeling) before the actual application. For example, comparing different unit selection methods in automatic speech recognition requires a statistical language model based on the extracted units in addition to the application-specific components such as a pronunciation dictionary, acoustic models, and a decoder. Of course, the possibility to use the same methods in different tasks provides many additional indirect evaluations. For example, calculating cross-entropy of a text, which is a direct evaluation for a language model, is an indirect evaluation for the selected lexical units.

Different types of units are likely to be useful in different applications. On one hand, units that do not carry independent meaning, such as letters, are unlikely to be useful in information retrieval. On the other hand, complex units such as phrases might be hard to apply in language identification because of the problems of coverage. The most common applications for different types of linguistic units are discussed in the Section 6.2.

6.1.2.2 Direct evaluations

Direct evaluation for a unit selection method is usually based on a manually annotated corpora. Sometimes automatic rule-based annotation tools can be applied, if they are known to be reliable enough. The type of the annotations that are needed depend on the target units. For many unit selections tasks, tokenization is simply segmentation, and thus the annotations should determine the correct segmentation points.

Another type of annotations include single labels per word (e.g., POS tags, named-entity tags), multiple labels per word (morphological analysis), and parse trees from different types of grammars (e.g., CFGs, lexicalized CFGs, or dependency grammars). In some cases, the evaluation is simple enough. For example, in supervised POS tagging, the goal is to map a sequence of words to a sequence of tags. The performance of a tagger can be assessed simply by counting how many of the predicted tags x_i are the same as the reference tags y_i for a set of held-out data.

However, evaluation of labeling is not as straightforward for unsupervised learning. In unsupervised POS tagging, there is no way to tell whether a certain predicted tag x_i (say, "tag number 23") matches the reference tag y_i (say, "adjective"). More generally, we have the predicted output $\boldsymbol{x}$ and some reference data $\boldsymbol{y}$, but there is no direct way to assess whether a single output x_i is correct or not. This problem is considered in more detail in Section 6.3.2.

Similarly to vector space evaluation, an alternative to linguistic annotations is to use some type of psycholinguistic data. The recorded data may be, for example, reaction times or eye movements. The advantage of this type of evaluation is the independence from traditional linguistic theories and their concepts. For example, as mentioned in Section 3.1.5, existence of part-of-speech categories of words in the human mind is controversial. If humans unconsciously apply another kind of categories, such categories may also be more useful for NLP than the traditional ones.

6.2 Comparison of linguistic units

The usual approach to unit selection is to select a linguistically well-defined unit type, such as syllables or words, and design suitable tokenization and learning algorithms for it. In this section, the criteria defined in the previous section (relevance, coverage, compactness, and ease of extraction) as well as other practical issues are considered for different types of linguistic units.

6.2.1 Letters and syllables

Letters are the smallest unit type available for languages with a phonemic writing system.[1] The correspondence of letters and phonemes varies among different languages. Finnish has almost one-to-one mapping between letters and phonemes (with the exception of velar nasal [ŋ] that is spelled with two letters). In contrast, English orthography does not reflect phonemes in such a direct manner, and phonemes are often spelled differently in different contexts. This is because of the long history of written English: the pronunciation has changed over time and spelling has not been reformed to reflect the discrepancy. Moreover, there are writing systems called *abjads* that exclude vowels (e.g., Arabic and Hebrew) and *abugidas* (e.g. Hindi and Thai) that mark vowels with diacritics or other systematic changes to consonants.

As lexical units, the obvious advantages of letters are that (1) the units are trivial to find, tokenize, and detokenize, and (2) the set of units is very small and closed. That the set is closed means that no new units will appear and the coverage is perfect if all letters occur in the training data. Thus there is no OOV problem for letter-based models.

The drawback is that a single letter does not have much semantic content. To estimate any phenomena based on syntactic or semantic regularities, the models will have to store long contexts. If the average word length was five letters, letter n-gram models would need contexts of over 20 letters to store as much information as a word 5-gram. Storing all letter 20-grams in a training corpus is impractical, but sophisticated model selection alleviates the problem (Section 4.2.4). Even then the final model may not be as small as the size of the alphabet would indicate. Due to the lack of semantic relevance, letter-based features are not likely to be useful in maximum-entropy or continuous-space language modeling.[2] Nor are they useful for most of the representation learning

[1] For those languages that are usually written with logographs, there is the option of using universal phonetic alphabets such as IPA. However, this limits both applications and resources such as corpora.

[2] A possible solution would be a hybrid model that is conditioned the probability of the next letter based on whole words in the history.

problems.

Automatic speech recognition, optical character recognition, as well as handwriting recognition systems apply models that transfer the observations (sound or image signals) to letters. However, language modeling is usually based on words, and there are only few reported results (e.g., Magdy and Darwish, 2006) on letter-based models for optical character recognition. Such results are even harder to find for ASR.

A more common application for letter-based models is language identification (e.g., Beesley, 1988; Dunning, 1994; Vatanen et al., 2010). In contrast to word-based identification, the input for letter-based models may be only few words or even a fraction of a word, which makes them a more general solution.

Letter-based models are also useful for studying morphology. For example, the classic letter successor variety method for unsupervised morphological segmentation can be improved by calculating entropy with letter-based models (Hafer and Weiss, 1974). In this thesis, Publication VII demonstrates that letter n-gram models are also good predictors of word recognition times in a lexical decision task.

Syllables are units that are based on phoneme sequences. They form the alphabet in syllabic writing such as Japanese Kana, but are often simple to find from phonemic writing, too. The number of syllables is larger than the number of letters—but still relatively low compared with longer units—and the need for context length is slightly smaller. Still, the advantages and drawbacks of using syllables as units are likely to be very similar to those of using letters.

6.2.2 Words

The word is the default choice for the unit selection problem. Words encode semantic information by themselves, they are usually easy to extract from text, and a lexicon of some tens or hundreds of thousands words are often enough to provide reasonable coverage. However, the particular language that is modeled has a large impact on all the criteria.

The main problem in word-based modeling is that the set of potential units is not closed. New lexemes and word forms emerge over time. In addition, there are foreign words and spelling mistakes that have to be considered. The vocabulary size required for a reasonable coverage depends heavily on the morphology of the language. A relevant statistic to consider is the index of synthesis (Section 3.1.7) that gives the average number of morphemes per word in a particular language. For a high index of synthesis, the out-of-vocabulary rates are always high (Section 3.1.1).

While words are semantically very relevant units, they still suffer from problems with semantic ambiguity (cf. Section 3.1.9). There are many homonymous words, and polysemy is even more common. Moreover, the correct interpretation for idiomatic and metaphoric use of words requires that the context is taken into use.

In word n-gram models trained for ASR or SMT, the typical length of the n-grams was three, ten years ago, and currently ranging between four and six. Evidently, the applied context lengths are likely to increase along with the increase of available data and computational resources. N-gram growing techniques make it possible to leave n to be decided based on the training corpus.

Because of the semantic relevance and easy detokenization, words are useful

units in almost all NLP applications. Their usage is restricted mostly by the high OOV rates or tokenization problems in certain languages. The latter issue is considered next.

6.2.2.1 Learning and tokenization

The problems in word learning and tokenization depend on the language. In Western writing systems, where whitespace already separates the word forms, tokenization requires only dealing with different punctuations. Abbreviations (**Ph.D.**), numeric expressions (**$2,499.90**) and some other cases (**dot.com**, **:-)**) that include punctuation marks and thus require special attention. Another complication is introduced by hyphenation practices. For example, **finite** and **state** should be separated in **finite-state machines**, but **co** and **operate** should not be separated in **co-operate**. However, if some mistakes are allowed, word tokenization requires only a list of the most common irregularities in the usage of the punctuation marks.

An important exception are the languages in which the word breaks are not marked in the text. In this case, the problem is often called *word segmentation*. Especially Chinese word segmentation has been studied since the 1980s. The first approaches to word segmentation were those of *dictionary-based* tokenization (see, e.g., Wu and Tseng, 1993). In dictionary-based approaches, all known words were first stored in the lexicon. Then the texts could be tokenized by greedily finding the longest substrings found in the lexicon (*maximum matching*), or by combining single characters to words using linguistic or non-linguistic techniques.

Later development has concentrated on supervised probabilistic models that predict the positions of the word boundaries. One early work is by Teahan et al. (2000), who proposes an approach that predicts the word breaks using an adaptive n-gram model trained on presegmented text. Currently the methods are compared in annual "bake-off" competitions started by Sproat and Emerson (2003). Many current state-of-the-art methods are based on discriminative structured models such as CRFs (see, e.g., Sun and Xu, 2011).

Unsupervised word segmentation has been mainly studied as a problem of language acquisition. The first known approach, letter successor variety (LSV) by Harris (1955) was mentioned in Section 3.1.3. A major problem in LSV is that it easily produces inconsistent units, as the segmentation decisions are based only on the local context and not on the lexicon constructed so far. Another approach is to search for a lexicon that gives a compact representation for the input data. For example, de Marcken (1996) and Brent (1999) have proposed segmentation models based on the MDL principle discussed in Section 2.6.8.

6.2.3 Lexemes and stems

An alternative to the use of all individual word forms is to reduce them to lexemes represented by lemmas (see Section 3.1.4). *Lemmatization* means that the information encoded in the inflections is lost, so that information should not be relevant for the application. Because of the lost information, ambiguity is often slightly increased, but lemmatization may also disambiguate some word forms (e.g., **building** to verb BUILD or noun BUILDING). Detokenization is very hard if not impossible. The advantages are that the vocabulary size is decreased and coverage improved.

Given the above restrictions and advantages, lexemes are naturally suited for IR type of tasks (Järvelin and Pirkola, 2005). Better coverage improves recall, although precision may be degraded because of the increased ambiguity. The benefit is usually larger for highly-inflecting languages.

Outside IR tasks, lexemes and stems are frequently used as extra features in language modeling and machine translation. This includes both factored models (Bilmes and Kirchhoff, 2003; Koehn and Hoang, 2007) and hierarchical lexicons (Nießen and Ney, 2004).

Lemmatization is, in principle, a clustering task for the inflected word forms. It is a complicated problem that may require disambiguation based on the word context. For well-resourced languages, there are often rule-based morphological analyzers that provide lemmas for the inflected forms. They are discussed further in Section 6.2.4.

If the language has a low index of fusion, stems can approximate the lexemes. *Stemmers* (e.g., Porter, 1980) are often relatively easy to develop. In the simplest case, all that is required is a list of suffixes of the language. The main difference to lemmas is that stems do not usually include derivational suffixes. Thus stemming can improve recall and degrade precision more than lemmatization. In the mono- and cross-lingual IR experiments of Järvelin and Pirkola (2005), lemmatization worked better than stemming for Finnish, Swedish, and German languages. However, for English, stemming outperformed lemmatization.

Unsupervised lemmatization may be based on stemming—first identify all suffixes and then strip them off—or direct clustering of the word forms. This topic has been studied by Schone and Jurafsky (2000, 2001), Baroni et al. (2002), Snover et al. (2002), Hammarström (2006), and many others. Also unsupervised methods for learning morphology (see Section 6.2.4) can be applied to stemming or lemmatization. If the goal is lemmatization and not only stemming, there are major issues in how to distinguish between derivational and inflectional suffixes and how to disambiguate based on the context. Fusional processes, found for example in irregular verbs of English, are problematic both for lemmatization and stemming. Including some supervision, for example by providing a list of the canonical suffixes for different parts-of-speech (Yarowsky and Wicentowski, 2000), makes the task more feasible.

6.2.4 Morphemes and morphs

Morphemes are an attractive choice for unit selection: the coverage is the best and the vocabulary size is the smallest that can be obtained without including units that do not bear any meaning of their own (and thus hurt relevance). Compared with words, ambiguity is often higher and longer contexts are required for language modeling, but overall there should be few theoretical drawbacks.

Comparing morphs and morphemes, the usage of morphs increases both vocabulary size due to allomorphy and ambiguity due to syncretism. In an IR setting, this means both lower recall and lower precision. Obviously, the disadvantage is larger for languages with high index of fusion. Morphemes might be preferred in machine translation due to the lower semantic ambiguity of the units. However, if the translation model outputs morpheme sequences, there is the problem of detokenization. The advantage of morphs is that detokenization is possible by simple concatenation and prediction of word breaks, whereas morphemes require a more complicated model of word generation. Similarly

in ASR, morphs are directly observable in the input signal, whereas using morphemes would require additional processing.

Among the applications, morphemes and morphs have been tested thoroughly at least in IR, ASR, and SMT. The test settings and results are discussed in more detail in Section 6.3.1.

6.2.4.1 Learning and tokenization

The main drawback of morpheme and morph tokenization is that separate tools are required to extract them. The extraction is trivial only in isolating languages with an index of synthesis of almost one: in that case there is no essential difference between morphemes and words. Otherwise there is a need for either morphological analyzers or machine learning algorithms. As they usually require word forms as input, word tokenization may have to be done as preprocessing.

Morphological analyzers. As mentioned in Section 2.3.5, finite-state machines have been successful for computational modeling of phonology and morphology. The finite-state morphology started to develop already in early 1980s (see Karttunen and Beesley, 2005), and even very recent analyzers are based on the same methodology (e.g. Çöltekin, 2010).

The first approaches in computational morphology used context-sensitive rewrite rules that, when applied sequentially, produced surface forms of the words from abstract phonological representations stored in the lexicon. While the rules where context-sensitive, they were not applied recursively to the same position, and actually implemented only regular relations that can be described with FSTs (Karttunen and Beesley, 2005).

With respect to morphological analysis, the problem of the rewrite rules was that while one lexical form generates one surface form, a surface form can typically be generated from more than one lexical form. Using the rewrite rules in reverse direction, starting from the surface form, could in the worst case produce a huge number of potential lexical forms.

Before the computational problems of the rewrite rules were solved, an alternative approach was introduced by Koskenniemi (1983). His *two-level morphology* (TWOL) paradigm was powerful enough that he could build a complete morphological analyzer of Finnish already in the 1980s. "Two-level" means that the model considers only the phonological or orthographical surface level and the underlying, abstract lexical level. This is in contrast to the earlier generative models of phonology, introduced by Chomsky and Halle (1968), that use many intermediate levels during the generation or analysis. The rules of TWOL are sensitive to both the surface and lexical context, and they are applied in parallel rather than sequentially. Moreover, lexical lookup and analysis is performed in tandem. The lexicon was first implemented as a trie-based structure. Later, Karttunen et al. (1992) noticed that it could be modeled as well with a finite-state transducer. Then the lexicon could be combined with the finite-state rules to produce a single FST that maps between lexical and surface forms.

Machine learning. The obvious drawback in morphological analyzers is the human effort required to construct the analyzer and make it cover enough of the real-world data. Machine learning methods provide approximate solutions with less cost.

The research on morphology learning includes many types of setups and goals. Excluding morphological clustering (i.e., lemmatization and stemming that were

discussed above), at least the following problems have been studied:

- *Learning of inflectional processes:* Given a list of lemmas and their inflected forms, learn to generalize the regular and irregular inflections present in the data. Rumelhart and McClelland (1986) used this task to demonstrate how artificial neural networks could acquire linguistic knowledge. The setup can also be used for learning two-level rules for morphological analyzers (Theron and Cloete, 1997).
- *Out-of-vocabulary generalization:* Given a morphological analyzer or a large set of data annotated by an analyzer, predict the lemma and/or paradigm for previously unknown word forms (e.g. Bharati et al., 2001; Lindén, 2008, 2009).
- *Supervised analyzer:* Given a large set of annotated data, learn a model that performs morphological analysis (e.g. Wicentowski, 2004; Chrupała et al., 2008). Instead of using annotated data, one can also resort to direct human supervision and let the machine learning algorithms solve only some part of the problem (e.g, Oflazer et al., 2001).
- *Partially supervised analyzer:* Given partially annotated data, learn a model that performs morphological analysis. The partial information can be, for example, suffixes (Yarowsky and Wicentowski, 2000), pairs of lemmas and their inflected forms (Shalonova et al., 2009), or hand-written rewrite rules (Tepper and Xia, 2008).
- *Semi-supervised analyzer:* Given a small amount of annotated data and large amount of unannotated data, learn a model that performs morphological analysis. A simpler case is semi-supervised *segmentation*, in which case both the annotations and output are morphs. This is studied in Publication X.
- *Unsupervised analyzer:* Given large amounts of unannotated text, learn a model that performs morphological analysis. This is studied in Publication VIII. Again, a simpler task is unsupervised segmentation of words into morphs.

The approaches for unsupervised learning have been recently reviewed by Hammarström and Borin (2011). Similar to word segmentation, many models are inspired either by the letter successor variety (LSV; Section 3.1.3) or the minimum description length (MDL; Section 2.6.8) principle.

6.2.5 Phrases and phrasal constructions

Finally, there is the possibility to use units longer than single words. This includes both simple word sequences and more complex hierarchical units. The former will be called "phrases" and the latter "phrasal constructions". The term "phrase" is used in a broad sense, similarly to its use in statistical phrase-based machine translation (Koehn et al., 2003), including everything from collocations to complete clauses.

The main advantage of using phrases (or phrasal constructions) in the lexicon is the ability to represent a larger part of the semantic information more accurately than with individual words. This can happen in two manners: nearby words can disambiguate their meaning, or a phrase can have a non-compositional meaning (Section 3.1.9).

The cost of phrase tokenization is the increased size of the lexicon. The growth is exponential with respect to the length of the phrases, which means that some

linguistic or non-linguistic criteria are needed to decide which phrases are stored. Coverage does not diminish if all individual word forms are also retained in the lexicon. In consequence, phrase lexicons are often redundant. This is in contrast to letter, word, or morpheme lexicons, which are minimal. While the redundancy further increases the size of the lexicon, the possibility of using tree structures for the nested sequences can alleviate the problem (cf. varigram models in Section 4.2.4).

Probably the most important application for phrase tokenization is statistical machine translation. The first SMT models, developed by Brown et al. (1993), define probabilities and alignments between individual words. The problem that there is no one-to-one mapping between the words of two languages was recognized, but they had no means to deal with the combinatorial explosion of multi-word units. The modern phrase-based systems (e.g. Koehn et al., 2007) often still use the IBM models by Brown et al. (1993) for the word alignment problem. However, the actual translation model consists of conditional probabilities $p(\boldsymbol{w}_t \mid \boldsymbol{w}_s)$ over a fixed set of phrase pairs $(\boldsymbol{w}_s, \boldsymbol{w}_t)$. Note that the probability of a phrase $\boldsymbol{w}_t$ is conditioned only on the corresponding phrase $\boldsymbol{w}_s$ in the other language, and not the other phrases in the language t. This demonstrates how phrases make it possible to deal with both ambiguity and non-compositionality. Hierarchical phrasal constructions have been used, for example, in data-oriented translation (Poutsma, 2000; Hearne and Way, 2006) and hierarchical phrase-based translation (Chiang, 2007).

In information retrieval and text categorization tasks, it would similarly seem useful to use phrases as index terms. However, the studies of "bag-of-phrases" in vector space models have shown that including multi-word units does not improve the results unless the phrases are selected carefully (Fagan, 1988; Lewis, 1992; Scott and Matwin, 1999; Caropreso et al., 2001; Sebastiani, 2002; Koster and Seutter, 2003; Coenen et al., 2007). In contrast, Shen et al. (2006) uses the multigram model by Deligne and Bimbot (1997) for text categorization, and obtain the best results with maximum length of three words. Also Koster et al. (2011) obtain improvements over bag-of-words, this time using triplets of two words and their relation extracted by a dependency parser. This question is certainly not settled, and the results of Publication III indicate that phrases are useful at least for sentence-level information (Section 5.4.3).

In ASR, ambiguity and non-compositionality are problems mainly for the language model component. In n-gram models, the probabilities are naturally conditioned on some kind of phrases, but using phrases as lexical units means that also the predictions are made over multi-word units. This kind of phrase-based language models has been studied, for example, by Deligne and Bimbot (1995), Ries et al. (1995), Giachin (1995), Klakow (1998), Kuo and Reichl (1999), and Yamamoto et al. (2003). The main benefits are that longer contexts are incorporated in the language model and ability to use different pronunciations for the phrases (e.g., **"going to"** pronounced as **"gonna"**). While this approach has been shown to outperform word-based n-gram models, there seems to be no comparison to modern variable-length n-gram models. In fact, Heeman and Damnati (1997) argue that it is more useful to keep words as the basic lexical entries for the language model, and add phrases just to help the speech recognizer with the acoustic models.

Identification of phrases is also important for speech synthesis, where the intonation patterns should be affected by the phrase boundaries (see, e.g., Wang and Hirschberg, 1992).

6.2.5.1 Learning and tokenization

Considering the extraction of phrasal units, there are various tasks to pursue. The most typical are the following:

- *Learning of collocations:* Find pairs of words that form collocations or idioms.
- *Named-entity recognition:* In named-entity recognition, the task is to find groups of words that together form a single proper name (e.g. person, location, or organization). In named-entity classification, the found entities are tagged according to the type of the name (Nadeau and Sekine, 2007).
- *Chunking and shallow parsing:* In chunking or shallow parsing, the task is to segment the text into syntactically related non-overlapping groups of words (Sang and Buchholz, 2000). Usually the task includes also identifying the type, for example noun or verb phrase, of each chunk.
- *Parsing:* Parsing generally outputs a hierarchical structure over the whole sentence, which is evidently too large a unit. However, the subtrees can be used as smaller units. In some lexicalized grammars such as tree-adjoining grammars, the lexical elements are already trees. In contrast to shallow parsing, these units can be discontiguous and contain open "slots" for new words or subtrees.

Collocations can be found in an unsupervised manner using rather simple statistical tests or information-theoretic measures on co-occurrences of words (Manning and Schütze, 1999, Ch. 5). In most cases, one can restrict the study to consecutive words. However, this may not work well for languages of relatively free word order (Pirkola, 2001; see Section 3.1.8.3). Even in English, the same dependency as in **strong coffee** occurs in the sentence **"The coffee was very strong"**. Partially filled idioms include a modifiable part, as in **"jog X's memory"**.

Methods for chunking include CFG parsers (Abney, 1991) and supervised learning (Ramshaw and Marcus, 1995), and more recently also unsupervised learning (Ponvert et al., 2010, 2011). The methods for named-entity recognition are predominantly supervised; Nadeau and Sekine (2007) give a survey of the topic.

As discussed in Section 3.2, parsers are usually trained on treebanks, but can be learned also in an unsupervised manner (e.g., Clark, 2001; Klein and Manning, 2002, 2004; Solan et al., 2005; Bod, 2006). In the contributions of this thesis, Publication XI considers an intermediate task between shallow parsing and full-tree parsing: learning of phrasal constructions that are otherwise contiguous but may contain open slots for a restricted set of words.

6.3 Evaluations for learning morphology

In this section, the problem of automatically evaluating one specific case of unit selection—learning of morphology—is considered in more detail. Section 6.3.1 discusses indirect evaluations, including evaluation via statistical language modeling and evaluation in the actual applications. The contributions of this thesis are related to using morphemes in statistical machine translation (Publication IV and Publication V).

For direct evaluations, two types of reference data is discussed: Section 6.3.2 considers linguistic gold standard analyses (usually derived by rule-based morphological analyzers) and Section 6.3.3 considers psycholinguistic data. The new contributions for these two topics are from Publication VI and Publication VII, respectively.

6.3.1 Indirect evaluations

The most well studied applications for morphology learning are speech recognition, information retrieval, and machine translation.[3] In addition, morphological units have been studied in statistical language modeling. In principle, evaluation via some direct evaluation method of representation learning would also be possible, but there seems to be no published results for that direction.

This section gives an overview of the prior studies on using morphemes in these tasks and describes the indirect evaluation setups used in Morpho Challenges (Kurimo et al., 2010a). The statistical machine translation evaluation setup has been developed in Publications IV and V. The results of these two Publications are explained in more detail.

6.3.1.1 Statistical language modeling

Statistical language modeling provides a very quick way to evaluate lexical units in the case that they are segments of the text. There are some pitfalls that have to be avoided: First, the cross-entropies or perplexities have to be normalized correctly per word (see Section 4.1). Second, the number of OOV words have to be the same. The OOV problem is avoided if the models can segment every possible word to the units in the lexicon—for example, by adding each individual letter as a unit. Third, the language model itself should not bias results because of different unit lengths: shorter units require a higher number of units in the context. Moreover, there has to be a way to compare the complexities (sizes) of two models that utilize different units.

Hirsimäki et al. (2006) compare gold standard morphs extracted from a two-level morphological analyzer (Creutz and Lindén, 2004) and statistical morphs from Morfessor Baseline (Creutz and Lagus, 2005b) using n-gram models of orders 2–7. They find that the statistical morphs work better for small language model sizes (in megabytes, models stored in a tree structure), but for larger models and higher n, the results even out.

6.3.1.2 Speech recognition

Hirsimäki et al. (2009) give an overview of speech recognition studies that compare word and morph-based systems. The studied languages have included Arabic, Czech, Dutch, Estonian, Finnish, German, Hungarian, Slovenian, and Turkish. Both morphological analyzers and unsupervised methods have been tested. In most of the cases, morphs have outperformed words.

The main benefit of morph-based speech recognition is the better handling of out-of-vocabulary words (Creutz et al., 2007). For a fair comparison, it is essential that the lexicon can be large enough for word-based models and that

[3] Other applications are not discussed here, but include, for example, grapheme-to-phoneme conversion (Demberg, 2007).

the order of the language model can be high enough for morph-based models. Hirsimäki et al. (2009) report extensive recognition experiments using various word and morph vocabularies and varigram models for Finnish and Estonian. They find, for example, that a recognizer using a lexicon of 500,000 words is outperformed by a recognizer using a lexicon of 2,000 morphs, when both use a varigram model of a similar size.

Different types of morphological segmentations have been tested also by Mihajlik et al. (2010). They compare Morfessor Baseline and Categories-MAP (see Section 6.4.1), Hungarian morphological analyzer Hunmorph (Tron et al., 2005), as well as a combined approach, where Morfessor Baseline is used to select a concise set of morphs from the multiple analyses provided by Hunmorph. All morph-based approaches outperform the word-based baseline. The best results are obtained with the combined approach, but those using Hunmorph or Morfessor Categories-MAP are not significantly worse.

Instead of using language models based on morphs, it is possible to combine morph and word lexicons with factored language models (e.g. Kirchhoff et al., 2006) or maximum-entropy models (Sarikaya et al., 2008). However, different algorithms for morphological analysis have not been compared in these combined approaches.

ASR experiments in Morpho Challenge 2005. Automatic speech recognition was used as an indirect evaluation for morphological segmentation in Morpho Challenge 2005 (Kurimo et al., 2006a). The tested languages were Finnish and Turkish. The best performing algorithms—including Morfessor Categories-MAP and algorithms by Bernhard (2006) and Bordag (2006)—yielded significantly lower error rates than the worst, but there were no statistical significance between the best algorithms.

6.3.1.3 Information retrieval

In IR type of tasks, where inflectional suffixes clearly have insignificant relevance, a lexicon of morphemes (morphs) is often reduced to lemmas (stems). A major part of the suffixes (and prefixes) can often be removed by using a maximum frequency threshold for the morphemes. Also global weighting schemes such as idf (see Section 5.1.1.2) reduce the effect of the non-content morphemes.

For morphological segmentation, the main aspect related to IR performance is how aggressively the words are segmented: oversegmentation (having too short stems) conflate semantically distinct forms and undersegmentation (having too long stems) keep semantically related forms distinct. The former is problematic for recall and the latter for precision. Compared with the ASR and language modeling task, IR evaluation is better in the sense that also non-concatenative processes of the language matter: modeling allomorphy should improve the recall of the retrieval system, and modeling syncretism should improve the precision.

Information retrieval experiments were performed already by Hafer and Weiss (1974). They used segmentation based on the letter successor varieties (LSV) and some heuristics to remove suffixes and prefixes. This simple automatic stemmer outperformed the word-based approach and yielded results similar to a traditional rule-based stemmer in two English tasks. To mention some of the more recent studies, Hull (1996) report extensive experiments regarding different stemming algorithms for English. They found that in most of the cases,

some sort of stemming is useful, and simple plural removal is less effective than more complex algorithms. However, they found no difference between the average performance of the more complex algorithms. Alkula (2001) use rule-based morphological analyzers to compare stem expansion (inflected words in index), lemmatization (lemmas in index), and lemmatization and compound splitting in a Finnish IR task. She achieved the best average recall using the index containing the lemmas and components of the compound words.

As in speech recognition, combining morphological units and words may be a practical approach for improving the results. For example, Turunen and Kurimo (2011) are able to improve spoken document retrieval by combining lemmas (from a morphological analyzer) and statistical morphs (from Morfessor).

Again, the need for morphological processing depends on the language. Detailed discussion is provided by Pirkola (2001), who classifies languages from the IR point of view. He considers both the index of synthesis and the index of fusion separately for inflections, derivations and compounding, as well as frequencies of semantic phenomena such as homonymy, polysemy, and synonymy. He proposes that different linguistic variables would be useful, for example, for predicting the effectiveness of a certain type of morphological processing. However, he does not consider any empirical results.

IR experiments in Morpho Challenges. Information retrieval evaluations were introduced in Morpho Challenge 2007 (Kurimo et al., 2008) in order to have an application that would benefit from morphological analysis of the input words more than from segmentation only. The same IR tests were applied also in the three subsequent Morpho Challenges (Kurimo et al., 2009, 2010c,b). The test languages have included English, Finnish and German.

In order to measure the effect of the morphological analysis in use, the other parameters of the system, such as weighting, have to be impartial. In the Morpho Challenge experiments, a modern tf-idf-style weighting called Okapi BM25 (Robertson and Zaragoza, 2009) was applied. One problem with the BM25 is that compared with the standard tf-idf, it suffers from terms that have a very high document frequency (Kurimo et al., 2008). As a solution, automatic "stop morph lists" were collected based on the corpus frequency of the morphs. With the stop lists, Okapi BM25 outperformed tf-idf regardless of the particular analysis, and has been used in the subsequent evaluations.

With the limited number of queries in the tested data sets, it has been hard to find statistically significant differences between the results. The 1–17 best performing algorithms have belonged to the top group that have no significant differences between their results (Kurimo et al., 2010a). However, some conclusions have been made. First, the language specific reference methods—stemmer by Porter (1980) for English and morphological analyzers based on the two-level morphology by Koskenniemi (1983) for Finnish and German—have yielded the best results. However, the best unsupervised algorithms—such as Morfessor Baseline, Monson et al. (2008), Bernhard (2006), McNamee (2008)—are almost at par, and the differences are not statistically significant.

6.3.1.4 Statistical machine translation

Processing morphologically complex languages is still an open problem in the statistical machine translation research. The standard word-based approach is evidently not good enough. For example, Koehn and Monz (2005), who built

110 translation systems between European languages, found that the most difficult language to translate both to and from was Finnish, known for its rich morphology.[4]

Using morphemes or morphs in SMT has been studied for example by Corston-Oliver and Gamon (2004), Lee (2004), Nießen and Ney (2004), Goldwater and McClosky (2005), Zollmann et al. (2006), Yang and Kirchhoff (2006), and Oflazer and El-Kahlout (2007). A common setup in these studies is that the translation is from a morphologically rich language (such as Arabic, Czech, German, or Finnish) to English, and morphological analysis is applied only to the former.

Moreover, apart from the Master's Thesis by Sereewattana (2003), all of the studies prior to Publication IV have used language-specific resources for the analysis: annotated data (Lee, 2004; Goldwater and McClosky, 2005), morphological analyzers (Zollmann et al., 2006; Oflazer and El-Kahlout, 2007), or stemmers (Yang and Kirchhoff, 2006; Oflazer and El-Kahlout, 2007).

As in speech recognition, the main benefit of morpheme-based machine translation is to decrease the number of OOV units. Larger improvements to evaluation scores have been obtained especially for small training corpora (Nießen and Ney, 2004; Sereewattana, 2003; Lee, 2004; Yang and Kirchhoff, 2006).

Statistical machine translation using unsupervised morphology induction. Publication IV describes a language-independent, morphologically aware translation system that uses Morfessor Categories-MAP (see Section 6.4.1) as a preprocessing tool, and studies models trained for a relatively large corpus, Europarl (Koehn, 2005). As Morfessor is an unsupervised method, no further linguistic resources are needed than in the standard, word-based approach.

The statistical machine translation system applied in Publication IV is based on Moses (Koehn et al., 2007), an open-source toolkit for phrase-based statistical machine translation. Moses is constantly developed, and provides very competitive results in shared tasks (Callison-Burch et al., 2011). The current version includes also tree-based translation models.

Prior to training the translation and language models, the source and target language sentences are tokenized by Morfessor Categories-MAP. Apart from segmentation, Categories-MAP tags each morph as a stem (`STM`), suffix (`SUF`), or prefix (`PRE`). The morphs that are not the last morphs of a word are marked with an additional symbol (+) in order to make detokenization easy.[5] Then the phrase pairs extracted during training consist of sequences of morphs, not words. Language models are trained on the morph sequences. The decoder outputs a sequence of morphs, which are then concatenated whenever marked with the correct symbol. The translation process is illustrated by Figure 6.2.

The translation system was tested on the Danish, Finnish and Swedish parts of the Europarl (v2) corpus (Koehn, 2005). BLEU by Papineni et al. (2002) was used as the evaluation score. The quantitative results of the translation experiments are shown in Table 6.1. On one hand, the morph-based models obtained lower BLEU scores, but the differences were statistically significant only for two tasks (Finnish–Swedish translations). On the other hand, morph-based translations had remarkably lower numbers of translations that included untranslated

[4] Apart from the out-of-vocabulary problem, further challenge with morphologically complex languages is that they tend to have a relatively free word order.

[5] Another option, used in the speech recognition experiments of the thesis, is to add a separate word boundary symbol to the lexicon. That approach did not work well in the translation experiments. In particular, the alignment tools of Moses had severe problems with the boundary symbols.

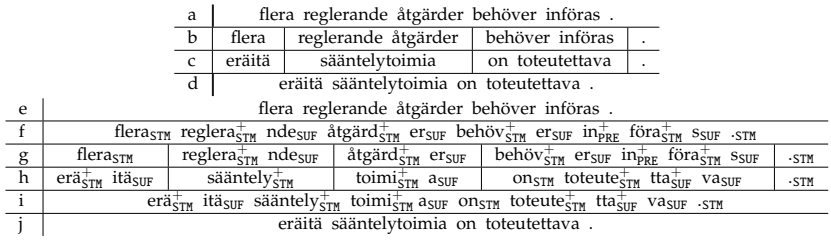


Figure 6.2. Example of translating the Swedish sentence **"Flera reglerande åtgärder behöver införas."** (*Several regulations need to be implemented.*) into Finnish (from Publication IV). The top figure shows the word-based translation process: the source sentence (a), the applied phrases (b), their corresponding translations (c), and the final hypothesis (d). The bottom figure illustrates the morph-based translation process of Publication IV: the source sentence as words (e), as morphs (f), the applied morph phrases (g), their corresponding translations (h), and the final hypothesis with morphs (i) and words (j).

words.

Table 6.1. Statistical machine translation results for word and morph based models in Publication IV. Statistically significant differences in BLEU scores are marked with an asterisk. The test set has 1,000 sentences.

	BLEU score (%) word / morph	OOV sentences (%) word / morph
Danish → Swedish	33.16 / 32.64	7.4 / 1.2
Swedish → Danish	35.95 / 35.49	7.6 / 2.1
Danish → Finnish	18.26 / 17.66	12.8 / 3.1
Finnish → Danish	23.63 / 22.40	18.9 / 4.1
Finnish → Swedish	22.85 / 20.71*	19.5 / 4.4
Swedish → Finnish	18.19 / 17.05*	13.2 / 4.2

The translation results were also studied qualitatively. In particular, the relevant question is whether the morph-based system can learn sensible phrase pairs that would not be possible for a word-based model. The answer was positive: Figure 6.3 shows phrase pairs that use the morphological segmentations in a productive manner. All examples are taken from the actual translations. The first three pairs show a similar structure across the languages, while the last four examples show how different languages prefer different linguistic structures. For example, in the fifth example, mood and person expressed by morphological markings of the verb in Finnish are translated to separate words in Danish. In the sixth example, a noun phrase in Finnish is translated to a verb in the infinitive form in Swedish.

Minimum Bayes risk combination of alternative morphological analyses. Publication IV showed that unsupervised morphological segmentation does not yield similar improvements in SMT as it has done in ASR. A particular problem might be the increased complexity of the morph alignment compared with the word alignment, as there are more morphs than words per sentence.

If morph-based models have problems improving the quality as such, but they still alleviate the out-of-vocabulary problems, one practical way to proceed is to combine word and morph-based approaches. This can be done either by devising a single model that use both kind of lexical units, or combining two different translation models as post-processing. The former option includes, for example, phrase-based back-off models (Yang and Kirchhoff, 2006) and factored translation models (Koehn and Hoang, 2007), and training the model with lattices gen-

Swedish:	Köpenhamn+ | s+ | kriterier+ | na	*Copenhagen+ | 's | criteria | the*
Danish:	København+ | s+ | kriterier+ | ne	
Swedish:	risk+kapital+ | marknad+er	*risk + capital | market+s*
Finnish:	riski+pääoma+ | markkina+t	
Finnish:	he | herja+ | sivat	*they | insult+ | ed*
Swedish:	de | förolämpa+ | de	
Swedish:	Litauen | för+fogar över en	*Lithuania | has_at_its_disposal a*
Finnish:	Liettua+ | lla on käytös+sä+ä+n	*Lithuania | by there_is at_its_disposal*
Finnish:	reagoi+ | si+mme	*react | would + we*
Danish:	vil vi | reagere på	*will we | react on*
Finnish:	etu+matkan kiinni kuro+ | minen	*advantage + distance's clos+ | ing*
Swedish:	att | komma i fatt	*to | catch up*
Finnish:	standard+i+en | nosta+ | misessa	*standard+s' | rais+ | ing_in*
Danish:	med | en for+høje+lse | af standard+er	*with | a raise | of standard+s*

Figure 6.3. Examples of productive morphology used by the morph-based translation model of Publication IV. Literal translations of the phrases are shown on the right.

erated by combining different source text analyses (Dyer et al., 2008). The latter option is studied in Publication V. The major benefit of integrating the results of multiple systems is its simplicity: standard, existing translation systems can be applied. Moreover, as the systems are independent until the combination, the burden of the training, tuning, and decoding is easy to divide.

In particular, Publication V applies the minimum Bayes risk (MBR) system combination proposed by Sim et al. (2007). Next, a short overview of the technique is given (based on Kurimo et al., 2010c).

In minimum Bayes risk decoding (Kumar and Byrne, 2004), the idea is to select the translation hypothesis $\boldsymbol{t} \in \mathrm{T}$ that has the lowest risk according to the applied probabilistic model $p(\boldsymbol{t}, \boldsymbol{s} \mid \boldsymbol{\theta})$ of target sentences $\boldsymbol{t}$ and source sentences $\boldsymbol{s}$. The risk means the expected value of the cost $L(\boldsymbol{t}, \boldsymbol{t}_{\mathrm{ref}})$ for selecting $\boldsymbol{t}$ instead of the correct answer $\boldsymbol{t}_{\mathrm{ref}}$. Note that the correct answer $\boldsymbol{t}_{\mathrm{ref}}$ is not known. Instead, it is assumed that the probability that the hypothesis $\boldsymbol{t}$ is correct is proportional to $p(\boldsymbol{t}, \boldsymbol{s} \mid \boldsymbol{\theta})$.

Equivalently to the minimal cost $L(\boldsymbol{t}, \boldsymbol{t}_{\mathrm{ref}})$, one can look for the maximal gain $G(\boldsymbol{t}, \boldsymbol{t}_{\mathrm{ref}}) = L_{\max} - L(\boldsymbol{t}, \boldsymbol{t}_{\mathrm{ref}})$, where $L_{\max}$ is the maximal cost. Using the gain function, MBR yields the decision rule

$$\hat{\boldsymbol{t}} = \arg\max_{\boldsymbol{t} \in \mathrm{T}} \sum_{\boldsymbol{t}_{\mathrm{ref}} \in \mathrm{T}} G(\boldsymbol{t}, \boldsymbol{t}_{\mathrm{ref}}) p(\boldsymbol{t}_{\mathrm{ref}} \mid \boldsymbol{s}, \boldsymbol{\theta}). \tag{6.3}$$

In SMT, an appropriate gain function is the sentence-level BLEU score. For efficient processing, the scores can be approximated by the following gain function (Tromble et al., 2008):

$$G(\boldsymbol{t}, \boldsymbol{t}_{\mathrm{ref}}) = \alpha_0 |\boldsymbol{t}| + \sum_{\boldsymbol{w} \in \mathrm{N}} \alpha_{\boldsymbol{w}} c(\boldsymbol{w}, \boldsymbol{t}) \, \mathrm{I}(\boldsymbol{w} \in \boldsymbol{t}_{\mathrm{ref}}), \tag{6.4}$$

where N is the set of n-grams, α_0 and $\alpha_{\boldsymbol{w}}$ are constants (with approximations derived by Tromble et al., 2008), and $c(\boldsymbol{w}, \boldsymbol{s})$ is the number of times the n-gram $\boldsymbol{w}$ occurs in sentence $\boldsymbol{s}$. The decision rule is then

$$\hat{\boldsymbol{t}} = \arg\max_{\boldsymbol{t} \in \mathrm{T}} \left\{ \alpha_0 |\boldsymbol{t}| + \sum_{\boldsymbol{w} \in \mathrm{N}} \alpha_{\boldsymbol{w}} c(\boldsymbol{w}, \boldsymbol{t}) p(\boldsymbol{w} \mid \boldsymbol{s}, \boldsymbol{\theta}) \right\}, \tag{6.5}$$

where the posterior probability of an n-gram is computed by

$$p(\boldsymbol{w} \mid \boldsymbol{s}, \boldsymbol{\theta}) = \sum_{\boldsymbol{t} \in \mathrm{T}} \mathrm{I}(\boldsymbol{w} \in \boldsymbol{t}) p(\boldsymbol{t} \mid \boldsymbol{s}, \boldsymbol{\theta}). \tag{6.6}$$

In the MBR system combination (Sim et al., 2007), the N-best lists (or lattices) T_1 and T_2 produced by the translation systems are combined to form the final set of hypotheses, and the probability of the n-gram $\boldsymbol{w}$ is calculated by linear interpolation between the two systems:

$$p(\boldsymbol{w} \mid \mathrm{T}_1 \cup \mathrm{T}_2, \boldsymbol{\theta}) = \lambda p(\boldsymbol{w} \mid \mathrm{T}_1, \boldsymbol{\theta}_1) + (1 - \lambda) p(\boldsymbol{w} \mid \mathrm{T}_2, \boldsymbol{\theta}_2), \tag{6.7}$$

where λ is the weight associated with each system. It is optimized using a development set.

In Publication V, the MBR combination approach was tested on two tasks: Arabic to English translation using two different morphological segmentations for Arabic, and Finnish to English translation using words and morphs induced by Morfessor Categories-MAP. In both cases, the combination provided a statistically significant improvement to the BLEU scores. For Arabic-to-English translation, the increase was from 52.7% and 52.8% to 54.6% for one test data set, and from 43.7% and 43.3% to 45.6% for the other data set. For Finnish-to-English translation, the increase was from 27.9% (word-based) and 27.4% (morph-based) to 28.9%.

SMT experiments in Morpho Challenges. Statistical machine translation experiments have been used in Morpho Challenge 2009 and 2010 (Kurimo et al., 2010c,b). They have included Finnish-to-English and German-to-English tasks, in which morphological analyses for Finnish and German are considered. The benefit of this setup is that the target language can be modeled based on words, so that there is no need for detokenization and the language model can be shared.

The translation system follows the setup of Publication V. First, two translation models are trained: one uses word lexicons for both source and target sentences, and other uses a morpheme lexicon for source sentences and a word lexicon for target sentences. The translation models are trained with Moses (Koehn et al., 2007). The weights of the different components (translation model, language model, reordering model, generation model) are tuned by maximizing the BLEU score for the development set.

Second, the translation results of the two models are combined by MBR system combination. At most 200 distinct hypotheses were generated for each sentence from both models; less if the decoder could not find as many. The best overall translation is found with the MBR decoding, and BLEU is used as the final evaluation metric.

As there are more morphemes than words in a sentence, two limitations affected the results: First, the alignment tool of Moses could not align sentences longer than 100 tokens. This means that algorithms that predict many morphemes per word may have reduced amount of training data for the translation models. To make the evaluation more fair in this aspect, all sentences that had more than 100 letters were discarded from the training data in Challenge 2010. (The algorithms of Morpho Challenge 2009 were re-evaluated in the modified setup for Publication VI.) Second, there is a freely selectable limit for how many units there can be in a phrase. The default value is 7. For the morpheme-based models, the limit was increased to 10.

The top performing algorithms for the tasks have been Morfessor Baseline and Categories-MAP (Section 6.4.1), Allomorfessor (Section 6.4.2), and MetaMorph

(Tchoukalov et al., 2010). They and a few other algorithms have yielded statistically significant improvements over the word-based baseline, while about half of the evaluated algorithms have not. Thus also the SMT evaluation is able to identify useful (and less useful) algorithms for the morphological unit selection, but finding significant differences between the best results is again difficult.

6.3.1.5 Conclusions

While indirect evaluations are essential for studying the problem of morphology induction, they are not very easy to design. The evaluation is quite straightforward in statistical language modeling, but it is essential that the language model is able to use long contexts whenever needed. The varigram models developed in Publication I should be suitable for this. In principle, one should select the best model of a fixed size for all different units, and then measure the normalized cross-entropies of these models. The only published comparison that we are aware of indicates that significant differences are found only for small model sizes, where morpheme-like units outperformed words (Hirsimäki et al., 2006).

Application evaluations provide more informative results, but require more work and resources. Regardless of the particular application, a common problem seems to be that very large test sets are required to see whether a difference between the results of two reasonably good methods is statistically significant.

The information retrieval experiments have so far given the most useful results in the Morpho Challenge evaluations. The standard models for IR are also relatively quick to train. However, an IR setting considers only usefulness of the morphological clustering indicated by a particular algorithm—suffixes and other grammatical morphemes are not relevant. An application that has to model individual sentences instead of documents should be meaningful for evaluating the results of a full morphological analysis. One example is machine translation. The SMT framework developed in this thesis has proven to be useful: statistically significant differences can be found between the results of different learning algorithms, and the best algorithms improve the translation results over word-based models.

6.3.2 Automatic linguistic evaluation

Most work on unsupervised learning of morphology is evaluated either by manual inspection or by comparing the predicted analyses with gold standard analyses produced by linguists, either manually or by rule-based methods. Manual evaluation is often essential during the development of the learning method, but too burdensome to use for large amounts of data. Manual inspections, such as the one by Goldsmith (2001), reveal that the decisions on correct and incorrect answers can be subjective, and usually a binary categorization to correct and incorrect is too coarse. For example, Goldsmith (2001) uses four categories: “good”, “wrong analysis”, “failed to analyze”, and “spurious analysis”. Thus also the automatic evaluations should consider similarities between the analyses, not only whether the result is exactly the same or not.

Most of the work on unsupervised learning of morphology concentrates on segmentation (Hammarström and Borin, 2011). It is a reasonable simplification for languages that are mostly agglutinative. Moreover, evaluation of morphological segmentations is simple: segmentation is equivalent to binary classification of each potential segmentation point either as a morph boundary or not a

morph boundary. It is straightforward to calculate precision and recall for the predicted boundaries given the reference segmentations. However, some consideration is needed if a word form has multiple alternative analyses. If only the reference has alternative analyses, it is sensible to select the one that gives the maximal score. Otherwise, the alternatives in the predicted and reference analyses can be optimally matched with the Hungarian algorithm (Kuhn, 1955; Munkres, 1957). This approach is used as a baseline evaluation method in Publication VI and called *boundary precision and recall* (BPR).

Evaluation methods for unsupervised morphological analyzers have not been considered until recently. The first method we are aware of was developed for Morpho Challenge 2007 and slightly refined for the subsequent Morpho Challenges. The limitations of these methods were analyzed by Spiegler and Monson (2010), who also proposed a more robust method. The first large-scale comparison of the methods is provided in Publication VI. This section first discusses different approaches to the evaluation problem and then summarizes the methods and results of Publication VI.

6.3.2.1 Information-theoretic measures

Consider the predicted morphological analysis of a word as a random variable X and the reference analysis as Y. In item-and-arrangement morphology, they would be sequences of morphemes from the disjoint sets of morpheme labels, M_x and M_y, respectively. Let $M_x = |\mathrm{M}_x|$ and $M_y = |\mathrm{M}_y|$. If the ordering of the morphemes inside a word is disregarded, X and Y can be encoded as non-negative vectors $\boldsymbol{x} \in \mathbb{Z}_*^{M_x}$ and $\boldsymbol{y} \in \mathbb{Z}_*^{M_y}$, where x_i (y_i) is the number of occurrences of morpheme $m_i \in \mathrm{M}_x$ (M_y) and $\mathbb{Z}_* = \{0\} \cup \mathbb{Z}_+$. A further simplification is to consider them as binary vectors.[6]

A theoretically motivated way to define how similar the predicted analyses are to the reference analyses is to measure the mutual information

$$I(X;Y) = E_{X,Y}\Big[\log \frac{p(x,y)}{p(x)p(y)}\Big]. \tag{6.8}$$

As a single measure of similarity that is bounded only by the entropies $H(X)$ and $H(Y)$, mutual information is not a very informative measure. However, Rosenberg and Hirschberg (2007) have proposed a mutual information based *V-measure*, which resembles the standard F-measure. It is the harmonic mean of homogeneity h, which is analogous to precision, and completeness c, which is analogous to recall (cf. Figure 2.6):

$$h = \frac{I(X,Y)}{H(X)}; \qquad c = \frac{I(X,Y)}{H(Y)}. \tag{6.9}$$

If the variables are discrete and one-dimensional, both entropy and mutual information can be computed over the clustering that they define over the data samples (Meila, 2003). Such clustering tasks include document clustering (Rosenberg and Hirschberg, 2007), morphological clustering, and part-of-speech tagging. Christodoulopoulos et al. (2010) compare several evaluation measures for unsupervised POS tagging, and find the V-measure to be the most stable one.

[6] The binary assumption may be a problem for languages where the same morpheme can occur more than once in a word because of reduplication or compounding. For example, Finnish word **maa+n+kuore+n** ("of earth's crust") contains two genitives marked by the suffix **n**.

Apart from morphological clustering and POS tagging, there are also other subtasks of morphology induction where information-theoretic measures can be useful. For example, Chan (2006) applies entropy-based measures to evaluate paradigmatic signatures using POS tags as reference data.

However, mutual information is infeasible to estimate even in the simplest representation of a morphological analysis discussed above, binary vectors. The first problem is how to estimate $p(X)$ and $p(Y)$: it is unrealistic to assume that the morphemes are independent, but there is hardly enough data to do anything else.[7] The second problem is that in order to calculate the expectations, one has to sum over 2^{M_x} and 2^{M_y} possible configurations of morphemes, which is practically impossible for any reasonable-sized morpheme lexicons. In fact, the situation is very similar to that of the VSM evaluation presented in Section 5.4, but now the feature vectors are discrete.

6.3.2.2 Graph and matrix representations

All the evaluation methods that are considered later in this section disregard the order of the morphemes. In this case, there are two equivalent representations for the analyses of a set of V words W. The first is a matrix $\mathbf{X} \in \mathbb{Z}_*^{M \times V}$, where x_{ij} indicates the number of times morph m_i occurs in word w_j. The second, as proposed by Spiegler and Monson (2010), is a *bipartite graph* $G = (\mathrm{M}, \mathrm{W}; \mathrm{E})$. The graph has two disjoint sets of vertices, morphemes $\mathrm{M} = \{m_1, \ldots, m_n\}$ and words $\mathrm{W} = \{w_1, \ldots, w_m\}$, and edges $e(m_i, w_j) \in \mathrm{E}$ that connect vertices in M to vertices in W. The edges can have weights that indicate how many times the morpheme m_i occurs in the word w_j.

Considering predicted and reference analyses for the same set of words, the two respective bipartite graphs have the same word vertices but different morpheme vertices. For example, Figure 6.4 shows graphs for a linguistic reference analysis and a predicted analysis based on segmentations **app+ly**, **app+lie+s**, **applied**, **application**, **application+s**, **expir+ing**, **expir+ed**, **explain**, **expla+nation**, **expla+nation+s**, and **explain+ing** for the same set of words.

Evidently, the set of word vertices can be used as the common ground for comparing two analyses $\mathbf{X}$ and $\mathbf{Y}$. The analyses are equivalent if and only if $M_x = M_y = M$ and there exists a permutation $\pi : \{1, \ldots, M\} \mapsto \{1, \ldots, M\}$ such that $x_{ij} = y_{\pi(i)j}$ for all i and j. Then the corresponding graphs are *isomorphic*. Because of this notion, the graph-based evaluation methods are called isomorphic evaluations (Spiegler and Monson, 2010). However, the methods should determine *how* similar the analyses are, not whether they are equivalent.

The isomorphic evaluation methods suggested so far can be divided into two types. The methods of the first approach, called here *morpheme assignment*, explicitly match the reference morphemes and the predicted morphemes. The methods of the second approach, called *co-occurrence analysis*, study if the same number of morphemes are shared in $\mathbf{X}$ and $\mathbf{Y}$ by each pair of words.

6.3.2.3 Co-occurrence analysis

In co-occurrence analysis, a bipartite morpheme-word graph is transformed into a *word graph* by removing the morpheme vertices and replacing each pair of

[7] And estimating a dependency structure between morphemes is likely to be as hard as solving the problem of morphology induction in the first place.

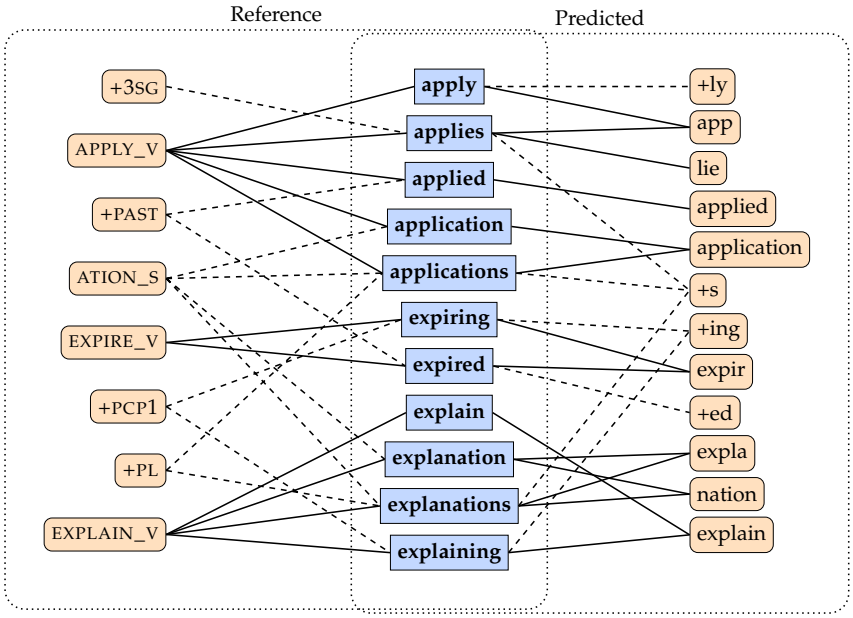


Figure 6.4. Example of a bipartite morpheme-word graph for a set of morphological gold standard analyses (left side) and segmentations (right side) of English words. An edge between a morpheme and a word indicates that the word contains the morpheme. Edges to suffixes are drawn with dashed lines.

edges $\big(e(m_i, w_j), e(m_i, w_k)\big)$ by edge $e(w_j, w_k)$. For a binary morpheme-word matrix $\mathbf{X}$, a word matrix equivalent to this is obtained by the product $\mathbf{X}^{\mathrm{T}}\mathbf{X}$. Figure 6.5 illustrates the word graphs corresponding to the analyses in the bipartite graph of Figure 6.4. After this transform, the set of the vertices of the graphs are equivalent regardless of the original analysis. For studying the isomorphism of the graphs, it is enough to compare the edges E between the vertices.

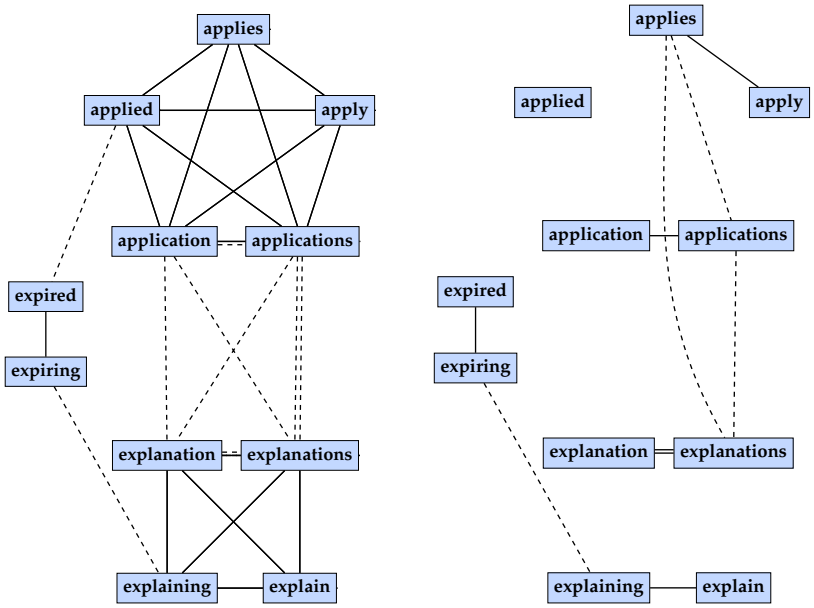
The precision and recall of the word graph $G_x = (\mathrm{W}; \mathrm{E}_x)$ against the reference word graph $G_y = (\mathrm{W}; \mathrm{E}_y)$ can be calculated by taking the averages of precisions and recalls for single words:

$$\mathrm{Pre} = \frac{1}{V} \sum_{i=1}^{V} \frac{|\{w_j \in \mathrm{W} : e_x(w_i, w_j) \wedge e_y(w_i, w_j)\}|}{|\{w_j \in \mathrm{W} : e_x(w_i, w_j)\}|} \tag{6.10}$$

$$\mathrm{Rec} = \frac{1}{V} \sum_{i=1}^{V} \frac{|\{w_j \in \mathrm{W} : e_x(w_i, w_j) \wedge e_y(w_i, w_j)\}|}{|\{w_j \in \mathrm{W} : e_y(w_i, w_j)\}|} \tag{6.11}$$

Note that the definitions are symmetric: precision of $\mathbf{X}$ against $\mathbf{Y}$ is the recall of $\mathbf{Y}$ against $\mathbf{X}$. A low recall in co-occurrence based metrics indicates missed co-occurrences (undersegmentation, allomorphs not identified), and a low precision indicates spurious co-occurrences (oversegmentation, syncretism not identified). For example, for the predicted analysis in Figure 6.5, $\mathrm{Pre}(\textbf{applies}) = 2/3$ and $\mathrm{Rec}(\textbf{applies}) = 2/4$. The precision is penalized because the plural **s** and 3rd person singular **s** are not separated. The recall is penalized because of the undersegmentation of **applied**, **application**, and **applications**.

Co-occurrence analysis has been used by Schone and Jurafsky (2000, 2001), Baroni et al. (2002), and Snover et al. (2002) for evaluating morphological clustering, but the first known method for evaluating full morphological analysis was developed for Morpho Challenge 2007 (Kurimo et al., 2008) and slightly re-



(a) Word graph for the reference analysis (b) Word graph for the predicted analysis

Figure 6.5. Example of a word graph for a set of morphological analyses of English words. An edge between two words indicates a co-occurring morpheme. Edges corresponding to suffixes are drawn with dashed lines.

vised for Morpho Challenge 2009 (Kurimo et al., 2010c). The Morpho Challenge evaluation (called briefly *MC*) assumes that there is a large set of words with reference analysis available, and does not try to consider all the edges between the words. Instead, it first samples randomly a number of *focus words*. Then for each morpheme of each focus word, another word that has the same morpheme is sampled. The result is a set of edges—that is, word pairs that have at least one morpheme in common. For precision, the edges are sampled from the predicted analysis and compared with the reference analysis. For recall, the edges are sampled from the reference analysis and compared with the predictions. The scores are normalized so that each focus word has the same weight on the overall score.

While the sampling approach of MC is well-motivated for large graphs that cannot be compared as a whole, it has two drawbacks: First, the word pairs that are sampled for calculating precision are dependent on the predicted analyses. Then two different algorithms will have different word pairs in the evaluation. Second, the approach is inconvenient if the reference set has only a small number of words, as it does not use all the information in the known analyses.

To account for ambiguous word forms, the MC evaluation is designed to give full precision if any of the reference analyses matches the predicted analysis. That is, the one that gives the highest precision is selected. If there are multiple predicted analyses, precision is calculated as the average over them. When recall is calculated analogously, it means that predicting multiple analyses provides the best recall and the average precision over them (Kurimo et al., 2010a). This is the third, and probably the most serious problem of the method: recall can be artificially boosted by adding alternative analyses for the predictions (Spiegler and Monson, 2010). The limitations of the MC evaluation have been analyzed in more detail by Spiegler (2011).

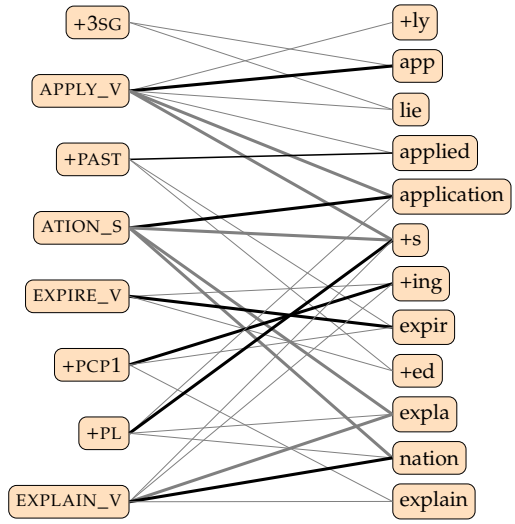


Figure 6.6. Example of a bipartite morpheme graph for reference (left) and predicted (right) morphemes of English words. An edge between two morphemes indicates that there is one (thin lines) or two (thick lines) words that have the left morpheme in the reference analysis and the right morpheme in the predicted analysis. Black lines show one possible assignment that maximizes the target criterion in EMMA.

6.3.2.4 Morpheme assignment

The evaluations based on morpheme assignment find a one-to-one or one-to-many assignments between the predicted and reference morphemes. Given an assignment, it is trivial to calculate precision and recall for the set of morphemes predicted for each word. The main problem is thus determining the assignment.

The problem can be illustrated by constructing another bipartite graph, where one set of vertices correspond to the reference morphemes and another set of vertices to the predicted morphemes. That is, given morpheme-word graphs for the reference and the predictions, the morpheme graph can be formed by removing the word vertices and replacing each pair of edges $\big(e_x(m_i, w_k), e_y(m_j, w_k)\big)$ by $e(m_i, m_j)$. The weight c_{ij} of the new edge is the number of replaced edges:

$$c_{ij} = |\{w_k \in \mathrm{W} : e_x(m_i, w_k) \wedge e_y(m_j, w_k)\}|. \tag{6.12}$$

For binary $\mathbf{X}$ and $\mathbf{Y}$, the matrix $\mathbf{C}$ is obtained also by $\mathbf{X}\mathbf{Y}^{\mathrm{T}}$. Figure 6.6 shows the morpheme graph for the analyses of Figure 6.4.

The evaluation method proposed by Spiegler and Monson (2010), *EMMA*, uses a one-to-one assignment. That is, each predicted morpheme is matched to at most one morpheme in the reference and vice versa. Matching two morphemes that have an edge increases both precision and recall. The larger the weight of the edge is, the larger the increase. Thus the task is to select such an assignment that maximizes the sum of the weights of the selected edges. Mathematically, the problem is defined by:

$$\arg\max_{\mathbf{B}} \sum_{i,j} (c_{ij} \times b_{ij}) \quad \text{s.t.} \quad \sum_i b_{ij} \leq 1, \sum_j b_{ij} \leq 1, b_{ij} \in \{0, 1\}, \tag{6.13}$$

where $\mathbf{B}$ is a binary assignment matrix. Each $b_{ij} = 1$ indicates that morpheme m_i in the predicted analysis is matched to morpheme m_j in the reference analysis. The black edges in Figure 6.6 indicate one of the several assignments that maximizes the criterion for the example graph.

To account for several alternative analyses per word, c_{ij} is redefined as the average over all the combinations of the alternative analyses. After obtaining **B** from Equation 6.13, the best one-to-one match between the alternatives is optimized.

The main drawback in EMMA is the time complexity required for solving Equation 6.13. Using the Hungarian algorithm, the complexity is $O(M^3)$, where $M = \max(M_x, M_y)$. This limits the usage of the evaluation method for large data sets.

6.3.2.5 New methods for linguistic evaluation

The evaluation methods considered above have their own strengths and weaknesses: the MC evaluation is quick but prone to gaming, while EMMA is robust but slow to run. Publication VI presents two new methods that overcome the problems: *CoMMA* (or *Co-occurrence-based Metric for Morphological Analysis*) is designed to be reliable and suitable for both small and large data sets, while *EMMA-2* is a assignment-based method that is considerably faster than EMMA.

CoMMA. There are two main differences between the MC and CoMMA evaluations. First, CoMMA considers all vertices and edges in a given word graph. Second, it treats the alternative analyses of words in a manner that cannot be exploited as easily as in MC. Actually, Publication VI presents two different versions, CoMMA-B and CoMMA-S, that differ in how the alternative analyses are processed.

Let **X** and **Y** be the morpheme-word matrices of the predicted and reference analyses, respectively. In essence, CoMMA compares the similarity of the word matrices $\mathbf{P} = \mathbf{X}^{\mathrm{T}}\mathbf{X}$ and $\mathbf{R} = \mathbf{Y}^{\mathrm{T}}\mathbf{Y}$. Each p_{ij} give the number of morphemes that co-occur in the predicted analyses of the words w_i and w_j. For non-binary **X**, p_{ij} can be defined as $|\mathrm{P}_j \cap \mathrm{P}_i|$, where P_j and P_i are the morphemes of w_i and w_j, respectively. The construction of **R** is analogous.

For the error in **P**, one could consider any matrix norm, for example the ℓ_1-distance $|\mathbf{P} - \mathbf{R}|_1 = \sum_i \sum_j |p_{ij} - r_{ij}|$. However, the standard measures of precision and recall are often more informative. Let the number of words with at least one common morpheme with word i be $n_i = |\{j : p_{ij} > 0\}|$ and $m_i = |\{j : r_{ij} > 0\}|$, and the number of words that have at least one common morpheme with *any* word $v_p = |\{i : n_i > 0\}|$ and $v_r = |\{i : m_i > 0\}|$. The overall precision and recall are then

$$\mathrm{Pre} = \frac{1}{v_p} \sum_{i:n_i>0} \frac{1}{n_i} \sum_{j:p_{ij}>0} \frac{\min(p_{ij}, r_{ij})}{p_{ij}}; \tag{6.14}$$

$$\mathrm{Rec} = \frac{1}{v_r} \sum_{i:m_i>0} \frac{1}{m_i} \sum_{j:r_{ij}>0} \frac{\min(r_{ij}, p_{ij})}{r_{ij}}. \tag{6.15}$$

For example, if there are two morphemes that are shared between words i and j in the predicted analyses ($p_{ij} = 2$) and one morpheme in the reference analyses ($r_{ij} = 1$), the precision increases 0.5 point per $v_p \times n_i$ and the recall increases one point per $v_r \times m_i$. One option is to set the diagonals of the matrices **P** and **R** to zeros, that is, $p_{ii} = r_{ii} = 0$ for all i. This excludes words that do not have a common morpheme with any other words from the evaluation.

If there are alternative analyses of words, the simplest option is to take the union of the edges of the analyses when determining **P** and **R**. That is, if P_{ik} is

the k^{th} alternative for the i^{th} word in the predicted analyses, and R_{il} similarly for the reference analyses:

$$p_{ij} = \max_k \max_l |\mathrm{P}_{ik} \cap \mathrm{P}_{jl}|, \qquad r_{ij} = \max_k \max_l |\mathrm{R}_{ik} \cap \mathrm{R}_{jl}| \tag{6.16}$$

This ensures that adding more alternatives in the prediction will increase some p_{ij} values, thus generally improving recall but degrading precision. This method is called CoMMA-B0 (if the diagonals of **P** and **R** are set to zeros) or CoMMA-B1 (otherwise).

Another approach is to enforce one-to-one match between the alternatives similarly to BPR and EMMA. In this case, **P** and **R** are expanded so that there is one row for each analysis of each word. In order to keep the matrices easily comparable, the columns are not expanded. Instead, the number of edges between the k^{th} analysis of word i and the word j is set to be the maximum number of edges over the analyses of word j:

$$p_{(i_k)j} = \max_l |\mathrm{P}_{ik} \cap \mathrm{P}_{jl}|, \qquad r_{(i_k)j} = \max_l |\mathrm{R}_{ik} \cap \mathrm{R}_{jl}|, \tag{6.17}$$

where (i_k) denotes the index for the k^{th} analysis of the i^{th} word. The numbers of words with shared morphemes are $n_{ik} = |\{j : p_{(i_k)j} > 0\}|$ for the predicted analyses and $m_{ik} = |\{j : r_{(i_k)j} > 0\}|$ for the reference analyses. Let $o_i = |\{k : n_{ik} > 0\}|$ and $q_i = |\{k : m_{ik} > 0\}|$ be the number of alternative analyses for word i in predicted and reference analyses, respectively. Now v_p and v_r are defined as $v_p = |\{i : o_i > 0\}|$ and $v_r = |\{i : q_i > 0\}|$ and the overall precision and recall are

$$\text{Pre} = \frac{1}{v_p} \sum_{i:o_i>0} \frac{1}{o_i} \max_{\mathbf{A}_i} \sum_{k:n_{ik}>0} \frac{1}{n_{ik}} \sum_{j:p_{ikj}>0} a_{ikl} \times \frac{\min(p_{(i_k)j}, r_{(i_l)j})}{p_{(i_k)j}}; \tag{6.18}$$

$$\text{Rec} = \frac{1}{v_r} \sum_{i:q_i>0} \frac{1}{q_i} \max_{\mathbf{A}_i} \sum_{k:m_{ik}>0} \frac{1}{m_{ik}} \sum_{j:r_{ikj}>0} a_{ikl} \times \frac{\min(r_{(i_k)j}, p_{(i_l)j})}{r_{(i_k)j}}. \tag{6.19}$$

$\mathbf{A}_i$ is an assignment matrix between predicted and reference alternatives of the i^{th} word. That is, we want $\sum_k a_{ikl} \leq 1$, $\sum_l a_{ikl} \leq 1$, and $a_{ikl} \in \{0, 1\}$ for all i, k, and l. The assignments can again be solved using the Hungarian algorithm. This method is called CoMMA-S0 or CoMMA-S1, depending on whether the diagonals of **P** and **R** are set to zero or not.

EMMA-2. EMMA-2 solves the assignment problem by replacing the single one-to-one assignment problem of EMMA with two many-to-one assignment problems. When calculating precision, several predicted morphemes may be assigned to one reference morpheme (many-to-one mapping). When calculating recall, several reference morphemes may be assigned to one predicted morpheme (one-to-many mapping). The intuition behind this approach is that failing to conflate two allomorphs (e.g., plural suffixes **-s** and **-es** in English) should not degrade precision—as it would not in an IR task. Similarly, failing to distinguish between homographs (e.g., plural **-s** and 3rd person singular **-s** in English) should not degrade recall.

The modified assignment problems in EMMA-2 are

$$\mathbf{B}_{\text{Pre}} = \arg\max_{\mathbf{B}} \sum_{i,j} (c_{ij} \times b_{ij}) \quad \text{s.t.} \quad \sum_j b_{ij} \leq 1, b_{ij} \in \{0, 1\}; \tag{6.20}$$

$$\mathbf{B}_{\text{Rec}} = \arg\max_{\mathbf{B}} \sum_{i,j} (c_{ij} \times b_{ij}) \quad \text{s.t.} \quad \sum_i b_{ij} \leq 1, b_{ij} \in \{0, 1\}, \tag{6.21}$$

where $\mathbf{B}_{\text{Pre}}$ and $\mathbf{B}_{\text{Rec}}$ contain the assignments for calculating precision and recall, respectively. Because the best match for each reference or predicted morpheme can be selected independently of the others, solving these two problems is very simple. For precision, $b_{ij} = 1$ only if $j = \arg\max_j c_{ij}$. For recall, $b_{ij} = 1$ only if $i = \arg\max_i c_{ij}$. The time complexity is $O(M_x M_y)$ for M_x predicted and M_y reference morphemes.

6.3.2.6 Experiments

One important aspect of the direct evaluation methods is that their results should correlate with the results of indirect evaluations. Studying this aspect requires a large database of results based on different algorithms. Such a database has been collected in the Morpho Challenge competitions.

While the results of the Morpho Challenges include already about fifty algorithms, five languages, and three applications, unfortunately not every algorithm has been evaluated on every language and task. For example, the data sets of Morpho Challenge 2005 differed significantly from those in the following Challenges and thus the results had to be excluded. The same goes for Arabic results: while Arabic data was provided in two Challenges, the only task was the linguistic evaluation and the two data sets were different. Table 6.2 shows the number of evaluated methods for the remaining languages and tasks.

Table 6.2. The number of methods evaluated in different tasks and languages. Boundary evaluation gives the number of methods that could be evaluated by measuring the precision and recall of the morph boundaries (BPR). No reference segmentations for German were available, so boundary evaluations could not be applied to it.

Evaluation	Number of methods			
	English	Finnish	German	Turkish
Linguistic evaluations:				
• Isomorphic evaluations	49	42	39	45
• Boundary evaluation	20	18	–	20
Information retrieval	36	31	25	–
Statistical machine translation	–	22	13	–

Apart from the correlations between the results of different evaluations, Publication VI considers four other aspects of the evaluation methods: (1) how robust they are with respect to gaming, (2) how useful they are for identifying the strengths and weaknesses of the evaluated algorithm, (3) how quickly they can be computed, and (4) how stable they are with respect to the size of the evaluation data.

Correlations with application evaluations. Figure 6.7 shows Spearman's rank correlations between the F-score of the isomorphic evaluations and the scores (MAP or BLEU) of the IR and SMT evaluations. Among the IR tasks, EMMA and EMMA-2 have the best overall correlations, while CoMMA-B0 and B1 have trouble with the Finnish and German tasks. For the two SMT tasks, only EMMA provides positive correlations—all CoMMA methods actually give moderate *negative* correlations for the German task.

However, the situation is not as problematic if weighted F_β-scores are considered instead of balanced F_1-scores. Evidently different applications may prefer different balance for precision and recall. In other words, some types of errors may be more serious than other types of errors.

The rank correlations with different values of β are shown in Figure 6.8. With

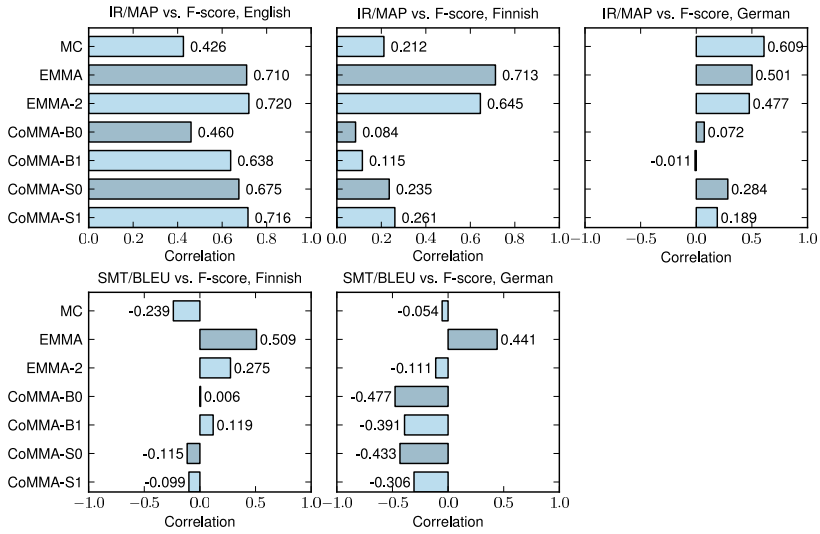


Figure 6.7. Spearman's rank correlations between the F-scores of the linguistic evaluation methods and the scores of information retrieval and statistical machine translation evaluations (Publication VI).

the optimal F_β, EMMA still yields the highest correlations for the IR tasks, but the best correlations for the co-occurrence-based metrics are much closer. For SMT, all the evaluations have the highest correlations for a low β, to the extent that simply the precision is sometimes the best predictor. It appears that none of the evaluated algorithms has too low a recall for this application.

Finding the weight β that maximizes the correlation between F_β and the application scores can be considered as tuning the evaluation metric. Then a relevant question is how general the found value is for the given language and application. That is, if the β is optimized for one set of algorithms, and then utilized for a second set of algorithms, are the correlations better than for the balanced score? This was tested for the English and Finnish IR tasks, for which the largest number of data points were available. All segmentation algorithms (17 for English and 15 for Finnish) were used to optimize the weights and then all non-segmentation algorithms (18 for English and 15 for Finnish) were used as a test set. The obtained β and correlations are shown in Table 6.3. The optimized F_β usually yields higher correlations also for non-segmentation algorithms than the balanced F_1-score. In the cases that it does not, F_β and F_1 are either equal or very close. Moreover, in half of the cases, the correlation of F_β tuned for the non-segmentation algorithms (shown in last column) is only slightly (≤ 0.05) higher than the one tuned for the segmentation algorithms. Thus using F_β-scores is clearly a feasible approach for the direct evaluations.

Correlations with the boundary evaluation. In many cases, a linguistic reference does not include a segmentation but only morpheme labels, while the evaluated algorithm does only segmentation. If there is a high correlation between an isomorphic evaluation and a boundary evaluation, it is possible to substitute the isomorphic evaluation for the boundary evaluation.

Figure 6.9 shows correlations between the F-scores of BPR and the isomorphic evaluations. All the correlations are reasonable, but there is clear variation over the languages. Only EMMA and EMMA-2 provide high correlations in all of

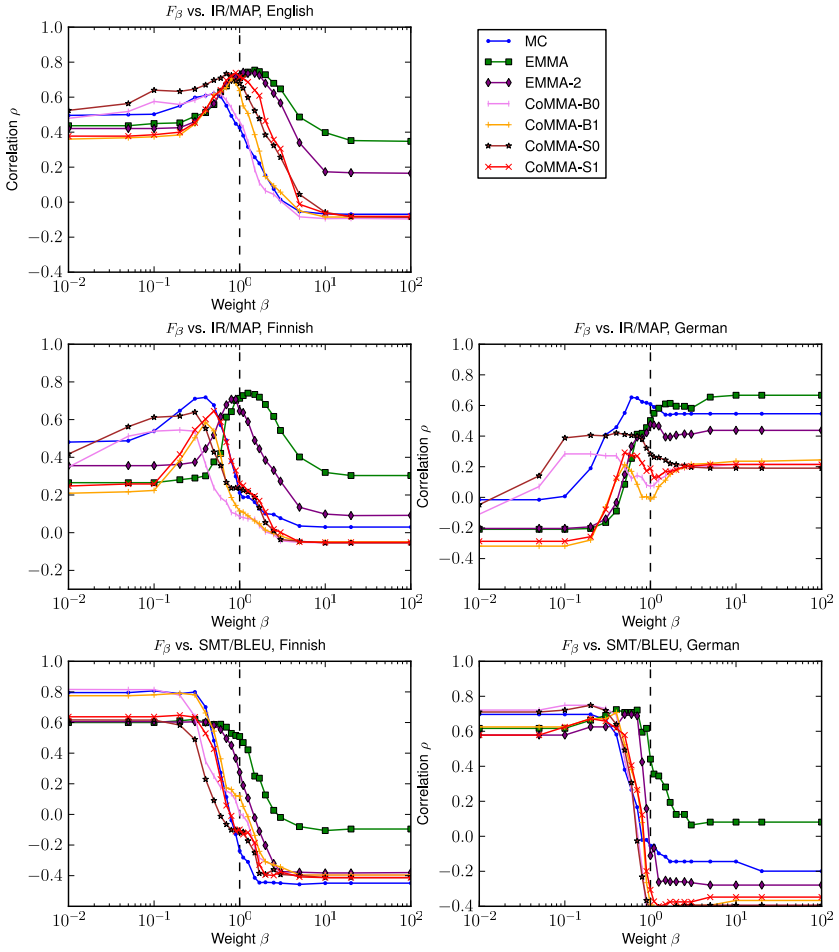


Figure 6.8. Spearman's rank correlations between the results of the application evaluations and weighted F_β-scores with varying β (Publication VI).

them. CoMMA-B1 and CoMMA-S1 have the best correlations in English, and MC in Turkish, but all of them have only moderate correlation in Finnish.

Robustness. The robustness of the evaluation methods with respect to gaming were addressed by two tests introduced by Spiegler and Monson (2010). The *ambiguity hijacking test* addresses if the evaluation method is robust in how it deals with alternative analyses: providing two alternative analyses for a non-ambiguous word should not give higher score than providing a reasonable combined analysis or just the better one. For example, *ParaMor-Morfessor*, which simply lists the analyses of ParaMor (Monson et al., 2008) and Morfessor Categories-MAP as two alternatives for each word, should not outperform *ParaMor-Morfessor Union* (Monson et al., 2010), which combines the morpheme boundary predictions as a single analysis. Figure 6.10 shows that MC and CoMMA-B give higher F-scores to ParaMor-Morfessor than to ParaMor-Morfessor Union, while EMMA, EMMA-2, and CoMMA-S are robust in this respect.

The *shared morpheme padding test* addresses the vulnerability of the evaluations to an artificial modification of the analysis. A unique bogus morpheme is added

Table 6.3. Correlations between IR results and F_β tuned for segmentation algorithms, calculated on non-segmentation algorithms. For comparison, the fifth column shows the correlation of balanced F-score for non-segmentation algorithms and the last column shows the correlation of the F_β-score optimized using the non-segmentation algorithms. The highest of each F_1 and F_β is marked with an asterisk.

Language	Method	Segmentation		Non-segmentation		
		β	F_β	F_1	F_β	best F_β
English	MC	0.6	0.62	0.43	0.56*	0.70
English	EMMA	1.25	0.87	0.73*	0.71	0.73
English	EMMA-2	1.1	0.86	0.74*	0.74*	0.75
English	CoMMA-B0	0.6	0.69	0.44	0.60*	0.72
English	CoMMA-B1	0.9	0.66	0.69	0.74*	0.76
English	CoMMA-S0	0.6	0.70	0.67*	0.65	0.82
English	CoMMA-S1	1.0	0.70	0.79*	0.79*	0.84
Finnish	MC	0.3	0.76	0.35	0.73*	0.73
Finnish	EMMA	1.25	0.91	0.59	0.61*	0.76
Finnish	EMMA-2	0.8	0.90	0.55*	0.53	0.67
Finnish	CoMMA-B0	0.3	0.65	0.18	0.47*	0.60
Finnish	CoMMA-B1	0.4	0.71	0.29	0.52*	0.52
Finnish	CoMMA-S0	0.3	0.74	0.44	0.61*	0.69
Finnish	CoMMA-S1	0.5	0.76	0.44	0.62*	0.66

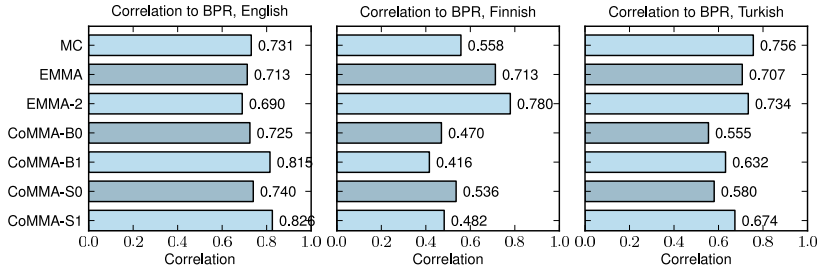


Figure 6.9. Spearman's rank correlations between the F-scores of the isomorphic evaluation methods and the BPR boundary evaluation (Publication VI).

to each predicted analysis. This would have no effect for a measure based on mutual information, but for methods based on co-occurrence analysis, it adds an additional edge between each word. In consequence, the recall scores are increased and the precision scores decreased.

In Publication VI, the shared morpheme padding was tested on 18 different algorithms. The average ratio of scores with and without padding was calculated. The increase of recall and decrease of precision was confirmed for the MC and CoMMA methods. For Finnish and Turkish, where high recall was hard to obtain, it improved the F-score (ratios between 1.19 and 1.76), while for English and German, where the recall was initially high, the F-score decreases (ratios between 0.23 and 0.84). EMMA-2 shows only small changes in the scores (ratios between 0.96 and 1.29), and EMMA even smaller (ratios between 0.76–0.86).

Interpretability. As defined by Spiegler and Monson (2010), interpretability of an evaluation method concerns how the evaluation results can be used for identifying the strengths and weaknesses of the predicted analyses. EMMA has the advantage of providing a mapping between the predicted and the reference morphemes. This is useful especially for human inspection of the results as it

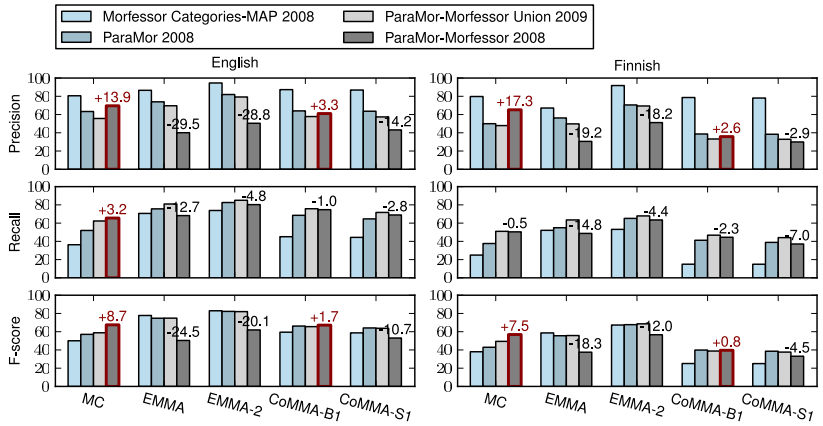


Figure 6.10. The results of Publication VI for gaming with ambiguity hijacking on English and Finnish data sets: ParaMor-Morfessor returns ParaMor and Morfessor Categories-MAP as two alternatives, whereas ParaMor-Morfessor Union combines the two predictions into a single analysis. The number above ParaMor-Morfessor 2008 shows the absolute difference to ParaMor-Morfessor Union. CoMMA-B0 provides similar results to B1 and S0 similar results to S1.

helps qualitative evaluation. The many-to-one assignments in EMMA-2 are not as easy to utilize. For example, they tend to have some obscure links for the morphemes that occur only once in the data.

Another aspect is whether the precision and recall of the evaluation method can provide useful information. As explained in Section 6.3.2.3, low recall in co-occurrence based metrics should be a indicator of not segmenting enough or not joining allomorphs, and low precision should be a indicator of segmenting too much or having the same label for different morphemes that are homographs. In contrast, one-to-one matching returns neither good precision nor recall if the number of predicted morphemes is wrong.

To study the behavior of precision and recall experimentally, Morfessor Baseline was trained with different likelihood weights, thus controlling the amount of segmentation (see Section 6.4.3). Then precision and recall was calculated for each evaluation method. This experiment showed that all co-occurrence-based methods, boundary evaluations, and EMMA-2 have recall and precision that consistently decrease and increase, respectively, when the words are segmented more. However, with EMMA, recall starts to decrease after a certain point, obstructing the interpretability.

Computational complexity. The average computation times of the evaluation methods for test sets of varying sizes are shown in Figure 6.11. As the sampling approach of the MC evaluation differs from the others, it was excluded from this experiment. The boundary evaluation (BPR), included for reference, is very fast and has in practice a linear complexity. All CoMMA variants show polynomial growth of the same order (linear growth and the same slope in the log-log scale). The largest evaluation set that could be used with EMMA with 16 gigabytes of memory was 2,000 words. The growth of the computation time is faster than with CoMMA, potentially exponential. EMMA-2 was very fast for the tested evaluation sets, but the super-linear trend in the log-log scale indicates exponential growth for it, too. The exponential growth in EMMA and EMMA-2 is an implementation issue, related to using the integer linear programming for mor-

pheme assignment (only in EMMA) and for matching the alternatives (in both). Using the Hungarian algorithm instead should provide a polynomial growth.

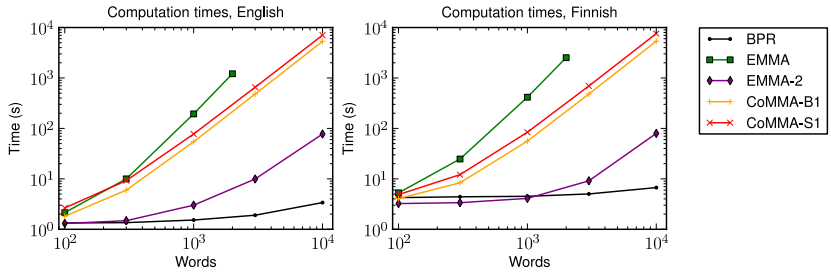


Figure 6.11. Computation times of different evaluation methods with respect to the size of the evaluation data (100-10,000 words) for English and Finnish. Both the time and the number of words are shown in logarithmic scale. The evaluated method is Morfessor Baseline. The computation times of CoMMA-B0 are comparable to B1 and S0 to S1.

Stability. A stable evaluation method should provide similar results for different random samples of evaluation data. In particular, it should not badly underestimate or overestimate the scores even if the evaluation set is small. In order to study the stability of the evaluation methods with respect to the size of the evaluation data, the evaluation scores were calculated for Morfessor Baseline using random sets of words, their size varying from 100 to 10,000. Figure 6.12 shows the mean and standard deviation for the precision, recall, and F-measure. The most stable method is, unsurprisingly, BPR. The MC evaluation shows more variation: for Finnish, all scores are underestimated with small data sets, while for English, they are first overestimated and then underestimated. EMMA and EMMA-2 give smaller standard deviations than the other methods, but they clearly overestimate the scores with small data sets. For CoMMA, the S0 and B0 variants that exclude isolated words show a similar pattern as the MC evaluation, but the changes are smaller especially for the recall in Finnish. Variants that include isolated words overestimate the scores with small data sets, but in contrast to EMMA and EMMA-2, the changes seem to get smaller as the size of the data grows.

6.3.2.7 Conclusions

Table 6.4 shows an indicative overview of the advantages and drawbacks of the studied evaluation methods. Based on the results, the new evaluation methods are clearly useful alternatives or replacements for their predecessors.

EMMA-2 maintains the strengths of EMMA—robustness and high correlation with application evaluations—while having substantially shorter computation times. The use of soft (many-to-one) assignments instead of the hard (one-to-one) assignment of EMMA reduces the interpretability of the morpheme assignments, but increases the interpretability of precision and recall. The combination of robustness and efficiency makes EMMA-2 a strong candidate for any large-scale experiments and competitions.

CoMMA, especially the CoMMA-S versions, fix the two main problems in the MC evaluation. First, they remove the need of sampling and are thus more suitable to use with small evaluation sets. Second, they deal with alternative analyses in a robust manner. Compared with EMMA and EMMA-2, CoMMA-S

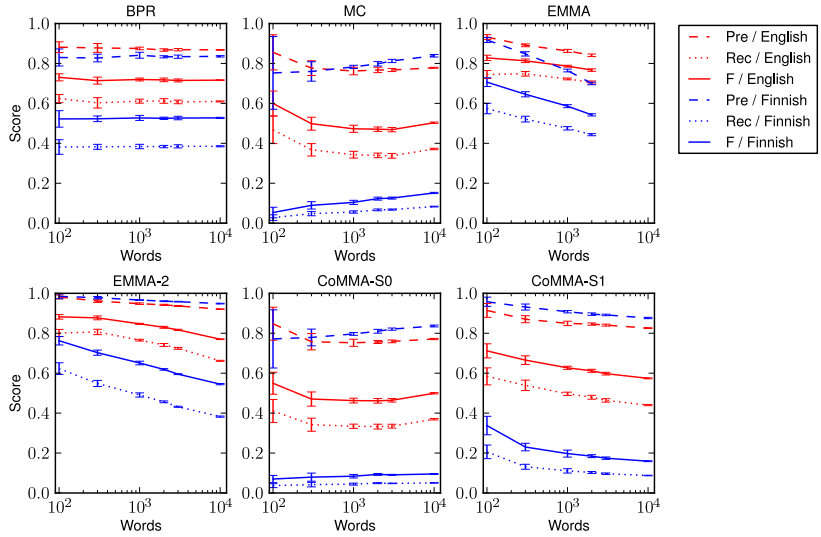


Figure 6.12. The mean and standard deviation for precision, recall, and F-score of different evaluation methods with respect to the size of the evaluation data (100-10,000 words) for English and Finnish. The evaluated method is Morfessor Baseline. CoMMA-B0 and CoMMA-B1 are not shown, as they give very similar results to CoMMA-S0 and CoMMA-S1, respectively.

Table 6.4. Overview of the advantages and drawbacks of the evaluation methods. Two plusses may be interpreted as an excellent result and two minuses as a serious weakness. CoMMA-B1 and CoMMA-S1 have slightly better correlations but not as good stability as CoMMA-B0 and CoMMA-S0, respectively.

	MC	CoMMA-B0	CoMMA-S0	EMMA	EMMA-2
Correlations	+	+	+	++	++
Robustness, ambiguity	−−	−	++	++	++
Robustness, padding	−	−	−	+	+
Time complexity	++	+	+	−−	+
Stability	+	+	+	−	−
Interpretable Pre and Rec	+	+	+	−	+
Provides label assignment	−	−	−	++	+

loses in the strength of the correlations with application evaluations, in particular with balanced F-scores and morphologically rich languages. However, it can still be recommended for English (and probably other mostly analytic languages), where it works as well as the assignment-based methods in practically all aspects. The advantage of CoMMA-S0 over EMMA or EMMA-2 is a better stability with respect to the size of the evaluation set, which helps comparing the results from a small development set with those of the final test set.

A part of the evaluation results used in Publication VI are shown in Appendix A.2. The full result tables are available from `http://research.ics.tkk.fi/events/morphochallenge/`.

6.3.3 Psycholinguistic evaluation

Apart from the computational linguistics, the questions of processing morphologically complex words has been considered in cognitive sciences. Psycholinguists are intrigued by how we are able to recognize and produce words and

how they are stored in our mind in the so-called mental lexicon (Baayen, 2007). The debate between the proponents of the symbolic models and the connectionist models has prevailed also within this topic (see, e.g., Pinker and Ullman, 2002; McClelland and Patterson, 2002; Albright and Hayes, 2003).

One important manner in which psycholinguistics is related to the field of computational linguistics is that both build models that are experimentally tested on data sets. However, the models differ in their purposes and the data sets in their type and size. While psycholinguists use behavioral measures such as response times and error rates, computational linguists use large amounts of raw or annotated text. For NLP research, the goal is engineering-oriented: it is important that the model works in the NLP applications, not whether it can say anything about language in the human mind. In contrast, psycholinguistic models should do exactly the latter (cf. Norris, 2005).

While the two lines of research, NLP and cognitive science, have been quite distinct, there is a clear motivation for bridging the gap. For cognitive scientists, explicit, machine-implemented models yield quantitative predictions that may be tested against measured values of performance and brain activation. For NLP research, psycholinguistic data provide new ways to evaluate the learning algorithms. As humans evidently are efficient users of their own languages, psycholinguistic evaluations should be at least as relevant for the development of the models as the linguistic evaluations are.

Psycholinguistic evaluations can be considered as direct evaluations, although the external "reference" is not an analysis by linguists but something recorded from human test subjects. As mentioned in Section 5.3, such evaluations have been used also for vector space models (e.g., Lund and Burgess, 1996; Jones et al., 2006).

There are at least three type of psycholinguistic data that have been applied to evaluate models of morphological processing or learning. The first type of data comes from the studies of the learning process in child language acquisition. In the famous work, Rumelhart and McClelland (1986) find a similar stage of learning observed in human children, where irregular verbs are regularized, from the learning curves of their neural network model. As in most work of this type, Rumelhart and McClelland (1986) use supervised learning. Recently, Lignos et al. (2010b) have used a similar approach for studying their unsupervised model of morphological acquisition. In particular, they consider whether the order in which different inflectional rules are acquired is similar.

The second type of data comes from how adults generalize the morphological processes to novel word-like utterances. For example, Albright and Hayes (2003) use so called "wug tests", where new words (e.g., **spling**) is presented to test subjects, and they have to either produce or rate suitable inflected tenses (e.g., **splinged** or **splung**).

A third option, considered in Publication VII, is to use the model to predict quantitative measurements from behavioral studies. As discussed in Section 3.1.7.6, a typical measurement is the processing latency in some task. For example, in a lexical decision task, test subjects have to decide as quickly and accurately as possible whether the letter string appearing on the screen is a real word or not, and to press a corresponding button. The time between the appearance of the string and the press is an indirect measure of the underlying mental processing. In general, longer reaction times reflect more effortful cognitive processing.

Such reaction time data has been applied to evaluate various types of computational models. For example, Norris (2006) proposes a stochastic model, the

Bayesian Reader, that is based on the assumption that human readers behave as optimal Bayesian decision-makers. The model is shown to explain with a high accuracy the effects of word frequency and orthographic neighborhood in word identification, lexical decision, and semantic classification tasks. However, the model was not intended to explain any other effects, such as those related to morphology. Moreover, it was tested on fixed-length words only.

As a second example, Lim et al. (2005) study a trie data structure for storing Korean words. They found that the search times correlated with three properties of words and non-words (word frequency, word length, and the similarity of non-words to a correct word) in a similar manner as human reaction times. Although the trie structure has some connections to LSV type of segmentation, their model does not perform any morphological analysis—it just stores the word strings.

As a third example, Baayen et al. (2011) have reported a large set of experiments using a two-layer neural network model called *naive discriminative reader*. It is based on the equilibrium equations of the Rescorla-Wagner model (see, e.g., Rescorla, 2007). The model does not try to find morphemes of the words, but directly ties the orthographic cues to given abstract meaning labels. Baayen et al. (2011) show that the activations of the correct abstract meanings for a given input correlate with the reaction times in lexical decision. As most of the models tested for reaction time data, the naive discriminative reader is trained in a supervised manner.

Publication VII presents, to our knowledge, the first attempt to evaluate an unsupervised model of morphological analysis using reaction time data. As models, it considers Morfessor Baseline and Categories-MAP (see Section 6.4.1), and compares their predictions with those of single psycholinguistic factors known to affect reaction times as well as to the predictions of letter-based n-gram model.

6.3.3.1 Evaluation framework

The experimental setup of Publication VII can be described in three steps: data recording, model estimation, and model evaluation.

Data recording. The data are human reaction times to individual inflected and non-inflected Finnish nouns in a lexical decision task, recorded by Lehtonen et al. (2007). The participants were 16 Finnish-speaking adults. The stimuli included 320 real Finnish nouns, extracted from an unpublished Turun Sanomat newspaper corpus of 22.7 million word tokens by using a search program by Laine and Virtanen (1996), and the same number of pseudowords. The words included 80 high-frequency monomorphemic, 80 high-frequency inflected, 80 low-frequency monomorphemic and 80 low-frequency inflected words. The lengths and bigram frequencies (average frequency of letter bigrams) were similar for words and pseudowords. The length of the strings in letters varied between 4 and 11, with mean 6 and standard deviation 1.2. As preprocessing, all incorrect responses and reaction times of three standard deviations longer or shorter than each subject's mean were excluded. Then a logarithmic function was applied to the remaining data points and they were normalized to zero mean for each subject. Finally, the average across the test subjects was calculated for each word.

Model estimation. The unsupervised models were trained on the Finnish corpus from Morpho Challenge 2007 competition (Kurimo et al., 2008). Because

the word frequencies were known to have a large effect on the models (Section 6.4.3), three different models were trained for each model type: one trained on word types, one trained on word tokens, and one trained on word types, but weighting each type by its frequency dampened by the logarithmic function $f(x) = \ln(1 + x)$.

For comparison, psycholinguistic factors that are known to affect the reaction times were collected (see Section 3.1.7.6). These factors included word length, cumulative base frequency, surface frequency, and morphological family size.

Model evaluation. Morfessor and the n-gram models examined here estimate a probability distribution $p(W)$ over the words. To predict the reaction time of the word w, the self-information $-\log p(w)$ is used. Self-information from a word-based unigram model corresponds to logarithmic surface frequency of the word.

The models are evaluated by comparing how well the model output x correlates with the reaction times y obtained from humans. Pearson product-moment correlation coefficient $\rho_{XY} \in [-1, +1]$ is used as a measure of the correlation. For uncorrelated variables $\rho_{XY} = 0$, and high absolute values of ρ_{XY} indicate high predictive power. In addition to calculating the empirical estimate of the correlation coefficient (r_{XY}), the probability of the null hypothesis that X and Y are uncorrelated ($\rho_{X,Y} = 0$) is estimated by using the z-test on the Fisher transformation of r_{XY}.

For additional baseline result, the reaction times of individual test subjects were treated as a model, and thus correlated with those of the rest of the subjects.

6.3.3.2 Results

Table 6.5 shows the statistical models and psycholinguistic factors that had the highest correlations, all of them statistically significant ($p(\rho_{XY} = 0) < 0.01$). Among the factors, logarithmic frequencies yielded higher correlations than linear frequencies, and the highest ones were obtained for the number of morphemes in the word and the surface frequency. Among the models, the n-grams were best trained with word types, while training with the logarithmic frequencies yielded the highest correlations for Morfessor. The highest correlations were obtained for the letter 9-gram model trained with word types—increasing the n-gram length above 9 did not improve the results—and Morfessor Categories-MAP trained with dampened frequencies. Morfessor Baseline did only slightly worse. All of them had markedly higher correlations than the maximum correlation obtained for a single test subject to the average reaction times of the others as well as all individual psycholinguistic factors.

Whereas n-gram models simply estimate how probable the seen letter frequency is, Morfessor estimates a lexicon of sub-word units, morphs, that efficiently encode the observed data. Both model types yield similar correlations, but the models of Morfessor require fewer parameters (about 178 000 transition and emission probabilities in Categories-MAP) than the n-gram models (almost 6 million n-gram probabilities).

A scatter plot of the logarithmic reaction times and the log-probabilities given by the best Morfessor Categories-MAP model is shown in Figure 6.13. Observing the words that have poor match between the predicted processing cost and reaction time indicates that some of the unexplained variance is caused by a training corpus that does not match the material that humans are exposed to.

Table 6.5. The correlation coefficients r of different word statistics and models to average human reaction times estimated in Publication VII. The surface frequency is from the Morpho Challenge corpus used for training the models, and other statistics are from the Turun Sanomat newspaper corpus. The last row shows correlations between the reaction times of individual subjects and the average reaction times.

Psycholinguistic factors		Logarithmic	Linear
Surface frequency		−0.535*	−0.238
Base frequency		−0.445*	−0.190
Morphological family size		−0.423*	−0.292
Length (letters)		+0.218*	+0.216
Length (morphemes)		+0.542*	+0.542*
Statistical models	**Types**	**Log-frequencies**	**Tokens**
Letter 1-gram model	+0.182*	+0.182*	+0.180
Letter 3-gram model	+0.302	+0.303	+0.307*
Letter 5-gram model	+0.539*	+0.538	+0.516
Letter 9-gram model	+0.695*	+0.692	+0.636
Morfessor Baseline	+0.661	+0.677*	+0.582
Morfessor Categories-MAP	+0.662	+0.695*	+0.547
Other	**Minimum**	**Median**	**Maximum**
Reaction times of a single subject	+0.203	+0.477	+0.568*

For example, words that have faster reaction times than predicted are often very concrete and related to family, nature, or stories: **tyttö** (*girl*), **äiti** (*mother*), **haamu** (*ghost*), **etanaa** (*snail* + partitive case), **norsulla** (*elephant* + adessive case). Similarly, words that have slower reaction times than predicted are often more abstract or professional: **ohjelma** (*program*), **tieto** (*knowledge*), **hankkeen** (*project* + genitive case), *käytön* (*usage* + genitive case), **hiippa** (*miter*), **kapselin** (*capsule* + genitive case).

A few additional models were trained to study the effect of the training corpus. To study the effect of the corpus size, models were trained on 30 000, 100 000, 300 000 and one million sentence random subsets of the Morpho Challenge corpus. In addition, models were trained on three smaller corpora that have different type of material: books (4.4 million words) and periodicals (2.1 million words) from the Finnish Parole corpus[8], subtitles of movies (3.0 million words) from the OpenSubtitles corpus[9] (Tiedemann, 2009), and their combination. Figure 6.14 shows that increasing the number of word types in the corpus clearly improves the correlation between model predictions and measured reaction times. However, the data from books, periodicals and subtitles usually give higher correlations than the same amount of the Morpho Challenge data.

6.3.3.3 Discussion

The experiments showed that the probabilistic model of Morfessor Categories-MAP is able to accurately predict reaction times for Finnish nouns. Moreover, the results of the experiments indicate that even higher correlations would be obtained if there were a data set more similar to what human observe in the course of their life. While such data is hard to obtain, an unsupervised algorithm makes it also possible to study, for example, acquisition of an artificial language so that the input is the same for both the learning method and the test subjects.

[8] By Department of General Linguistics, University of Helsinki and Research Institute for the Languages of Finland (gatherers), 1998.

[9] Data extracted from `http://www.opensubtitles.org/`.

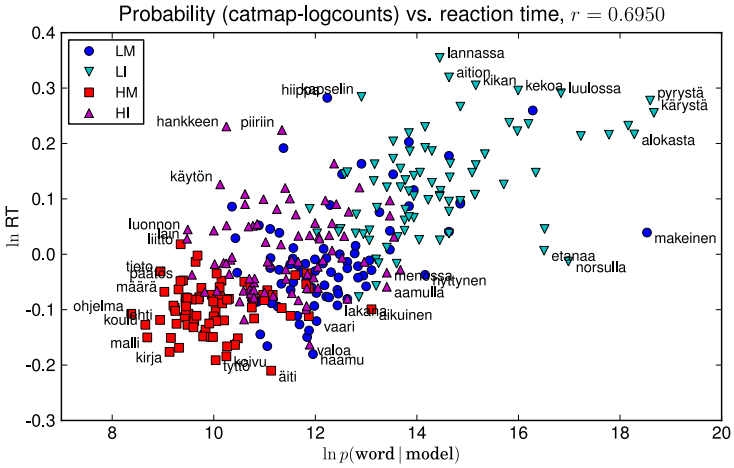


Figure 6.13. Scatter plot of reaction times and log-probabilities from Morfessor Categories-MAP (Publication VII). The words are divided into four groups: low-frequency monomorphemic (LM), low-frequency inflected (LI), high-frequency monomorphemic (HM), and high-frequency inflected (HI).

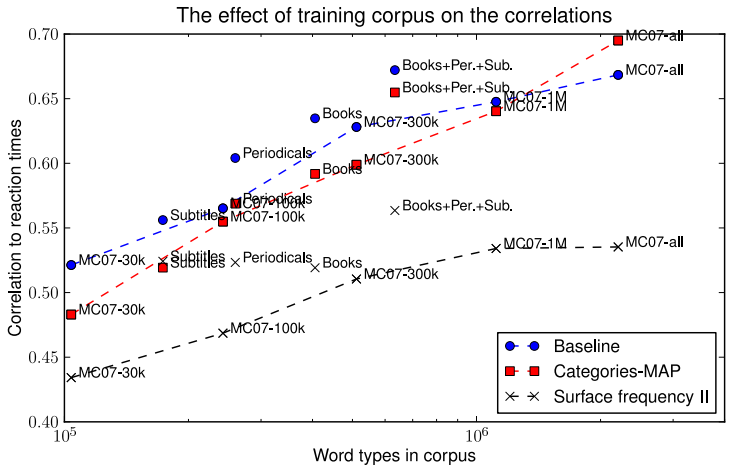


Figure 6.14. The effect of training corpus on correlations of Morfessor Baseline (blue circles), Categories-MAP (red squares), and logarithmic surface frequencies (black crosses) found in Publication VII. The dotted lines show the results on subsets of the Morpho Challenge 2007 corpus. Unconnected points show the results using different types of corpora (books, periodicals, subtitles, and their combination).

Another observation was that a letter-based 9-gram model yielded correlations equally high to those of Morfessor, although with much larger number of model parameters. As the n-gram models provide very precise estimates on how probable a certain sequence is in the language, they may be good predictors especially for early visual processing stages. While the different stages of word processing cannot be identified from behavioral reaction times, brain activation measures could provide insight on where the predictive power of the models stems from.

6.4 MDL-inspired models for learning constructions

This section describes the contributions of Publications VIII, IX, X, and XI. These four publications are based on Morfessor (Creutz and Lagus, 2002; Creutz, 2003; Creutz and Lagus, 2004, 2005b,a; Creutz, 2006; Creutz and Lagus, 2007), a family of methods for unsupervised morpheme induction. Thus, before going into the new contributions, an overview of Morfessor is given. The overview and the mathematical notation used in this section is based on Publications IX and X, and it is slightly different from the one used in the original Morfessor articles. In particular, Morfessor is described within a more general framework of probabilistic generative models for item-and-arrangement (IA) morphology.[10]

6.4.1 Morfessor

Morfessor is a family of methods for unsupervised learning of morphology designed for agglutinative languages. As a parametric learning method, it can be characterized by three components: model, cost function, and training and decoding algorithms. While all of them depend on the particular version of Morfessor, some basic assumptions are shared across the Morfessor family. Among the different versions of Morfessor, this section concentrates on the two most popular: Morfessor Baseline (Creutz and Lagus, 2002, 2005b) and Morfessor Categories-MAP (Creutz and Lagus, 2005a, 2007).

6.4.1.1 Model

The models of the Morfessor family are generative probabilistic models that predict words W and their morphological analyses A given model parameters $\boldsymbol{\theta}$. That is, they define the joint probability distribution $p(A, W \mid \boldsymbol{\theta})$.[11,12] In an item-and-arrangement type of approach (Section 3.2.3), an analysis can be considered as a list of morpheme labels: $\boldsymbol{a} = (m_1, \ldots, m_n)$. In the case of Morfessor, the labels are morphs, non-overlapping segments of the words.

The probability of an analysis $\boldsymbol{a}$ for a given word $\boldsymbol{w}$ is obtained by

$$p(\boldsymbol{a} \mid \boldsymbol{w}, \boldsymbol{\theta}) = \frac{p(\boldsymbol{w} \mid \boldsymbol{a}, \boldsymbol{\theta}) \times p(\boldsymbol{a} \mid \boldsymbol{\theta})}{p(\boldsymbol{w} \mid \boldsymbol{\theta})}. \tag{6.22}$$

[10] IA and other linguistic models of morphology are discussed in Section 3.2.3.

[11] A discriminative model would define only $p(A \mid W, \boldsymbol{\theta})$, for example by predicting whether there is a morph boundary between each letter of the word. Such a model provides analysis for any given word, but is not able to generate new words.

[12] However, only some Morfessor versions define the joint distribution $p(A, W, \boldsymbol{\theta})$ that is required in Bayesian approaches.

It is sensible to assume that for fixed parameters $\boldsymbol{\theta}$, a certain analysis $\boldsymbol{a}$ produces only one word form. This is marked with the detokenization function $\phi^{-1}(\boldsymbol{a}, \boldsymbol{\theta})$. For Morfessor, $\phi^{-1}(\boldsymbol{a}, \boldsymbol{\theta})$ is simply the concatenation of the morphs in $\boldsymbol{a}$. Then

$$\begin{aligned} p(\boldsymbol{a} \mid \boldsymbol{w}, \boldsymbol{\theta}) &\propto p(\boldsymbol{w} \mid \boldsymbol{a}, \boldsymbol{\theta}) \times p(\boldsymbol{a} \mid \boldsymbol{\theta}) \\ &= \mathrm{I}(\phi^{-1}(\boldsymbol{a}, \boldsymbol{\theta}) = \boldsymbol{w}) \times p(\boldsymbol{a} \mid \boldsymbol{\theta}) \\ &= \mathrm{I}(m_1 \dots m_n = \boldsymbol{w}) \times p(m_1, \dots, m_n \mid \boldsymbol{\theta}), \end{aligned} \tag{6.23}$$

and the model is defined simply by the choice of $p(m_1, \dots, m_n \mid \boldsymbol{\theta})$ and the parameter space Θ. In Morfessor Baseline, morphs m_i are assumed to be independent, whereas in Morfessor Categories-MAP, the probability of $m_1, \dots, m_n$ is estimated by a hidden Markov model.

6.4.1.2 Cost function

Morfessor tries to find a single point estimate for the model parameters $\boldsymbol{\theta}$. In Baseline and Categories-MAP, the cost function is based on the MAP estimate (Equation 2.49, page 53). However, as the prior probability $p(\boldsymbol{\theta})$ is defined by a coding scheme inspired by the MDL principle, the cost function is equivalently that of the two-part MDL (Equation 2.58, page 55).

In the latest formulation of Morfessor (Creutz and Lagus, 2007), the model parameters $\boldsymbol{\theta}$ are divided into a *morph lexicon* $\mathcal{L}$ and *grammar* $\mathcal{G}$: $p(\boldsymbol{\theta}) = p(\mathcal{L}) \times p(\mathcal{G} \mid \mathcal{L})$. The lexicon includes the properties of the morphs and the grammar determines how the morphs can be combined to form words.

The priors of Morfessor assign a higher probability to lexicons that store fewer morphs, where the morph m_i is considered stored if $p(m_i \mid \boldsymbol{\theta}) > 0$. Let the number of stored morphs—that is, size of the lexicon—be μ. The probability of the lexicon is then

$$p(\mathcal{L}) = p(\mu) \times p(\text{properties}(m_1), \dots, \text{properties}(m_\mu)) \times \mu!, \tag{6.24}$$

where the factorial term is explained by the fact that there are $\mu!$ possible ways to order a set of μ items and the lexicon is equivalent for different orders of the same set of morphs. The properties of the morphs are further divided into those related to *form*—such as the string representations of the morphs—and *usage*—such as their frequencies. The priors for the morph properties and the grammar depend on the particular version of the model.

The second part of the cost function is likelihood of the training data. The training data $\boldsymbol{D}_W$ consists of word forms—either tokens of a corpus or only different word types. Assuming that the probabilities of the words are independent, the likelihood of the data is

$$\begin{aligned} p(\boldsymbol{D}_W \mid \boldsymbol{\theta}) &= \prod_{j=1}^{|\boldsymbol{D}_W|} p(W = \boldsymbol{w}_j \mid \boldsymbol{\theta}) \\ &= \prod_{j=1}^{|\boldsymbol{D}_W|} \sum_{\boldsymbol{a}} p(W = \boldsymbol{w}_j \mid A = \boldsymbol{a}, \boldsymbol{\theta}) p(A = \boldsymbol{a} \mid \boldsymbol{\theta}) \\ &= \prod_{j=1}^{|\boldsymbol{D}_W|} \sum_{m_1 \dots m_n \in \mathcal{L}^*} \mathrm{I}(m_1 \dots m_n = \boldsymbol{w}_j) p(m_1, \dots, m_n \mid \boldsymbol{\theta}). \end{aligned} \tag{6.25}$$

6.4.1.3 Training and decoding algorithms

Expectation-Maximization algorithm. In theory, Morfessor type of models can be trained with the EM algorithm. Given the old parameters $\boldsymbol{\theta}^{(t-1)}$, a new estimate of the parameters are obtained by:

$$\begin{aligned}\boldsymbol{\theta}^{(t)} &= \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t-1)}) \\ &= \arg\min_{\boldsymbol{\theta}} E_Y\left[L(\boldsymbol{\theta}, \boldsymbol{D}_W, Y) \mid \boldsymbol{D}_W, \boldsymbol{\theta}^{(t-1)}\right] \\ &= \arg\min_{\boldsymbol{\theta}} \sum_{\mathbf{Y}} p(\mathbf{Y} \mid \boldsymbol{D}_W, \boldsymbol{\theta}^{(t-1)}) L(\boldsymbol{\theta}, \boldsymbol{D}_W, \mathbf{Y}),\end{aligned} \tag{6.26}$$

where Y is a hidden variable that gives the assignments of the words $\boldsymbol{w}$ in the training data to their possible analyses $\Phi(\boldsymbol{w}) = \{\boldsymbol{a} : \phi^{-1}(\boldsymbol{a}) = \boldsymbol{w}\}$, and the MAP/MDL cost function is

$$\begin{aligned}L(\boldsymbol{\theta}, \boldsymbol{D}_W, \mathbf{Y}) &= -\log p(\boldsymbol{\theta}) - \log p(\boldsymbol{D}_W \mid \mathbf{Y}, \boldsymbol{\theta}) \\ &= -\log p(\boldsymbol{\theta}) - \log \prod_{j=1}^{|\boldsymbol{D}_W|} p(\boldsymbol{y}_j \mid \boldsymbol{\theta}).\end{aligned} \tag{6.27}$$

There are two problems in this approach. First, in the E-step, taking the expectation over all possible assignments $\mathbf{Y}$ in Equation 6.26 is generally infeasible. However, for some models, $\mathbf{Y}$ will yield to a simpler form. For example, if the morphs are assumed independent, the assignments will be to all substrings in the training data. For Markov or hidden Markov models, the Baum-Welch algorithm is applicable. The second problem is that there is no closed form solution to the M-step if the value of the cost function changes discontinuously with the number of non-zero morph probabilities. Testing all possible morph lexicons is clearly infeasible.

In consequence, the EM algorithm is straightforward to apply only with maximum-likelihood cost function and particular model types. For example, Deligne and Bimbot (1995) use Baum-Welch for their multigram model for word segmentation. Also maximum-likelihood versions of Morfessor, such as Categories-ML (Creutz and Lagus, 2004), which uses an HMM, can be trained with the Baum-Welch algorithm (Creutz, 2006, page 57). However, ML estimation requires heuristics to prevent overfitting. Moreover, maximum-likelihood EM cannot change the number of morphs stored in the lexicon: If $p(m_i \mid \boldsymbol{\theta}^{(t-1)}) = 0$, any $\boldsymbol{y}$ that contains m_i will have a zero probability. If $p(m_i \mid \boldsymbol{\theta}^{(t-1)}) > 0$, m_i is present in some $\boldsymbol{y}$, and the maximum-likelihood cost requires that also $p(m_i \mid \boldsymbol{\theta}^{(t)}) > 0$. In consequence, non-sparse initialization of the parameters is necessary—Deligne and Bimbot (1995) take all word sequences that have occurred at least four times—and the parameter vectors will also stay non-sparse.

Viterbi algorithm. A simple approximation to EM is to first take the most probable analysis $\phi_{\text{best}}(\boldsymbol{w}, \boldsymbol{\theta}^{(t-1)})$ for each word and then update the parameters to minimize the cost function:

$$\boldsymbol{\theta}^{(t)} = \arg\min_{\boldsymbol{\theta}} \left\{ -\log p(\boldsymbol{\theta}) - \log \prod_{j=1}^{|\boldsymbol{D}_W|} p\big(\phi_{\text{best}}(\boldsymbol{w}_j, \boldsymbol{\theta}^{(t-1)}) \mid \boldsymbol{\theta}\big) \right\} \tag{6.28}$$

The relation between this approximation and EM is analogous to that of K-means and EM for Gaussian mixture models (see Sections 2.8.4 and 2.8.5). Using

Equation 6.23,

$$\phi_{\text{best}}(\boldsymbol{w}, \boldsymbol{\theta}) = \arg\max_{\boldsymbol{a}} p(\boldsymbol{a} \mid \boldsymbol{w}, \boldsymbol{\theta}) = \arg\max_{\substack{m_1,\ldots,m_n: \\ \boldsymbol{w}=m_1\ldots m_n}} p(m_1, \ldots, m_n \mid \boldsymbol{\theta}). \tag{6.29}$$

The best segmentation can be solved by a generalization of the Viterbi algorithm for hidden Markov models (Section 2.3.3). Here, the observation is the sequence of $|\boldsymbol{w}|$ letters that form the word $\boldsymbol{w}$, and the hidden states are the morphs of the word. In contrast to the standard Viterbi, one state (morph) can overlap several observations (letters). This adds to the time complexity of the algorithm by a factor of $|\boldsymbol{w}|$. For example, the complexity is $O(|\boldsymbol{w}|^2)$ for a zero-order model and $O(|\boldsymbol{w}|^2 K^2)$ for a first-order model that has K morph categories.

The problem of the Viterbi approach is that the algorithm cannot assign a non-zero probability to any morph that was not stored in the previous lexicon. As the morph lexicon can only be reduced, the initialization has a huge impact on the results.[13] However, the Viterbi search of Equation 6.29 is useful also as a tokenization (decoding) algorithm. That is, it finds the most likely analyses for new words after the model parameters have been set in the actual training phase.

Publications VIII, IX, and X introduce an augmented Viterbi algorithm that can deal with out-of-vocabulary morphs. In the augmented version, the probability of a morph m that is not in the lexicon is set to

$$p_{\text{new}}(m \mid \boldsymbol{\theta}) \approx \frac{p(\tilde{\boldsymbol{\theta}}) p(\boldsymbol{D}_W \mid \tilde{\boldsymbol{\theta}})}{p(\boldsymbol{\theta}) p(\boldsymbol{D}_W \mid \boldsymbol{\theta})}, \tag{6.30}$$

where $p(\tilde{\boldsymbol{\theta}})$ and $p(\boldsymbol{D}_W \mid \tilde{\boldsymbol{\theta}})$ are approximated prior and likelihood probabilities in the case that m is added to the lexicon. For example, if the proper noun **matthew** was never observed in the training data, it would likely to be oversegmented by the standard Viterbi (e.g. **m+at+the+w**). If $p_{\text{new}}(\textbf{matthew} \mid \boldsymbol{\theta})$ was higher than the likelihood of the segmentation, the augmented Viterbi would leave the word intact.

For a maximum-likelihood cost function, the augmented Viterbi is equivalent to smoothing of the probability distribution of morphs. The grammar induction results of Spitkovsky et al. (2010b) indicate that smoothing may help in Viterbi training. If the augmented Viterbi was applied to train Morfessor, the new morphs would be added to the lexicon after determining the new tokenization $\phi_{\text{best}}(\boldsymbol{w}_j, \boldsymbol{\theta}^{(t-1)})$ for all j. However, while being able to introduce new morphs, it would still be a very conservative algorithm, as it does not take into account that adding a new morph to the lexicon is likely to increase likelihoods of many word forms, not just the current $\boldsymbol{w}_j$.

Especially for tokenization, it may sometimes be useful to get the N-best list of parses instead of the single best parse. For example, Turunen and Kurimo (2011) found that using the second best segmentation of Morfessor in addition to the best segmentation improved spoken document retrieval. The N-best decoding algorithms are rather straightforward extensions of the standard Viterbi (see, e.g., Soong and Huang, 1991; Seshadri and Sundberg, 1994).

[13] Interestingly, Deligne and Bimbot (1995) find that the Viterbi training with a simple non-sparse initialization provides as good results as the EM algorithm. More recently, Spitkovsky et al. (2010b, 2011) have compared the Viterbi and EM algorithms for grammar induction and found that both using only Viterbi or alternating between the Viterbi and EM objectives outperforms using only the EM algorithm.

Local optimization algorithms. Because of the problems of the global EM and Viterbi algorithms, Morfessor instead relies on local, greedy search algorithms. That is, at each step, changes that modify only a small part of the parameters are considered, and the change that returns the minimal cost is selected.

Similar to the Viterbi approach, only one potential analysis $\boldsymbol{y}_j \in \Phi(\boldsymbol{w}_j)$ is set to be active at a time. In consequence, a zero probability will be assigned to a large part of the potential morphs. As they do not have to be stored in the lexicon, this type of an algorithm is very memory-efficient.

In the simplest case, the local optimization algorithm considers one word $\boldsymbol{w}_j$ at a time. First, the analysis that minimizes the cost function with the optimal model parameters is selected:

$$\boldsymbol{y}_j^{(t)} = \arg\min_{\boldsymbol{y}_j \in \mathcal{Y}_j} \Big\{ \min_{\boldsymbol{\theta}} L(\boldsymbol{\theta}, \boldsymbol{\Upsilon}^{(t-1)}, \boldsymbol{D}_W) \Big\}. \tag{6.31}$$

Then the parameters are updated:

$$\boldsymbol{\theta}^{(t)} = \arg\min_{\boldsymbol{\theta}} \Big\{ L(\boldsymbol{\theta}, \boldsymbol{\Upsilon}^{(t)}, \boldsymbol{D}_W) \Big\}. \tag{6.32}$$

As neither of the two steps can increase the cost function, the algorithm will converge to a local optimum. Depending on the particular model, the possible choices $\mathcal{Y}_j$ may be restricted to a small number of options. Moreover, the analysis may be optimized for particular substrings shared by multiple words.

6.4.1.4 Morfessor Baseline

Morfessor Baseline (Creutz and Lagus, 2002, 2005b) is the simplest method of the Morfessor family that uses a MAP estimate. It assumes that the morphs of a word occur independently. The cost function is simply

$$L(\boldsymbol{\theta}, \boldsymbol{D}_W, \boldsymbol{\Upsilon}) = -\log p(\boldsymbol{\theta}) - \sum_{j=1}^{|\boldsymbol{D}_W|} \sum_{i=1}^{|\boldsymbol{y}_j|} \log p(m_{ji} \mid \boldsymbol{\theta}). \tag{6.33}$$

The above likelihood is actually deficient, as it does not encode when a sequence of morphs is followed by a word boundary. Omitting the word boundaries does not affect the results in the case that the they are encoded with a fixed number of bits, as also their number is fixed ($|\boldsymbol{D}_W|$). One way to do this is to predict the number of morphs in a word from a uniform prior distribution with a fixed upper limit. However, if the upper limit was a valid parameter of the model, it would still affect the likelihood.

Morfessor Baseline has no grammar parameters, so that part of the prior is omitted and $p(\boldsymbol{\theta}) = p(\mathcal{L})$. The prior for the lexicon size μ has negligible effect and can be omitted.[14] The properties of the morphs in Equation 6.24 contain strings $\boldsymbol{\sigma}_i \in \Sigma^*$ (form properties), and counts $\tau_i \in \{1, \ldots \nu\}$, where $\nu = \sum_i \tau_i$ is the token count of morphs (usage properties). The morph string prior is based on length distribution $p(L)$—Creutz (2003) proposes a Gamma distribution—and categorical distribution $p(C)$ of characters over the character set Σ, both assumed to be known:

$$p(\boldsymbol{\sigma}_i) = p(L = |\boldsymbol{\sigma}_i|) \prod_{j=1}^{|\boldsymbol{\sigma}_i|} p(C = \boldsymbol{\sigma}_{ij}) \tag{6.34}$$

[14] For a complete code, Creutz and Lagus (2007) propose using Rissanen's universal prior for positive integers. Hirsimäki et al. (2006) describe an alternative approach in which μ is determined during the generation of the lexicon.

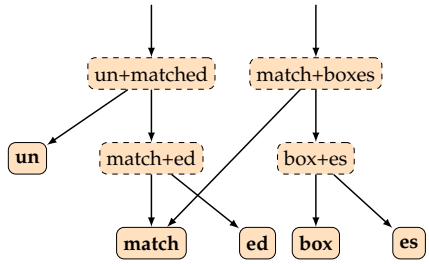


Figure 6.15. Example of an analysis graph for words **unmatched** and **matchboxes** used in Morfessor Baseline.

The character distribution is assumed to be known; in practice, it is estimated from the training data. An implicit exponential length prior is obtained by removing $p(L)$ and using an end-of-word marker as an additional character in $p(C)$ (Creutz and Lagus, 2005b). Given μ and ν, a non-informative prior for the morph counts is

$$p(\tau_1, \ldots, \tau_\mu \mid \mu, \nu) = 1 / \binom{\nu - 1}{\mu - 1}. \tag{6.35}$$

The prior for ν can be omitted for its negligible effect.[15] The counts provide ML estimates for the probabilities of the morphs: $p(M = m_i \mid \boldsymbol{\theta}) = \tau_i / \nu$.

The training algorithm of Morfessor Baseline is described by Creutz and Lagus (2005b). It exploits the assumption that the morphs occur independently. Thus the optimal analysis of a segment is context-independent. The analyses $\mathbf{Y}$ are stored in a binary directed acyclic graph, in which the top nodes are words and the leaf nodes are morphs.[16] Such a graph is illustrated in Figure 6.15. The initial parameters are obtained by adding all the words into the morph lexicon (and thus as root nodes of the graph). In one training epoch, all words are processed once in random order. The local optimization step (Equations 6.31–6.32) modifies the word nodes: for the current node, it considers every possible split into two morphs, as well as no split. If the node is split, the search is applied recursively to its child nodes. The algorithm stops when the overall cost decrease of a training epoch is less than a given threshold.

6.4.1.5 Morfessor Categories-MAP

Categories-MAP (Creutz and Lagus, 2005a, 2007) is the latest publicly available version of Morfessor. It estimates the probability of a morph sequence with an HMM. The states c include four categories for morphs—prefix (`PRE`), stem (`STM`), suffix (`SUF`), and non-morpheme (`NON`)—and a word boundary state (`#`). Given the analyses $\mathbf{Y}$, the data likelihood is

$$p(\boldsymbol{D}_W \mid \boldsymbol{\theta}, \mathbf{Y}) = \prod_{j=1}^{|\boldsymbol{D}_W|} \Big[p(c_{j1} \mid \#) \prod_{i=1}^{|\boldsymbol{y}_j|} \big[p(m_{ji} \mid c_{ji}) p(c_{j(i+1)} \mid c_{ji}) \big] p(\# \mid c_{j|\boldsymbol{y}_j|}) \Big]. \tag{6.36}$$

[15] Most of the Morfessor publications do not mention the prior on the token count ν. Any non-informative prior would do; Hirsimäki et al. (2006) use Rissanen's universal prior.

[16] Note that the graph structure is applied only during training. It is not stored by the model parameters.

A few transition probabilities are restricted to zeros: prefixes cannot end a word ($p(\#\,|\,\texttt{PRE}) = 0$) and suffixes cannot start a word or directly follow a prefix ($p(\texttt{SUF}\,|\,\#) = p(\texttt{SUF}\,|\,\texttt{PRE}) = 0$). The non-zero transition probabilities are maximum-likelihood estimates from the data. The emission probabilities are estimated by

$$p(m_i \,|\, c_k, \boldsymbol{\theta}) = \frac{p(m_i \,|\, \boldsymbol{\theta}) p(c_k \,|\, m_i, \boldsymbol{\theta})}{\sum_{j=1}^{\mu} p(m_j \,|\, \boldsymbol{\theta}) p(c_k \,|\, m_j, \boldsymbol{\theta})}. \tag{6.37}$$

The category-independent probabilities $p(m_i \,|\, \boldsymbol{\theta})$ are provided by the morph counts τ_i as in Morfessor Baseline. The probability of morph belonging to a particular category, $p(c_k \,|\, m_i, \boldsymbol{\theta})$, depends on three properties of the morph: length, *right perplexity*, and *left perplexity*. Right perplexity is calculated by

$$\text{right-perp}(m_i) = \left(\prod_{m_j \in \text{right-of}(m_i)} p(m_j \,|\, m_i) \right)^{-\frac{1}{\tau_i}}, \tag{6.38}$$

where product is over context morphs m_j that immediately follow m_i in the segmented corpus. Left perplexity is calculated analogously.[17] Then a logistic function $f(x) = (1 + e^{-a(x-t)})^{-1}$ is used to obtain measures of prefix-likeness, suffix-likeness, and stem-likeness from right perplexity, left perplexity, and length, respectively. The function implements graded thresholding, where t controls the threshold point and a steepness of the function. Finally, the likeness-measures are used to obtain the probability values for each of the four categories (for details, see Creutz and Lagus, 2007).

Another feature in Categories-MAP, inspired by de Marcken (1996), is that it applies a *hierarchical* lexicon (Creutz and Lagus, 2005a). That is, a morph can either consist of a string of letters (as in Baseline), or of two *submorphs* that are defined elsewhere in the lexicon. The probability of a form is then

$$p(\text{form}(m_i)) = \begin{cases} p(\text{sub}) p(c_{i1} \,|\, \text{sub}) p(m_{i1} \,|\, c_{i1}) p(c_{i2} \,|\, c_{i1}) p(m_{i2} \,|\, c_{i2}) & \text{if } m_i \text{ has substructure} \\ (1 - p(\text{sub})) p(|\boldsymbol{\sigma}_i|) \prod_{j=1}^{|\boldsymbol{\sigma}_i|} p(\sigma_{ij}) & \text{otherwise} \end{cases}, \tag{6.39}$$

where $p(\text{sub})$ is the probability that a morph has substructure, m_{i1} and m_{i2} are the two submorphs of m_i and c_{i1} and c_{i2} their categories, respectively. The probabilities $p(\text{sub})$ and $p(c_{i1} \,|\, \text{sub})$ are estimated from the current lexicon, and transition and emission probabilities are the same as in Equation 6.36.

An advantage of the hierarchical lexicon is that it reduces the cost of storing a new form if a large part of it is already in the lexicon. For example, consider storing **matchboxes** in the lexicon with the correct segmentation if **matchbox** is already stored. The cost function would include two more emission probabilities, $p(\textbf{matchbox})$ and $p(\textbf{es})$. In contrast, Baseline or Categories-ML, that do not use a hierarchical lexicon, would add three more emission probabilities ($p(\textbf{match})$, $p(\textbf{box})$, $p(\textbf{es})$). If **matchbox** was not in the lexicon, a strictly hierarchical model (i.e., a PCFG or some context-sensitive equivalent) would require adding it and thus also one more emission probability than a non-hierarchical

[17] Note that the perplexity measures applied in Morfessor Categories-MAP are similar to the measures used in the LSV method by Harris (1955) and Hafer and Weiss (1974). However, in Morfessor, they affect the segmentation points only indirectly.

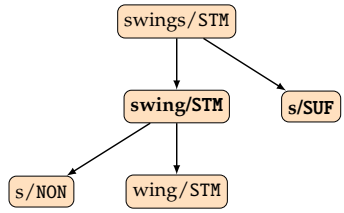


Figure 6.16. Example of a hierarchical segmentation in Morfessor Categories-MAP. Morphs are tagged with their most likely categories in the context. Because the first **s** is tagged as a non-morpheme, the output analysis is **swing/STM + s/SUF**.

model, but Categories-MAP is more likely to use the "surface" HMM of Equation 6.36 to segment **matchbox**.

The output of Categories-MAP is not the fully segmented form, but the substructures are expanded only up to the level in which there is no morphs of the non-morpheme category. For example, the stem **swing** in **swings** may be analyzed as having submorphs **s** and **wing**, but because **s** starts the word and has small length and right perplexity, it is likely to be categorized as a non-morpheme in this context (Figure 6.16).

Because of the various complex dependencies between the properties of the morphs (forms, lengths, left and right perplexities, and counts), probabilities of the HMM, and finally the segmented data, it is not clear how a proper prior should be designed for the Categories-MAP model. Creutz and Lagus (2007) use the following scheme: The prior for the grammar is omitted, as it has a fixed number of parameters (transition probabilities). The lexicon again stores the form and usage properties of the morphs. The frequency and length priors are set similarly to Morfessor Baseline, and priors for the left and right perplexities are taken from Rissanen's universal prior for integers.

The training of the Categories-MAP model includes several phases (Creutz and Lagus, 2005a): First, the segmentation is initialized with the Baseline method. Then the morphs are tagged with the four categories using the emission probabilities. The analyses in **Y** contain both the segmentation and the categories of the segments. Three phases are repeated once: (1) local splitting of morphs into submorphs, (2) local joining of two adjacent morphs either by direct concatenation or by adding a higher level morph that has the focus morphs as submorphs, and (3) the Viterbi algorithm of Equation 6.28. Finally, the substructures are expanded to the finest level without non-morphemes.

6.4.2 Learning of allomorphy

Most of the work on unsupervised learning of morphology concentrates either on morphological segmentation or clustering of related word forms (i.e., lemmatization). There is much less work on how to train an actual morphological analyzer in an unsupervised manner. From the viewpoint of the IA approach to morphology, the analyzer has to model the phenomena of allomorphy and syncretism that separate the abstract morphemes from their surface forms, morphs.

The possible sources of information for this task include the context distribution of the morpheme (especially the restriction on complementary distributions of allomorphs), the context distribution of the word (Schone and Jurafsky, 2001; Baroni et al., 2002), the frequency of the word (Yarowsky and Wicentowski, 2000), and—relevant only for allomorphy—phonetic and orthographic similar-

ity.

There are at least two approaches for solving the problem: (1) extending a segmentation based method by clustering related morphs and disambiguating others, or (2) extending a clustering based method by identifying inflectional and derivational affixes that are shared among the clusters. The latter has been studied by Bernhard (2010b,a). Bernhard's MorphoNet algorithm first finds transformation rules, encoded as regular expressions, between orthographically similar words. Then it constructs a graph where the nodes are words and the edges are the extracted rules. Groups of related word forms are found by a community detection method. Finally, the shortest word within each group is selected as a lemma, and fixed parts of the transformation rules are mapped to strings representing affixes.

Extending a method based on segmentation has been a more popular approach. Already Morfessor Categories-MAP can deal with some cases of syncretism (i.e., if the two morphemes with the same surface form have different categories). The same applies also to other algorithms that identify suffixes from stems (e.g. Bernhard, 2008). However, excluding lemmatization (Schone and Jurafsky, 2001; Baroni et al., 2002) and partially supervised learning (Yarowsky and Wicentowski, 2000; Shalonova et al., 2009), there is very little work on modeling allomorphic variations of a single morpheme. Two exceptions are Dasgupta and Ng (2007) and Demberg (2007), who independently extend the segmentation algorithm by Keshava and Pitler (2006). In contrast to the original algorithm, both extensions allow segmentation to many morphemes per word. In addition, the former extension detects incorrect analyses using word frequency information, and learns stem allomorphs using context-sensitive orthographic rules that may replace, insert, or delete a single letter in a stem. The latter extension removes the restriction that stems should be valid words in the training lexicon, applies a probabilistic segmentation by a bigram language model (bootstrapped with the original segmentation algorithm), and finally identify allomorphs produced by regular processes such as ablauting and alternations in morpheme boundaries. However, the modeled alterations are quite specific to German, and not even tested on other languages.

Most of the previous attempts to learn allomorphy have been limited in their treatment of agglutinative languages: For example, Yarowsky and Wicentowski (2000) consider only stem-suffix pairs. Bernhard (2010b) allows multiple suffixes, but only one stem per word. The approach of Dasgupta and Ng (2007) is more general and allows multiple stems, but cannot find, for example, affixes between stems. The probabilistic segmentation by Demberg (2007) does not have any limitations in this regard.

Publication VIII presents a method called *Allomorfessor Baseline* that attempts to deal with stem allomorphy within a flexible agglutinative morphology. As indicated by the name, it belongs to the Morfessor family. In particular, it extends the Morfessor Baseline method by including a sub-model for allomorphic variations.

6.4.2.1 A simple model of stem allomorphy

The orthographic changes in the allomorphs of the same stem are often small. Typically one or two letters are deleted, inserted, or substituted. However, edit distance is too general a measure for selecting candidates for allomorphs. Especially for short words, there may be a larger number of valid word forms already

Table 6.6. The edit operations of the transformations used in Allomorfessor and some English and Finnish examples.

Operation	Notation*	Description
substitution	*k*`x|y`	Change k^{th} `x` to `y`
deletion	`-`*k*`x`	Remove k^{th} `x`

*k is omitted if one

Source	Transformation	Target
wife	`(f|v)`	**wive** (e.g. **wive+s**)
try	`(y|i)`	**tri** (e.g. **tri+es**)
invite	`(-e)`	**invit** (e.g. **invit+ed**)
sing	`(i|a)`	**sang**
bring	`(-g -n i|o)`	**bro** (**bro+ught**)
fight	`(-t -h -g i|o)`	**fo** (**fo+ught**)
kenkä (shoe)	`(k|g)`	**kengä** (e.g. **kengä+ssä**, in shoe)
tanko (pole)	`(k|g)`	**tango** (e.g. **tango+t**, poles)
ranta (shore)	`(-a t|n)`	**rann** (e.g. **rann+oi+lla**, on shores)
ranta	`(a|o t|n)`	**ranno** (e.g. **ranno+i+lla**)
ihminen (human)	`(2n|s)`	**ihmisen** (human's)
ihminen	`(-n n|s)`	**ihmise** (e.g. **ihmise+n**)

within an edit distance of one. For example, **silk** has edit distance one to (at least) **ilk**, **milk**, **sulk**, **sink**, **sick**, **silo**, and **silt**. Fortunately, the modifications are typically very regular, so that there is possibility to capture many allomorphs with the same modification.

Publication VIII attempts to model the orthographic changes with transformations that consist of short sequences of edit operations.[18] Transformations make minor modifications to the surface forms of the morphemes. The main problem in this setting is to find a suitable balance for expressiveness of the transformations. The transformations should be general enough so that similar variations in different stems can be modeled with the same transformation, but they should also be restricted so that the number of spurious analyses becomes as small as possible.

The transformations applied in Publication VIII are presented in Table 6.6. They consist of a sequence of deletion and substitution operations. Insertions are not allowed, because they would make it possible to model entire suffixes via transformations. The operations are applied in order and the current position in the string is stored. Each operation modifies the k^{th} target letter in the left side of the current position. A dynamic programming algorithm similar to those in normal edit distances can be applied to find the shortest transformation between two arbitrary strings.

As indicated by the examples in Table 6.6, transformations work well enough for allomorphic variations caused by many regular inflections, but, for example, irregular English verbs cause problems. The ablaut in **sing**-**sang** can be modeled with a single operation. There will not be any morpheme that would correspond to the past tense, but at least the two stems could be combined. In contrast, **bring**-**brought** cannot be modeled in the same manner, as insertions are not allowed. It is possible to find a suffix that fits multiple verbs, such as **+ught**, but then the transformations are very long and distinct.

To ensure that suffixes would still be correctly segmented and not modified by

[18] In Publication VIII, the transformations were called mutations. Here we use the more standard term.

Table 6.7. The proportion of morphs with allomorphic variation and how many variants can be modeled with the transformations described in Publication VIII for English, Finnish, and Turkish.

	English		Finnish		Turkish	
Morph types	21173		68743		23376	
• Is allomorph?	10858	(51%)	56653	(82%)	646	(2.8%)
• Transformation possible?	9812	(82%)	36210	(64%)	102	(16%)
Morph tokens ($\times 10^3$)	76968		73512		23289	
• Is allomorph?	42283	(55%)	61583	(84%)	18751	(51%)
• Transformation possible?	14707	(35%)	11978	(30%)	226	(1.9%)

the transformations, the use of transformations was ruled out for morphemes shorter than four characters and last morphemes of the words regardless of their length. Publication VIII includes an estimate of how many allomorphs in linguistic gold standards could be described by the applied transformations. Table 6.7 shows the estimates for morph types and morph tokens in the training data. For example, the English gold standard has 21 thousand morphs, of which about 11 thousand have other allomorphic variants. Of these, 9800 (85%) could be transformed to the canonical form of the morph, lemma.[19] The training corpus had 77 million morph tokens, 55% of them had other allomorphs, and 35% of the allomorphs could be transformed to the lemma. Allomorphy is even more prevalent in Finnish. In Turkish, the number of morph types that have other allomorphs is much lower, but they are still frequent in the corpus. Most of them are likely to be suffixes and thus outside the scope of the transformations used here.

6.4.2.2 Allomorfessor Baseline

In Allomorfessor, an analysis $\boldsymbol{a}$ of a word is no longer a simple list of morphs. Instead, it is a list of pairs of morphemes m_i and transformations $\mathfrak{d}_i$:

$$\boldsymbol{a} = \big((\mathfrak{d}_1, m_1), \ldots, (\mathfrak{d}_n, m_n)\big). \tag{6.40}$$

Each transformation $\mathfrak{d}_i$ is applied to the string representation of the previous morpheme to obtain the corresponding surface form $\boldsymbol{s}_{i-1}$: $\mathfrak{d}_i(\sigma_{i-1}) = \boldsymbol{s}_{i-1}$. Any of the transformations can be empty, so that $\mathfrak{d}_\epsilon(\sigma) = \sigma$. The first transformation $\mathfrak{d}_1$ as well as the (non-existing) morpheme before it are always empty. Thus the detokenization function is

$$\phi^{-1}(\boldsymbol{a}, \boldsymbol{\theta}) = \mathfrak{d}_1(\epsilon)\mathfrak{d}_2(\sigma_1)\mathfrak{d}_3(\sigma_2)\ldots\mathfrak{d}_n(\sigma_{n-1})\sigma_n. \tag{6.41}$$

For example, if $\boldsymbol{w} = \textbf{wives}$, a possible analysis is $\boldsymbol{a} = \big((\epsilon, \textbf{wife}), (\mathfrak{d}_{\texttt{(f|v)}}, \textbf{s})\big)$ and $\phi^{-1}(\boldsymbol{a}, \boldsymbol{\theta}) = \mathfrak{d}_\epsilon(\epsilon)\mathfrak{d}_{\texttt{(f|v)}}(\textbf{wife})\textbf{s} = \textbf{wives}$.

The above notation, which groups the transformations and morphemes together into pairs $(\mathfrak{d}_i, m_i)$, indicates the independence assumptions of the model. Theoretically, the probability of $\mathfrak{d}_i$ should depend on the morpheme m_{i-1} that it is applied to: if they are independent, the model can generate invalid combinations of morphemes and transformations (e.g., $\mathfrak{d}_{\texttt{(-e)}}$ for a morpheme that does not have letter **e**). However, the probability of $\mathfrak{d}_i$ should also depend on the

[19] The most frequent allomorph that occurred alone as a word was selected as the lemma. If none did, the morph was considered as a suffix or prefix and skipped.

following suffix m_i, as adding the suffix is exactly what triggers the transformation. For example, $\mathfrak{d}_{(\mathtt{f|v})}$ should be applied to **wife** *only* if the next morpheme is the plural suffix. Moreover, $p(\mathfrak{d}_i, m_i)$ is often easier to estimate than $p(\mathfrak{d}_i, m_{i-1})$ because of the higher frequency of suffixes.

Making both of the pairs $(\mathfrak{d}_i, m_i)$ and $(\mathfrak{d}_i, m_{i-1})$ dependent would give a first-order Markov model even if morphemes were assumed to be independent given $\mathfrak{d}_i$. Because of the increased computational burden of finding the transformations, Allomorfessor Baseline keeps the independence assumption of Morfessor Baseline. That is, each pair $(\mathfrak{d}_i, m_i)$ is assumed to be independent, but $\mathfrak{d}_i$ depends on m_i. The conditional likelihood is then

$$p(\boldsymbol{D}_W \mid \mathbf{Y}, \boldsymbol{\theta}) = \prod_{j=1}^{|\boldsymbol{D}_W|} \prod_{i=1}^{|\boldsymbol{y}_j|} p(m_{ji}) p(\mathfrak{d}_{ji} \mid m_{ji}). \tag{6.42}$$

While allowed by the probabilities, a transformation incompatible with the previous morpheme is never selected during the training or when new analyses or words are generated.

6.4.2.3 Model prior

The prior of Allomorfessor Baseline is otherwise similar to Morfessor Baseline, but the set of transformations is added to the grammar $\mathcal{G}$. Similarly to the morpheme lexicons,

$$p(\mathcal{G} \mid \mathcal{L}) = p(\xi \mid \mathcal{L}) \times p(\text{properties}(\mathfrak{d}_1) \ldots \text{properties}(\mathfrak{d}_\xi) \mid \mathcal{L}) \times \xi!, \tag{6.43}$$

where ξ is the number of distinct transformations, including the empty transformation $\mathfrak{d}_1 = \mathfrak{d}_\epsilon$. Usage properties of the transformations include the frequencies of the transformations (v_i) and their co-occurrences with the suffix morphs (ω_{ij}), and form properties of the transformations encode their edit operations.

Assuming that each morpheme has at least one co-occurrence with the empty transformation, the count v_1 of the empty transformation is at most ν (morpheme tokens) and at least μ (morpheme types). Sampling from a uniform distribution gives

$$p(v_1 \mid \mu, \nu) = \frac{1}{\nu - \mu + 1}. \tag{6.44}$$

There are as many transformation tokens as there are morph tokens in the data, so the other $\xi - 1$ transformations have $\nu - v_1$ occurrences in total. A non-informative prior similar to Equation 6.35 is then

$$p(v_2, \ldots, v_\xi \mid \mu, \nu, v_1) = 1 / \binom{\nu - v_1 - 1}{\mu - 2}. \tag{6.45}$$

Next, there are priors for the co-occurrence counts ω_{ij} of transformations $\mathfrak{d}_i$ and morphemes m_j. The co-occurrences form a $\xi \times \mu$ sparse matrix $\boldsymbol{\Omega}$. As $\sum_{i=1}^{\xi} \omega_{ij} = \tau_j$, co-occurrence counts can be determined for one of the transformations given all the rest. A good choice for this is the empty transformation, because it is likely to have the largest number of occurrences. For each non-empty transformation $\mathfrak{d}_i$, the v_i occurrences has to be divided into μ possible morphemes. A non-informative prior is

$$p(\boldsymbol{\Omega} \mid \mu, \nu, v_1 \ldots v_\xi) = \prod_{i=2}^{\xi} \left[\binom{v_i + \mu - 1}{\mu - 1} \right]^{-1}. \tag{6.46}$$

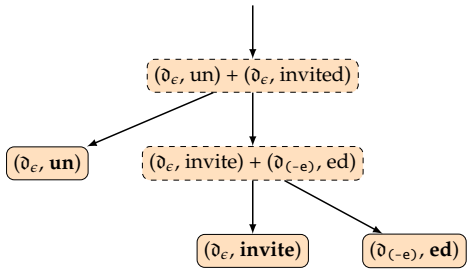


Figure 6.17. Example of an analysis graph for **uninvited** for Allomorfessor Baseline.

Finally, there is a prior for the forms of the transformations. The forms are assumed to be independent. Let $\boldsymbol{o}_i = (o_{i1}, \ldots o_{in})$ be the operations in ϑ_i. Each operation o_{ij} includes the position indicator k, type of the operation, and 1–2 letters from the alphabet Σ (depending on the operation type). The prior is set to:

$$p(\boldsymbol{o}_i) = p(|\boldsymbol{o}_i|) \prod_{j=1}^{|\boldsymbol{o}_i|} p(o_{ij}) \tag{6.47}$$

$$p(o_{ij}) = \begin{cases} p(k_{ij})p(\text{del})\frac{1}{|\Sigma|} & \text{if } o_{ij} \text{ is a deletion} \\ p(k_{ij})(1-p(\text{del}))\frac{1}{|\Sigma|^2} & \text{if } o_{ij} \text{ is a substitution.} \end{cases} \tag{6.48}$$

The probabilities $p(|\boldsymbol{o}_i|)$ and $p(k_{ij})$ can be taken from any suitable distribution that prefers small values; Gamma distributions with scale and shape parameters equal to one are used in Publication VIII. The probabilities of the two operations are set to be equal ($p(\text{del}) = 0.5$).

6.4.2.4 Training algorithm

Also the training algorithm of Allomorfessor is a direct extension of the Morfessor Baseline. A similar analysis graph, illustrated in Figure 6.17, is applied during the training. The only difference is that if a segment $\boldsymbol{w}$ is split, both parts include a transformation. The transformation of the left segment is always empty, while the transformation of the right segment may be non-empty. The first (left-most) transformation of a child node is inherited from its parent node.

Partial pseudocode for the training algorithm is shown in Figure 6.18. OPTIMIZESINGLE implements the standard local training of Morfessor Baseline, with the exception that analyses with non-empty transformations are included if the length of the current strings is at least four (line 4). In one training epoch, it is run once for each word form in the training data. As the potential number of analyses may be very large, the algorithm actually tests at most $K = 50$ options (line 6). The K first options always include no split and splits with empty transformations (if $n \leq K$). GETALLOMORPHANALYSES shows some more heuristic restrictions for the search. The suffix $\boldsymbol{s}$ can be at most five letters long, and it has to already exist in the lexicon (line 8). At line 9, candidate lemmas are selected from the training data. The first $n - d$ letters of the surface form $\boldsymbol{w}$ and candidate lemma $\boldsymbol{v}$ has to be equal; for short words, $d = 3$, and for longer words, $d = 4$. Moreover, the candidate lemma can be at most two letters longer than the surface form. GETSHORTESTTRANSFORMATION at line 10 returns the shortest list of operations that transform the first argument to the second argument.

```
OPTIMIZESINGLE(w_j)
 1  n ← |w_j|
 2  A ← [(w_j, ∅_ε, ε)] + [(w_{j(1...i)}, ∅_ε, w_{j((i+1)...n)}) : i ∈ (1, ..., n − 1)]
 3  if n ≥ 4
 4     A ← A + GETALLOMORPHANALYSES(w_j)
 5  bestcost ← ∞
 6  for k ∈ (1, ..., min(K, |A|)) do
 7     y_j ← a_k
 8     θ ← arg min_{θ*} L(θ*, Y, D_W)
 9     cost ← L(θ, Y, D_W)
10     if cost < bestcost
11        bestcost ← cost
12        k* ← k
13  y_j ← a_{k*}
14  θ ← arg min_{θ*} L(θ*, Y, D_W)
15  if y_j involves a split
16     OPTIMIZESINGLE( left morpheme of y_j)
17     OPTIMIZESINGLE( right morpheme of y_j)

GETALLOMORPHANALYSES(w_j)
 1  n ← |w_j|
 2  d ← 3
 3  if n ≥ 6
 4     d ← 4
 5  A ← []
 6  for i ∈ (1, ..., n − 1) do
 7     s ← w_{j((i+1)...n)}
 8     if |s| ≤ 5 ∧ s ∈ L
 9        for v_j ∈ {v ∈ D_W : i ≤ |v| ≤ i + 2 ∧ v_{1...(n−d)} = w_{j(1...(n−d))}} do
10           A ← A + [(v_j, GETSHORTESTTRANSFORMATION(v_j, w_{j(1...i)}), s)]
11  Sort A by ascending suffix length |s| and descending lemma length |v_j|
12  return A
```

Figure 6.18. Local search algorithm for Allomorfessor. $\boldsymbol{A}$ is a list of possible analyses and $+$ indicates list concatenation. Each analysis is stored as a triple consisting of a prefix morpheme, a transformation, and a suffix morpheme. The second subscripts of $\boldsymbol{w}_j$ indicate substrings. Empty morphemes and transformations are denoted by ϵ and $\emptyset_\epsilon$, respectively. $\boldsymbol{D}_W$ is the training data, $\boldsymbol{\theta} = (\mathcal{L}, \mathcal{G})$ model parameters, and $\mathbf{Y}$ contains the current analyses of the word segments.

Let the number of word forms in the training data be W. For each $\boldsymbol{w} \in \boldsymbol{D}_W$, there can be at most $2|\boldsymbol{w}|$ recursions of OPTIMIZESINGLE, but assuming the word lengths are bounded, that can be considered a constant. Assuming that the words are sorted (in $O(W \log W)$ time), finding candidate lemmas that match the first letters of $\boldsymbol{w}$ can be done with range search (time complexity $O(\log W)$). The complexity of finding the shortest transformation is again limited by the maximum word length. Thus, as the number of analyses is limited to K, one epoch of the algorithm has a time complexity of $O(KW \log W)$.

Also the Viterbi algorithm is slightly more complicated for Allomorfessor. The states are morphemes, and the observed sequence of $|\boldsymbol{w}|$ letters are emitted via the transformations of the morphemes. The worst case time complexity is $O(\mu\xi|\boldsymbol{w}|^2)$ for μ morphemes and ξ transformations. In practice, however, the numbers of morphemes and transformations that have to be considered in a certain step are much more limited.

6.4.2.5 Experiments

In Publication VIII, Allomorfessor Baseline was evaluated in the Morpho Challenge 2009 competition (Kurimo et al., 2010c). Morpho Challenge 2009 included three types of evaluations: (1) direct comparison to a linguistic gold standard analysis for Arabic, English, Finnish, German, and Turkish, (2) indirect evaluation in information retrieval tasks for English, Finnish, and German, and (3) indirect evaluation in a machine translation task for Finnish and German. Compared with other algorithms evaluated in the Challenge, Allomorfessor performed fairly well, winning the linguistic evaluation for English and both machine translation evaluations. However, it is more revealing to compare it with other versions of Morfessor, especially to Morfessor Baseline.

A further complication is that Allomorfessor was trained on data sets for which all word forms that occurred only once were discarded. While this was originally done just to speed up the training, it was quickly noticed that removing the singletons improved the results of the linguistic evaluations considerably. (The reasons for this are discussed in Section 6.4.3 and Publication IX.) In order to make a fair comparison, also Morfessor Baseline was trained on the same data set. After training, the augmented Viterbi algorithm (Section 6.4.1.3) was applied to obtain analyses for all words in the data sets.

The results of Morfessor and Allomorfessor for English, Finnish, German, and Turkish are compared in Table 6.8.[20] The linguistic evaluation of Morpho Challenges (called the MC evaluation) estimates precision and recall of the result by sampling pairs of words that share the same morphemes (see Section 6.3.2.3). In theory, finding the allomorphs should improve recall of the evaluation. However, Table 6.8 shows improved recall only for English. In contrast, recall is decreased for Finnish, German, and Turkish, and precision is increased for all languages. In the IR evaluation, there is no statistically significant difference between the mean average precisions of Morfessor and Allomorfessor.

Table 6.8. Comparison of Morfessor Baseline (Morf.) and Allomorfessor Baseline (Allom.) in Morpho Challenge 2009 evaluations. Precision (Pre), recall (Rec), and F-measure (F) are for linguistic MC evaluation and mean average precision (MAP) for IR evaluation. The superscripts $+$ and $-$ indicate statistically significant increase or decrease, respectively. The differences of the IR results are not statistically significant. The last three rows give the number of morpheme tokens, morpheme types, and the average number of morphemes in an analysis $\boldsymbol{a}$.

	English Morf. / Allom.	Finnish Morf. / Allom.	German Morf. / Allom.	Turkish Morf. / Allom.
Pre (%)	68.43 / 68.98^{+}	86.07 / 86.51^{+}	76.47 / 77.78^{+}	85.43 / 85.89^{+}
Rec (%)	56.19 / 56.82^{+}	20.33 / 19.96^{-}	30.49 / 28.83^{-}	20.03 / 19.53^{-}
F (%)	61.71 / 62.31^{+}	32.88 / 32.44^{-}	43.60 / 42.07^{-}	32.45 / 31.82^{-}
MAP (%)	38.73 / 38.52	44.75 / 46.01	47.28 / 43.88	-
Types	23673 / 23741	69638 / 70228	43324 / 43609	29178 / 29193
Tokens ($\times 10^6$)	1.001 / 0.997	5.519 / 5.423	3.386 / 3.327	1.488 / 1.460
Mean $\lvert\boldsymbol{a}\rvert$	2.60 / 2.59	2.50 / 2.46	2.67 / 2.63	2.41 / 2.37

The slight increase in precision and decrease of recall in most languages is explained by the fact that Allomorfessor uses somewhat larger lexicons and less morphemes per word than Morfessor does (see last rows in Table 6.8). Thus Al-

[20] The Arabic results are omitted here, because the evaluation method proved to be problematic for the Arabic reference analysis: none of the algorithms submitted to the Challenge could outperform splitting to individual letters (Kurimo et al., 2010c).

lomorfessor seems to have a different optimum in the trade-off between prior and likelihood with respect to these numbers. On one hand, both increasing the morpheme type count μ and the token count ν increase the cost $-\log p(\boldsymbol{\theta})$ more in Allomorfessor than in Morfessor (given that non-empty transformation are included), and it is not evident which has a larger effect. On the other hand, at least the data cost $-\log p(\boldsymbol{D}_w \mid \boldsymbol{\theta})$ will often be larger in Allomorfessor because of the new probabilities $p(\mathfrak{d}_i|m_i)$. Adding the dependency between m_i and $\mathfrak{d}_{i+1}$ should reduce the difference, but at the expense of making the likelihood calculations less trivial.

The number of the transformations used by the model are shown in Table 6.9. For English, Finnish, and Turkish, they can be compared with the approximate upper bound from the gold standard analyses. For English and Finnish, the actual usage is only one hundredth of what might be possible. For Turkish, the usage is closer to what would be achievable in theory, but also the upper bound is much lower.

Table 6.9. The number of non-empty transformations in Viterbi analysis of the full training data. Transformation usage is the number of non-empty transformation tokens divided by the number of morpheme tokens. Gold standard usage is the equivalent proportion from the linguistic gold standard when transformations are applied whenever possible. For Arabic, "v" indicates vowelized script.

Language	Arabic	Arabic (v)	English	Finnish	German	Turkish
Transformation types	0	69	15	66	26	55
Transformation usage	0.0%	4.61%	0.18%	0.44%	0.17%	0.12%
Gold standard usage	n/a	n/a	21.15%	31.06%	n/a	0.86%

Interestingly, for Arabic, Allomorfessor applied no transformations for the standard non-vowelized script, but quite many for the vowelized script. It seems that the model started encoding some of the vowel patters of Arabic using the transformations. In any case, this demonstrates how the method can select the encoding that gives the most compact representations without any supervision.

The quality of the analyses with transformations was studied manually. Tables A.11 and A.12 (pages 232–233) in Appendices show the most frequently applied transformations for English and Finnish. Many of the frequent transformations make sense from a linguistic viewpoint. Especially for Finnish words, the selected lemma of the stem is often something else than the linguistically motivated lemma, but this is only a minor problem: the labels are arbitrary in any case. For English, transformations are frequently used to fix misspelled word forms or modify proper names. While this outcome was not intended, it is still likely to be useful in practice.

Overall, the main reason that Allomorfessor outperformed Morfessor Baseline only for English seems to be that the undersegmentation of many word forms made it unnecessary to use transformations: if a stem and the following suffix are stored as a whole by the model, there is no need to transform the stem. Comparing the balance between precision and recall in Table 6.8, the precision and recall are indeed closest to each other for the English data, indicating that the words are least undersegmented for English.

6.4.3 The effect of corpus size and word frequencies

As defined at the beginning of Section 6.4, probabilistic generative models of morphology define a joint distribution for the words and their analyses. Regardless of whether model selection and parameter estimation are based on maximum likelihood, maximum a posteriori, the MDL principle, or a full Bayesian approach, one has to define the likelihood of the training data. Then a fundamental question is what the training data should be: a set of words in the training corpus (i.e., observed word types), the corpus itself (observed word tokens), the set of possible words in the language (as suggested by Hammarström, 2006), or something else?

In practice, of course, there is no way to describe all possible words in a language, and training has to be based on a corpus of a limited size and quality. Then the questions include how to deal with the word frequency information, what is the effect of the corpus size, and finally, what is the effect of those words in the corpus that are *not* correct words of the language. Especially for methods that use an MDL-style prior and look for the optimal balance between the prior and the likelihood, the size and type of the training data has a drastic effect on the results.

6.4.3.1 Types and tokens

Creutz and Lagus (2004, 2005b, 2007) have observed that for Morfessor Baseline, training on word types provided large improvement in performance over word tokens, when evaluating against a linguistic gold standard segmentation. A similar effect has been observed at least for the Bayesian two-stage models of Goldwater et al. (2006, 2011) and for the log-linear model by Poon et al. (2009).

The two-stage models (see Section 2.7.4) provide further insight into the question. Using a Pitman-Yor process adaptor, the discount parameter d actually controls the effect of word frequencies: $d = 0$ uses a type-based distribution, whereas $d = 1$ uses a token-based distribution. Using a simple stem-and-suffix model, Goldwater et al. (2006, 2011) get the best experimental results for morphological segmentation when $0 \leq d \leq 0.7$. If the parameter is increased to emphasize frequent words more than that, the model starts to undersegment.

Another approach that provides something between between word types and tokens is to collect the word frequencies over a set of unique fragments of language longer than a word, but shorter than the full corpus. For example, training on unique phrases means that the common words will have more weight, but not as much as their direct corpus frequency would indicate. In particular, this kind of approach has been applied by Snyder and Barzilay (2008b), who train a bilingual morphological model on aligned phrases that consist up to 4–6 words. However, they did not study the effect of the frequencies.

6.4.3.2 Size of the training data

Creutz and Lagus (2005b, 2007) have tested the various Morfessor models with different sizes of the training data. Especially for the Baseline models, increase of the data drastically improves precision but degrades recall. If the precision was better to start with, balanced F-measure will degrade. Thus, in contrast to intuition, increasing the amount of training data will degrade the results.

In Morfessor Categories-MAP, the hierarchical lexicon alleviates the problem

partially, but does not eliminate it. For example, Figure 8 by Creutz and Lagus (2007) shows that the recall of Categories-MAP clearly decreased for English when the training data is increased from 250 000 words to 1.2 million words. Of course, a substantial increase in corpus size may also add a large number of foreign words and misspelled words, and thus impede the task. Creutz and Lagus (2007) note an increase of the patterns that do not belong to contemporary English morphology with the 1.2 million word data. However, if the problem with the very large data sets is the number of non-standard word forms, it should certainly be better to prune all low-frequency words from training than to restrict the amount of data.

6.4.3.3 Experiments with weighted log-likelihood

Publication IX presents new experimental results for the effect of word frequencies in probabilistic generative models of morphology. It considers the following questions: If the goal is to get a segmentation as similar to the linguistic gold standard as possible, (1) should the model be trained on word types, word tokens, or something in between? (2) is it useful to exclude very low-frequency words from the training data?

In order to study these questions, a simple, deterministic function is applied to the word counts. First, note that if the training corpus has $|\boldsymbol{D}_W|$ word types w_j with their respective counts c_j, the logarithm of the corpus likelihood is

$$\log p(\boldsymbol{D}_W \mid \boldsymbol{\Upsilon}, \boldsymbol{\theta}) = \prod_{j=1}^{|\boldsymbol{D}_W|} c_j \log p(\boldsymbol{y}_j \mid \boldsymbol{\theta}). \tag{6.49}$$

This can be changed to a type-based likelihood simply by setting $c_j = 1$ for all j. More generally, a *weighted log-likelihood*

$$\log p(\boldsymbol{D}_W \mid \boldsymbol{\Upsilon}, \boldsymbol{\theta}) = \sum_{j=1}^{|\boldsymbol{D}_W|} f(c_j) \log p(\boldsymbol{y}_j \mid \boldsymbol{\theta}) \tag{6.50}$$

makes it easy to test various settings with different functions $f : \mathbb{Z}_+ \mapsto \mathbb{R}_+ \cup \{0\}$. This can be done as preprocessing for most existing algorithms. In Publication IX, the following family of functions is considered:

$$f(x) = \begin{cases} 0 & \text{if } x < T \\ \alpha g(x) & \text{otherwise} \end{cases} \tag{6.51}$$

Standard type-based training is given by $\alpha = 1$ and constant function $g(x) = 1$, and token-based learning by $\alpha = 1$ and linear function $g(x) = x$. An intermediate approach is to use a logarithmic function $g(x) = \ln(1 + x)$. Note that this corresponds to the logarithmic term frequency component in vector space models (Section 5.1.1). The frequency threshold T can be used to exclude rare word forms. The parameter α can be taken outside the sum and considered as a global weight for the likelihood. In Section 4.5, a similar weight was used to modify the balance between the size and the accuracy of the proposed context-cluster language model. Here, it modifies the amount of segmentation: When $\alpha \to \infty$, it reduces the effect of the prior until the cost function is based only on the likelihood, and the highest likelihood is obtained for a word lexicon. When $\alpha \to 0$, only the prior affects the cost function, and the highest prior is obtained for a character lexicon.

The free parameters of $f(\cdot)$ can be optimized for a given development data set and a target measure. As this discriminative learning may result in models that overfit the development data, the final results have to be calculated for a separate test set.

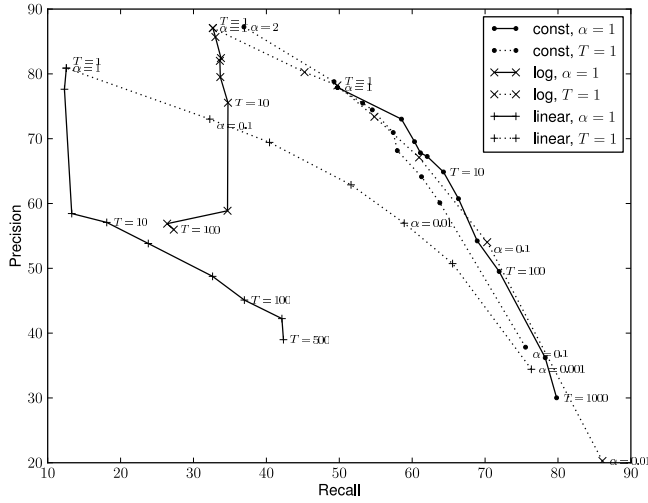
In Publication IX, the effect of the word frequency weighting was tested for two languages, English and Finnish. The training sets, evaluation metric, and test sets were taken from Morpho Challenge 2009 (Kurimo et al., 2010c). The development data sets consisted of 2,000 word forms in English and 10,000 word forms in Finnish, selected randomly from the unannotated training set. To avoid extensive amounts of computation time, the parameters T and α were varied within manually selected ranges. Still, dozens of models had to be trained for each function type. Thus the quickest available algorithm, Morfessor Baseline, was applied in the experiments.

Prior to finding the optimal T and α for each function type, the effect of keeping another at one and varying the other was studied. The results on the development set scores are shown in Figure 6.19. Regardless of the function type and language, precision was much higher than recall for the default values $T = 1$ and $\alpha = 1$, and either increasing T or decreasing α balances the situation. However, increasing the threshold T was a poor option both with linear counts and logarithmic counts: it degraded precision without improving recall. For English, the linear function seemed inferior to constant and logarithmic functions also with a decreased α.

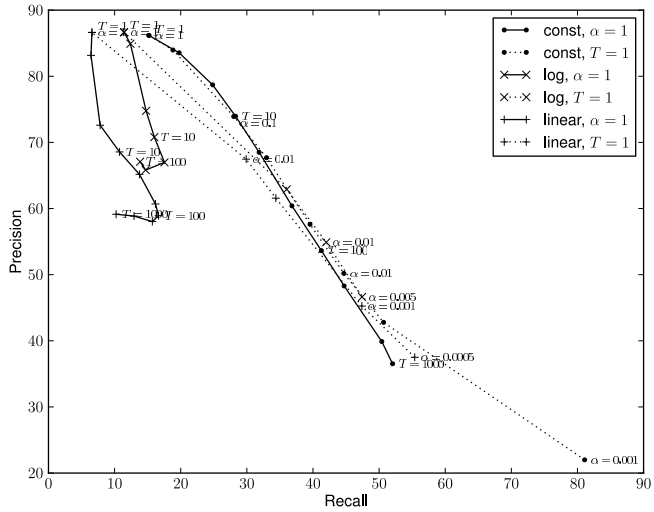
The final test set results for optimized α and T are shown in Table 6.10. Setting a frequency threshold was useful only for the English data with types or logarithmic counts. For Finnish, precision and recall were balanced by only by decreasing the weight parameter. As some of the differences between precision and recall are still quite high, the optimizations succeeded only approximately. For English, the logarithmic function yielded the highest F-measure, but the difference to the constant function was not statistically significant. For Finnish, all optimized F-measures were very close to each other, but the differences were statistically significant, and constant function provided the best result.

Table 6.10. The precision, recall and F-measure of Morfessor Baseline on the final test set with different weighted log-likelihoods. In optimized cases, T and α were selected according to the best F-measure for the development set. In the default case, both are one.

		Constant $f(\cdot)$		Log. $f(\cdot)$		Linear $f(\cdot)$	
		default	optimized	default	optimized	default	optimized
English	T	1	10	1	20	1	1
	α	1	1.1	1	0.2	1	0.01
	Pre	76.13	62.04	87.76	57.85	84.93	53.96
	Rec	48.97	62.27	31.77	67.62	12.00	56.42
	F	59.60	62.16*	46.65	62.35*	21.03	55.16
Finnish	T	1	1	1	1	1	1
	α	1	0.01	1	0.01	1	0.001
	Pre	89.50	53.77	91.24	57.87	91.82	48.86
	Rec	15.70	45.16	11.95	42.06	6.75	47.37
	F	26.72	49.09*	21.13	48.72	12.57	48.10



(a) English



(b) Finnish

Figure 6.19. The precision and recall scores of Morfessor Baseline on the development set with constant (const), logarithmic (log), and linear frequency function types and varying function parameter α or T (Publication IX).

6.4.3.4 Discussion

Overall, the experiments of Publication IX confirmed that if the goal is to get results close to the linguistic gold standard, using word types as training data is usually the best choice. However, because using the logarithmic frequency yielded as good or almost as good results, it should not be ruled out, either. Moreover, the effect has not yet been studied for indirect evaluations. It may well be useful to leave many frequent words undersegmented in, for example, the machine translation task. Later, in Section 6.3.3, it is shown that both Morfessor Baseline and Categories-MAP yield better results in a psycholinguistic prediction task when trained with logarithmically weighted counts.

Whether the words should be pruned using a frequency threshold seems to

depend, unsurprisingly, on the training data. Especially if the frequencies are also used to weight the likelihood, thresholding should be tested with caution. In any case, it is likely to be a better option than limiting the amount of training data.

The effect of the training data can also be considered from the theoretical point of view. Let us consider the case that the model is trained on a corpus. Goldwater (2006, Sec. 4.3) argues that for any segmentation model in which (a) there exists a solution with no boundaries that matches the empirical distribution in the corpus exactly, and (b) independence assumptions cause an imperfect match between the empirical distribution and any solution containing boundaries, the maximum-likelihood solution corresponds to the empirical word distribution (i.e, $\boldsymbol{\theta}_{\mathrm{ML}}$ applies a unigram distribution of words). While this would hold true for Morfessor Baseline in theory, it actually does not hold true for the current implementation because of the deficiency of the likelihood function (see Sec. 6.4.1.4, p. 188).[21] However, the experimental results still indicate that word lexicon is often close to the ML parameters even with the deficient likelihood function. Morfessor Categories-MAP does not make as crude independence assumptions as Baseline, but there are certainly more dependencies between the morphs than modeled by its four-state HMM, so the ML parameters are again likely to be close to those of the word unigram model.

If the maximum-likelihood parameters would match a word lexicon, how does the prior of the model affect this? The concept of universal coding from Section 2.6.8 provides some answers. If the coding is universal, the regret (difference to the ML parameters for the current data) increases sublinearly in the number of training samples $n = |\boldsymbol{D}_W|$. Then the effect of the prior will diminish and the optimal parameters approach $\hat{\boldsymbol{\theta}}_{\mathrm{ML}}(\boldsymbol{D}_W)$ as n approaches infinity. Whether the Morfessor priors have this property is not evident. However, for Morfessor (and Allomorfessor) Baseline, it is easy enough to show that the description length of the model increases sublinearly with the morph token count ν and linearly with the morph type count μ. The token count, in turn, grows linearly with the number of training samples n. Thus, at least if the training data contains a finite number of word forms so that new morphs will not emerge, the ML parameters will be reached as n grows. The situation is less clear if the training data contains only word types.

6.4.4 Semi-supervised learning of morphology

If the goal of unit selection is to improve the performance of NLP applications or help linguists in language documentation, a relevant question is how much linguistic supervision can improve the unsupervised methods. In the case of morphology, small amounts of annotated data may be relatively easy to obtain. For example, morphological segmentation of an agglutinative language does not require much linguistic expertise, only a well-educated native speaker.

Moreover, as noted by Hammarström and Borin (2011), most of the so-called unsupervised methods for learning morphology actually use at least few thresholds and parameters set by humans. Even in a method as simple as Morfessor Baseline, one has to define at least the parameters of the morph length prior. As

[21] This is easy to see from a trivial training data such as $\boldsymbol{D}_W = (\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{ab})$. Without segmentation, $-\log_2(\boldsymbol{D}_W \mid \boldsymbol{\theta}) = -4\log_2(1/4) = 8$ bits. Segmenting **ab** will give $-\log_2(\boldsymbol{D}_W \mid \boldsymbol{\theta}) = -4\log_2(2/5) - \log_2(1/5) \approx 7.61$ bits.

it is difficult to avoid biases toward certain kinds of languages and analyses, it should in many cases be preferable to base the thresholds and parameters on annotated data.

While there are some work for partially supervised learning (Yarowsky and Wicentowski, 2000; Shalonova et al., 2009), actual semi-supervised learning of morphology has been considered in surprisingly few studies. Two recent exceptions are Snyder and Barzilay (2008a) and Poon et al. (2009). However, they use a small corpus of 6,139 short phrases, and significant proportions (at least 25%) of it is annotated in the semi-supervised task. This contrasts with the very common setting, in which there are plenty of unannotated data and only a tiny annotated corpus.

Semi-supervised learning is more common in other sequential labeling tasks, such as word segmentation, POS tagging, shallow parsing, and named-entity recognition. Especially word segmentation is very similar to morphological segmentation. However, the annotated data sets in popular tasks such as Chinese word segmentation are large, and unannotated data is used just as an additional source of information. For example, Li and McCallum (2005) use an unannotated corpus to obtain clusters of words, which are then used as features for a supervised classifier.

Publication X is the first systematic study of a semi-supervised setting where only a small annotated data set is available. It applies a semi-supervised extension of Morfessor Baseline and evaluates it against a linguistic gold standard analysis using data from Morpho Challenge 2009. With annotated segmentations for one thousand word forms, the algorithm outperforms practically all unsupervised algorithms tested in Morpho Challenges for English and Finnish. Later, a similar semi-supervised evaluation setting was added to Morpho Challenge 2010 (Kurimo et al., 2010b).

6.4.4.1 Semi-supervised Morfessor

In a semi-supervised setup, there are two training data sets. As before, the unannotated data will be denoted $\boldsymbol{D}_W$. The annotated data will be denoted $\boldsymbol{D}_{W \mapsto A}$, as it provides analyses for the word forms in the sets. A slight complication is that $\boldsymbol{D}_{W \mapsto A}$ may include alternative analyses for some of the words. The known analyses for word w_j will be denoted as $\mathrm{A}(w_j) = \{\boldsymbol{a}_{j1}, \ldots, \boldsymbol{a}_{jk}\}$.

Considering the training algorithm of Morfessor Baseline, a straightforward extension to semi-supervised learning is to fix the analyses $\mathbf{Y}$ for the annotated word forms. Assuming the training samples are independent, and giving equal weight for each alternative analysis, the likelihood of the annotated data would be

$$p(\boldsymbol{D}_{W \mapsto A} \mid \boldsymbol{\theta}) = \prod_{j=1}^{|\boldsymbol{D}_{W \mapsto A}|} \prod_{\boldsymbol{a}_{jk} \in \mathrm{A}(w_j)} \prod_{i=1}^{|\boldsymbol{a}_{jk}|} p(m_{jki} \mid \boldsymbol{\theta}). \tag{6.52}$$

However, the training algorithm assumes that the analyses of the words are fixed to a single choice when calculating the likelihood. Then the product over alternative analyses in $\mathrm{A}(w_j)$ is problematic, because some of them might get a zero probability. A sum over $\mathrm{A}(w_j)$ would avoid this problem, but then the logarithm of the likelihood function becomes non-trivial (i.e., logarithm of sum of products) and slow to calculate during the training.

A feasible solution is to use the hidden variable Y to select only one analysis

among $\mathrm{A}(w_j)$ for each sample in the annotated data. The cost function is then

$$L(\boldsymbol{\theta}, \mathbf{Y}, \boldsymbol{D}_W, \boldsymbol{D}_{W \mapsto A}) = -\log p(\boldsymbol{\theta}) - \log p(\boldsymbol{D}_W \mid \mathbf{Y}, \boldsymbol{\theta}) - \log p(\boldsymbol{D}_{W \mapsto A} \mid \mathbf{Y}, \boldsymbol{\theta}), \tag{6.53}$$

and the likelihoods are defined as in the unsupervised case. Because the local search algorithm assumes that a substring is segmented in the same way independent of its context, some morphs in the annotated data may still get a zero probability. In practice, zero probabilities in the likelihood can be treated as large but finite costs.

The above algorithm is simple to implement and it does not add any computational costs compared with the unsupervised algorithm. However, if $\boldsymbol{D}_{W \mapsto A}$ is small, it will have little effect on the results: a larger $\boldsymbol{D}_W$ and the prior of the model will dominate the outcome. For example, with unannotated data of 2.2 million Finnish word types, 10,000 annotated word types had practically no effect to the segmentation results.

Given the discussion in Section 6.4.3, the solution should be clear enough: use weight parameters to modify the balance of the likelihoods and the prior. The weighted likelihood is

$$\begin{aligned} &L(\boldsymbol{\theta}, \mathbf{Y}, \boldsymbol{D}_W, \boldsymbol{D}_{W \mapsto A}) = \\ &\quad -\log p(\boldsymbol{\theta}) - \alpha \log p(\boldsymbol{D}_W \mid \mathbf{Y}, \boldsymbol{\theta}) - \beta \log p(\boldsymbol{D}_{W \mapsto A} \mid \mathbf{Y}, \boldsymbol{\theta}), \end{aligned} \tag{6.54}$$

where α controls the effect of the unannotated data and β the effect of the annotated data. Another parametrization would be $\alpha = \hat{\alpha}(1 - \hat{\beta})$ and $\beta = \hat{\alpha}\hat{\beta}$, in which $\hat{\alpha} > 0$ controls the amount of segmentation and $0 \leq \hat{\beta} \leq 1$ the balance between annotated and unannotated data. Regardless of the parametrization, the weights have to be optimized on a separate held-out set, adding another layer of supervision.

A similar weighting approach has been used, for example, by Nigam et al. (2000) for semi-supervised text classification. They call the modified algorithm as EM-λ, and select the weight parameter λ by leave-one-out cross-validation.

6.4.4.2 Experiments on semi-supervised segmentation

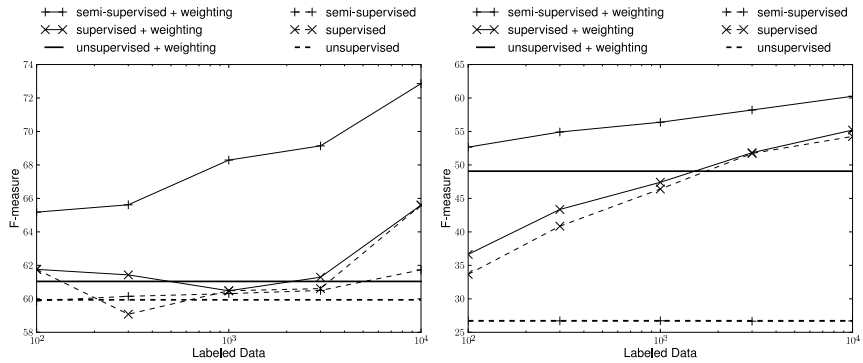
The main experiments of Publication X compare six different variants of the Morfessor Baseline algorithm:

- **Unsupervised:** The standard unsupervised Morfessor baseline.
- **Unsupervised + weighting:** A held-out set is used for adjusting the weight of the likelihood α. When $\alpha = 1$ the method is equivalent to the unsupervised baseline.
- **Supervised:** The semi-supervised method trained with the annotated data only. That is, Morfessor just selects which of the alternative analyses of the gold standard segmentation to use. A similar procedure has been independently proposed by Mihajlik et al. (2010).
- **Supervised + weighting:** As above, but the weight of the likelihood β is optimized on the held-out set. The weight can only affect which segmentations are selected from the possible alternative segmentations in the labeled data.
- **Semi-supervised:** The semi-supervised method trained with both annotated and unannotated data but without likelihood weighting.

- **Semi-supervised + weighting:** As above, but the parameters α and β are optimized using the the held-out set.

The models are trained for English and Finnish data and evaluated using the linguistic gold standard evaluation of Morpho Challenge 2009 (Kurimo et al., 2010c). For supervised and semi-supervised methods, the amount of annotated (segmented) data is varied between 100 and 10,000 words. The held-out set has gold standard analyses for 500 words. After training the model, the augmented Viterbi algorithm was applied to find the optimal segmentation of each word in the final test data.

The results are shown in Figure 6.20. For English, the baseline F-measure of 60% is increased to 65–73% depending on the amount of annotated data. 1,000 samples are enough to outperform the best unsupervised approaches such as 66.24% by Bernhard (2008). For Finnish, the baseline F-measure 27% is increased to 53–60%, outperforming the 52.45% by Bernhard (2008) already with 100 annotated samples.



(a) English (b) Finnish

Figure 6.20. The F-measure for unsupervised, supervised, and semi-supervised Morfessor Baseline as a function of the number of annotated training samples (Publication X).

Supervised learning starts to work reasonably well on both languages if more than one thousand annotated samples are available. The main difference between English and Finnish results is the effect of the likelihood weighting for the unsupervised results. For English, the balance of likelihood and prior is almost optimal without weighting, but for Finnish, the weight of the likelihood has to be decreased in order to obtain balanced precision and recall.

Table 6.11 shows the weight parameters α and β and the results for weighted semi-supervised algorithm. The optimal β decreases with the increase in annotated data, with the exception of 100 samples for Finnish. For English, the optimal α is always close to one, while for Finnish, it varies between 0.005 and 0.1. Both precision and recall of the English results improve with the amount of data, while the recall of the Finnish results ceases to increase already after the first 100 samples. Thus it seems that for Finnish, the recall of the MC evaluation cannot be increased solely by improving the segmentation.

While the main task in Publication X is morphological segmentation, it also considers using the annotated data to tag the morphs to the morpheme labels defined by the gold standard analysis. This would be a very practical solution for dealing with allomorphy in the affix morphemes, which is, as discussed in Section 6.4.2, a hard task for any unsupervised algorithm. The preliminary ex-

Table 6.11. Precision, recall, F-measure and the values for the weights α and β that the semi-supervised algorithm chose for different amounts of labeled data when optimizing the F-measure. The last row shows the results of context-free morpheme labeling made after the segmentation.

	English					Finnish				
$\lvert D_{W \mapsto A} \rvert$	α	β	Pre	Rec	F	α	β	Pre	Rec	F
0	0.75	-	68.48	55.07	61.05	0.01	-	53.72	45.16	49.07
100	0.75	750	67.82	62.74	65.18	0.01	500	50.67	54.81	52.66
300	1	500	71.38	60.73	65.63	0.005	5000	55.65	54.21	54.92
1000	1	500	69.72	66.92	68.29	0.05	2500	61.03	52.38	56.38
3000	1.75	350	76.85	62.82	69.13	0.1	1000	65.39	52.45	58.21
10000	1.75	175	77.35	68.85	72.85	0.1	500	69.14	53.40	60.26
+labeling	1.75	175	77.07	77.78	77.42	0.1	500	66.90	74.08	70.31

periment in Publication X applied a simple context-free tagging: each segment in the test data was replaced by the most common label it had in the annotated training data whenever such was available. As shown in the last row of Table 6.11, labeling provided high increase in the recall of the evaluation metric. With the 10,000 word annotated data, F-measure increased to 77.42% for English and to 70.31% for Finnish. The labeling approach has been developed further by Kohonen et al. (2010) and evaluated in Morpho Challenge 2010.

6.4.5 Learning of phrasal constructions

As the previous sections have shown, the Morfessor method is relatively easy to extend to various problems in morphological unit selection, at least as long as they can be mainly solved by segmentation. Related MDL-based approaches have been applied also to unsupervised word segmentation (de Marcken, 1996; Brent, 1999). A natural question is then whether a similar approach would be useful also for extracting units that encompass several words.

The most similar problem to the morphological segmentation in the level of sentences would be shallow parsing (chunking). Instead of using completely flat chunks, Publication XI considers a model that encodes sentences with a slightly more complex phrasal constructions. The applied generalization is that each construction may have a slot for a word that is not fixed but selected from a particular construction-specific category. Figure 6.21 illustrates this type of a model.

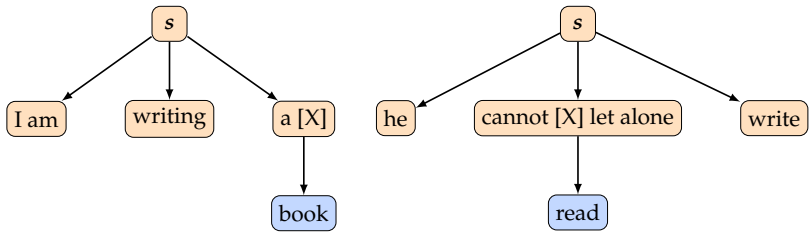


Figure 6.21. Two examples of the type of sentence parses enabled by the construction model of Publication XI.

6.4.5.1 Related work

On one hand, the task of unsupervised learning phrasal constructions can considered as a part of grammar induction. Most work on this topic has focused on learning context-free grammars, either from a theoretical (Clark, 2001; Clark et al., 2008, 2010) or practical (Chen, 1995; de Marcken, 1996; Adriaans et al., 2000; van Zaanen, 2000; Klein and Manning, 2002; Seginer, 2007) point of view. As noted in Section 3.2.4, parsing with a CFG is equivalent to hierarchical segmentation of the word sequences, bracketing. Unsupervised grammar learning, if evaluated against a treebank, is a difficult task: Klein and Manning (2002) were the first to outperform a simple right-branching heuristic. Most methods rely on that the input corpus is part-of-speech tagged, and using induced tags often degrades the results (Klein and Manning, 2002; Cramer, 2007). Moreover, while the results of the well-known methods are acceptable for commonly used English evaluation corpora, they fail for more complicated data sets (Cramer, 2007).

On the other hand, the task of Publication XI is similar to shallow parsing, that is, segmenting words to their low-level constituents. An early example of unsupervised shallow parsing is the multigram model by Deligne and Bimbot (1995, 1997). Their model is actually very similar to Morfessor Baseline, but they use ML estimation with heuristic complexity control instead of MDL, and train the model with EM and Viterbi algorithms (with similar results for both). Apparently, their model has never been evaluated as a shallow parser.

Recently, Ponvert et al. (2010) have shown that a state-of-the-art unsupervised parser by Seginer (2007), CCL, is actually outperformed by taking *only* the low-level constituents and building a right-branching tree based on them, indicating that simply solving the unsupervised shallow parsing can help with the deep parsing. Ponvert et al. (2011) show that HMM and PRLG[22] trained with the EM algorithm can outperform CCL in shallow parsing. Moreover, they show that a cascade of shallow parsers can be used to get state-of-the-art results also in deep parsing.

Regarding other types of grammars, dependency grammar induction has been studied, for example, by Chen (1995) and Klein and Manning (2004). However, more interesting from the point of construction learning is the unsupervised data-oriented parsing (U-DOP) model by Bod (2006, 2007, 2009b). In DOP, sentences are created by combining tree structures from previously seen trees, and it is compatible with several linguistic grammar formalisms, including unification grammars and tree-adjoining grammars (Bod, 2009b). Similarly to the (shallow) constructions that are learned in Publication XI, the trees can be discontiguous. Bod (2009b) has evaluated U-DOP both against treebanks and in a psycholinguistic setting, where it is shown to mimic children's language development.

6.4.5.2 Model and priors

As illustrated by Figure 6.21, the model considered here is defined to learn two types of constructions:

- Word sequences of different lengths: **went to**, **red car**, **and**

[22] PRLG or probabilistic right linear grammar is a extension of HMM, in which the emission x_i is dependent both the current state y_i and the previous state y_{i-1}.

- Sequences that contain one slot for words of a specific *category*, where a category refers simply to a group of words that is expected to be used within this sequence: **went to buy [X], [X] was**.

If only the former kind of structure is allowed, the model is equivalent to the Morfessor Baseline model, but for sentences consisting of words instead of words consisting of letters. Initial experiments with such a model showed that while the algorithm finds sensible structure, the constructions found are very redundant and therefore impractical and difficult to interpret. The latter construction type was included to get more interesting results. Only one free slot per construction is allowed in order to make the training fast.

The likelihood of the training data $\boldsymbol{D}_S$ is

$$\begin{aligned} p(\boldsymbol{D}_S \mid \mathbf{Y}, \boldsymbol{\theta}) &= \prod_{j=1}^{|\boldsymbol{D}_S|} p(\boldsymbol{s}_j \mid \mathbf{Y}, \boldsymbol{\theta}) \\ &= \prod_{j=1}^{|\boldsymbol{D}_S|} \prod_{i=1}^{|\mathbf{y}_j|} p(x_{ji} \mid \boldsymbol{\theta}) p(w_{ji} \mid x_{ji}, \boldsymbol{\theta}), \end{aligned} \tag{6.55}$$

where $\mathbf{Y}$ contains the analyses of the sentences $\boldsymbol{s}_j$: constructions $\mathbf{x}_{ji}$ and their category words w_{ji}.

Following Morfessor, the prior of the model with μ constructions is set to

$$p(\mathcal{L}) = p(\mu) \times p(\text{properties}(x_1), \ldots, \text{properties}(x_\mu)) \times \mu!. \tag{6.56}$$

The properties of the constructions x_i include:

- Lengths of the construction: $p(|x_i|) = 1/l_{\max}$, where $l_{\max}$ is the maximum length (for example, the length of the longest sentence in the training data).
- Indicator β_i that tells whether the construction has a category slot: $p(\beta_i = 0) = p(\beta_i = 1) = \frac{1}{2}$.
- Fixed words of the constructions $\boldsymbol{\omega}_i$:

$$p(\boldsymbol{\omega}_i) = \prod_{j=1}^{|\boldsymbol{\omega}_i|} p(\omega_{ij}), \tag{6.57}$$

  where $p(\omega_{ij})$ is the probability of the j^{th} word of the construction (here based on ML estimates from the training data).
- Counts of the constructions τ_i:

$$p(\tau_1, \ldots, \tau_\mu \mid \mu, \nu) = 1 / \binom{\nu - 1}{\mu - 1}, \tag{6.58}$$

  where ν is the token count of the constructions.
- If the construction has a category slot: the number of words in the category (bounded by the total number of words), a list of the category words (similarly to Equation 6.57), and the counts of the category words (similarly to Equation 6.58).

6.4.5.3 Search algorithm

Training the model with the EM algorithm has the same problem of sparse, discontinuous prior as in other MDL-based models. Moreover, the standard local

training algorithm of Morfessor Baseline is not applicable because of the word categories. Instead, the local optimization procedure applied in Publication XI is the following:

1. Initialize the analysis so that each word is a construction by itself and there exist no other constructions.
2. Generate all possible constructions of length $\leq t_L$ and frequency $\geq t_F$ from the corpus.
3. Sort the candidate constructions by likelihood ratio

$$r(x) = \frac{p(\boldsymbol{D}_S \mid \mathbf{Y}^{+x}, \boldsymbol{\theta}^{+x})}{p(\boldsymbol{D}_S \mid \mathbf{Y}, \boldsymbol{\theta})}, \tag{6.59}$$

   where $\boldsymbol{\theta}^{+x}$ includes the construction x in the lexicon and $\mathbf{Y}^{+x}$ uses it wherever possible.
4. In the descending order of likelihood ratios:
   (a) Store the current value of the cost function $L(\boldsymbol{\theta}, \boldsymbol{D}_S, \mathbf{Y})$.
   (b) Apply the construction to all sentences where applicable and calculate the value of the cost function $L(\boldsymbol{\theta}^{+x}, \boldsymbol{D}_S, \mathbf{Y}^{+x})$. If the construction improves the cost, accept the changes, otherwise discard them.
   (c) Proceed from (a) with the next construction.

As each construction is tested only once, the training is relatively fast. The thresholds were set to $t_L = 6$ and $t_F = 11$.

6.4.5.4 Experiments

There is no evident quantitative evaluation methods for the kind of constructions induced by the method. To make the manual inspection of the results interesting, the method was applied to a corpus consisting of stories told by Finnish children. Compared with data observed by children, as in the commonly (e.g. Berant et al., 2007; Bod, 2009b) applied Childes corpus by MacWhinney (2000), data *produced* by children provides a better look at the representations that are actually used by a cognitive system in the middle of its development. Moreover, such a corpus should have many frequent and simple constructions to observe.

The corpus contains 2,642 stories told by children to an adult, typically a member of day care personnel or a parent. The adults were instructed to write the story down exactly as it was told, without changing or correcting anything. A minority of the stories were told together by a pair or by a group of children. The age of the children varied from 1 to 7 years. The story mark-up contains the age and the first name(s) of the storyteller(s). The stories contain a lot of spoken-language words and even some non-grammatical forms. For example, a story told by a three year old girl, is the following:

> **Mun *äitin nimi on äiti. Mun iskän nimi on iskä. Iskä tuli mun kanssa tänne. Mun nimi on Oona. Jannen nimi on Janne.**[23]
> (*My mommy's name is mommy. My daddy's name is daddy. Daddy came here with me. My name is Oona. Janne's name is Janne.*)

[23] Correct genitive inflection of **äiti** (*mother*) would be **äidin** instead of ***äitin**. **Mun** (*my*) and **iskä** (*daddy*) are colloquial words.

Another story, told by a boy of 5 year and 11 months, is as follows:

> **Dinosaurus meni kauppaan osti sieltä karkkia sitten se meni kotiin ja söi juuston. Sitten se meni lenkille ja se tappoi pupujussin iltapalaksi ja sitten se meni uudestaan kauppaan ja se ei *näkenyt mitään siellä kun kauppa oli kiinni.**[24]
> (*A dinosaur went to a shop and bought there candy then it went home and ate a cheese. Then it went for a walk and it killed a bunny for an evening snack and then it went again to the shop and it didn't see anything there because the shop was closed.*)

A more extensive description of the corpus is given by Klami (2005).

The stories were preprocessed by removing the headers and replacing punctuation marks with a single symbol #. Then each story was divided into sentences. After preprocessing, the total number of sentences in the corpus was 36,542. The number of word tokens was 244,274 and word types 24,242.

Table 6.12 shows the most frequent constructions that the algorithm has discovered. The algorithm was successful in that the frequent constructions can often be considered meaningful. For example, **"olipa kerran [X]"** (*once upon a time there was a [X]*) is the archetypical way to start a fairy tale in Finnish. The prominence of **"ja sitten"** (*and then*) is caused by many stories following a pattern where the child explains some event, then uses **"ja sitten"** to move on to the next event, and so on. The method has also found several variants of this construction: **sit**, **sitt**, and **sitte** are all spoken language forms of **sitten**.

Table 6.12. The most frequent two- and three word constructions with their five most frequent category words.

Count	Form	Category words (count)
891	hän [X] *he [X]*	meni (68), oli (50), lähti (32), löysi (29), otti (19) *went, was, left, found, took*
885	ja sitten *and then*	
798	[X] on *[X] is*	se (82), hän (24), täällä (20), tässä (20), nyt (17) *it, he/she, here, here, now*
768	meni [X] *went [X]*	metsään (33), ulos (33), sinne (30), # (25), nukkumaan (18) *(into the) forest, outside, there, #, (to) sleep*
694	sit [X] *then [X]*	se (302), ne (81), kun (20), hän (17), # (12) *it, they, when, he/she, #*
632	ja [X] se *and [X] it*	sitten (303), sit (155), sitte (109), sitt (18), kun (5) *then, then, then, then, when*
337	[X] se meni *[X] it went*	sitten (125), sit (66), sitte (58), ja (35), kun (14) *then, then, then, and, when*
245	olipa kerran [X] *once (upon a time) there was (a) [X]*	pieni (8), tyttö (7), yksi (6), koira (6), hiiri (5) *little, girl, one, dog, mouse*
235	ja [X] ne *and [X] they*	sitten (129), sit (37), sitte (28), kun (6), niin (5) *then, then, then, when, so*
197	ja [X] tuli *and [X] came*	sitten (91), se (9), sinne (6), ne (4), niistä (3) *then, it, there, they, (of) them (be-)*

Many of the categories found by the method appear to consist of one or a few semantic or syntactic (POS) categories. For example, **söi [X] #** (*ate [X] #*) contains mostly edible arguments: **banaania** (*banana*), **mansikkaa** (*strawberry*), **jäniksen** (*a rabbit*) or a pronoun **hänet** (*him/her*), **ne** (*them*). However, some categories are too general. For example, **"meni metsään"** and **"meni #"** are analyzed as **"meni**

[24] Punctuation is missing between the first two clauses (probably a mistake of the writer). The correct inflection of **nähdä** (*see*) would be **nähnyt** instead of ***näkenyt**.

[X]", although in the former case the free slot is the argument of the verb, and in the latter case the verb takes no arguments, but happens to be at the end of a sentence.

Whereas the frequent constructions found by the method are fairly reasonable, the analyses of individual sentences generally leave much of the structure unanalyzed. Consider, for example, the following analysis:

että hirveä hai tuli niitten [perään {X → ja}] [söi {X → ne}] #
that terrible shark came them [after {X → and}] [ate {X → them}] #

A large part of the sentence is not analyzed as any abstract construction. There are a lot of possible constructions that might be expected, but that the algorithm does not discover: for example, constructions such as **[X] hai**, where the category contains adjectives or **hai [X]**, where the category contains an action verb. Note also that both of these two constructions could not currently be used at the same time, but one would have to choose one or the other.

6.4.5.5 Discussion

In contrast to the related work discussed in the beginning of this subsection, Publication XI does not present a full-fledged method for unsupervised parsing or grammar induction. It is rather a proof of concept for demonstrating that non-trivial constructions, that are neither complete shallow nor fully hierarchical (i.e., equivalent to CFG rules), can be found with an MDL-inspired model similar to Morfessor. Prior to quantitative evaluations, improvements would be required at least to the training algorithm to make it possible to train the model on larger data sets. However, as shown by Spitkovsky et al. (2010a), a useful approach is to first concentrate on training samples that are not too complex.

Direct evaluations for grammar inference are based on comparisons to treebanks. It is not clear what should be considered as the low-level constituents in the case of the open-slot constructions. However, the same algorithm could also be used for induction of complete parse trees using cascaded models similarly to Ponvert et al. (2011). That is, each item of a construction ω (either fixed or category-specific) could be a construction found by the previous model of the cascade.

As the model defines a probability distribution over sentences, one simple way to evaluate the results would be a statistical language modeling evaluation along the lines of Deligne and Bimbot (1995). They obtain improved perplexity over standard n-gram models for a small telephone conversation data set, but it is unlikely that such a "phrase unigram" model would outperform modern varigram models for predicting open domain data.

A more suitable application for a phrase-learning algorithm would be statistical machine translation. The process of extracting the phrase pairs is often quite heuristic in the current SMT systems, and the phrases do not commonly include any abstract categories. Finding more abstract constructions should help to alleviate the data sparsity problems.

Another potential evaluation and application domain is in document modeling for IR and text categorization tasks. Intuitively, vector space models based on bag-of-constructions should be able to outperform bag-of-words. The constructions could also be used in probabilistic topic modeling similarly to Shen et al. (2006).

7. Conclusions and future directions

This thesis has centered around the question of lexical unit selection and its evaluation: how text strings—in any written language—should be processed to get discrete units suitable for statistical language modeling or representation learning, and how the results of the unit selection should be evaluated.

The main hypothesis was that machine learning is useful for the unit selection problem. Especially unsupervised and minimally supervised learning should help in adapting the models for different languages and domains. The machine learning approach for unit selection can also be called the task of learning *constructions*, form-meaning pairs of language considered in the construction grammars.[1]

The work discussed in this thesis, including both the publications of the thesis and the reviewed research, demonstrates that minimally supervised learning of constructions is indeed useful for many purposes. This chapter summarizes the main results and discusses their implications as well as promising directions for future research.

7.1 Models for learning constructions

The minimal, non-complex constructions are morphemes. Assuming the existence of morphemes in a language is one of the most universal, language independent assumptions that can be made. Thus the first step in learning the constructions of a language is identifying its morphemes.

Regarding the algorithms for morphology induction, this thesis has provided two new contributions: Allomorfessor is still one of very few unsupervised algorithms that do not consider only segmentation, but also non-concatenative processes that produce allomorphs. The semi-supervised Morfessor is, to the best of our knowledge, the first semi-supervised algorithm shown to get significant improvements from small amounts of annotated training data.

These two algorithms for morphology induction have been evaluated in the original Morpho Challenge evaluations. As a part of this thesis, they have been re-evaluated along with all other algorithms submitted to Morpho Challenges 2007–2010 using the new evaluation methods. Thus their performance can now be reconsidered. Appendix A.2 shows complete results of the direct and indirect evaluations, using EMMA (Spiegler and Monson, 2010) as the direct evaluation

[1] To be precise, the algorithms discussed in this thesis consider only learning of the forms of the constructions, not their meanings.

method.

Allomorfessor fares moderately in the linguistic evaluations, being in the top half of the methods for all languages and in the top third for most languages (English rank 14 of 49, Finnish 14/42, German 7/39, and Turkish 20/45). The same holds for application evaluation in information retrieval tasks (English 11/42, Finnish 10/37, German 13/31). However, in the machine translation tasks, Allomorfessor yields the top results both for Finnish and German.

The semi-supervised Morfessor algorithm, especially when combined with morpheme labeling, outperforms unsupervised Morfessor by a large margin and gets the top results among all algorithms for all tested languages (English, Finnish, and Turkish). And this is albeit the fact that the weight parameter of the model was optimized on the F_1-score of the MC evaluation method, not the newer EMMA method that is considered here.[2] The results of the application evaluations are only moderate, but also they might improve significantly if the weight parameter was optimized for a direct measure that has a high correlation with the application evaluations (such as F-score of EMMA or optimized F_β-score of any other isomorphic evaluation method).

A combination of the presented methods—Allomorfessor and semi-supervised Morfessor—is likely to be a significant step towards a high-accuracy probabilistic morphological analyzer. On one hand, Allomorfessor is well-suited for learning regular transformations of the stems. On the other hand, frequent and irregular words and their transformations can be easily learned in a semi-supervised manner. This line of research will certainly be pursued in the future.

While the MDL-inspired prior of Morfessor is an essential part of its success, the study of the effects of a training corpus reveals a drawback that calls for further discussion: increasing the amount of training data decreases the amount of segmentation and increases the size of the morpheme lexicon. From the theoretical side, using a universal coding (in which the regret increases sublinearly in the number of samples; see Section 2.6.8) for a model family means that the optimal hypothesis will be the one that gives the maximum likelihood when the number of training samples approaches infinity. This is not a desired property for Morfessor Baseline, for which the maximum-likelihood parameters are—at least approximately—equivalent to using a word lexicon. Known solutions to this problem include training the model with word types (or using dampened word frequencies), using more realistic model dependencies between the morphemes or a hierarchical lexicon (as in Morfessor Categories-MAP), or decreasing the likelihood weight (as proposed in the semi-supervised extensions). Yet another solution would be gradually forgetting (i.e., removing it from the likelihood term) the past data while processing new data.

The example of phrasal construction learning presented in this thesis is rather a proof of concept than a practical unit selection method. As such, it has not been evaluated in a quantitative manner, but the initial experiments show that it has potential for finding constructions useful also for practical applications if the model and the training algorithm are developed further.

[2] Using the wrong optimization criterion shows in the Turkish results of Morfessor U+W algorithm, which has an optimized likelihood weight but does not otherwise use labeled training data: it severely oversegments the words.

7.2 Direct evaluations

The goal for learning constructions of a language is usually to improve the performance of the NLP applications. While some general criteria for useful constructions can be identified (see Section 6.1.1), different applications require different types of constructions. As the indirect evaluations in applications are often complicated and expensive in terms of time and manual work required, it may still be useful to rely on direct evaluations if such are available.

Direct evaluations that utilize available linguistic resources provide the most straightforward goal for unit selection. For example, the first goal for a morphology induction algorithm should be to get something that resembles the morphological analyses provided by experts on linguistics. This thesis includes the first major study of evaluation methods for comparing the results of unsupervised algorithms with linguistic analyses. Hopefully the results and the new evaluation methods will help researchers working on this area.

As optimizing the score of a linguistic evaluation is not the main goal, a relevant question is how well they can predict the performance in application evaluations. The experiments of this thesis have shown that with a suitable direct measure with optimized parameters, the scores are likely to have high correlations with application evaluation scores also for novel sets of algorithms. However, this does not indicate that application evaluations would be unnecessary. A simple reason is that the best unsupervised algorithms may actually outperform the grammatical gold standard analyses for the applications. In other words, the grammatically correct analysis is not necessarily the optimal solution for the applications. The results of this thesis show that if the weight parameter between precision and recall in F_β-scores of the direct evaluations is optimized to maximize the correlation between the F_β-score and the scores of the application evaluations, precision is weighted over recall. This is particularly visible for the agglutinative Finnish and German languages, and indicates that undersegmentation is sometimes useful for the applications. It is possible that this tendency originates from application-specific biases, as the systems are developed usually for English. However, the phenomenon can also be considered in the context of the psycholinguistic discussion on whether inflected words are stored as full-forms or inferred from their morphological parts in the human mind (e.g., Pinker and Ullman, 2002; Baayen, 2007). That is, the mental lexicon is not necessarily minimal, but may incorporate redundant forms whenever they make the processing efficient enough compared with the additional cost of memory resources.

It is well possible that the same (unconscious) language processing strategies applied by humans would be useful also for computational methods. This raises the question on whether psycholinguistic studies can be used to evaluate tasks such as morphology induction. In this thesis, it is shown that one promising type of evaluation is the prediction of reaction times in a lexical decision task. While behavioral reaction time studies are very common in psycholinguistic research, this is apparently the first time that an unsupervised method for morphology induction has been evaluated in this type of a task.

The results were promising in that the probability estimates from the Morfessor models had high correlations with the human reaction times of Finnish nouns: Spearman's correlation coefficient was almost 0.7. Equally high correlations were provided by a letter-based n-gram model, although with much larger

number of model parameters. It seems that even higher correlations could be easily obtained if there was a data set more similar to what humans observe in the course of their life. This is clear evidence for that probabilities are relevant in human language processing. However, any further conclusions are hard to draw, as behavioral reaction times do not provide information on different stages of visual word processing. Neurolinguistic experiments would be a next step towards the cognitive evaluation of the probabilistic models.

The direct evaluations are applicable only if there are suitable linguistic (or psycholinguistic) resources available. They are not possible for many other types of constructions, including the phrasal constructions considered in Publication XI. In fact, the development of the direct evaluation method for vector representations of documents presented in Publication III was partially motivated by the lack of simple evaluation methods for learning this kind of constructions.

The CCA-based evaluation method is apparently the first direct evaluation method proposed for document representations. It passed all except one of the validation experiments, and seems to work very well for evaluating different weighting schemes and dimensionality reduction methods. The only unexpected result was that the evaluation yielded higher scores when simple n-gram features were appended to the standard bag-of-words features. While this may be explained by the short documents (individual sentences) used in the experiments, further study is required to confirm how selecting partially redundant features affects the canonical correlations.

7.3 Applications

The general goal stated in the beginning of this thesis was to use machine learning to solve the fundamental learning problems encountered in the NLP applications. Thus it is appropriate to conclude this overview by discussing the three applications that the thesis has mainly concentrated on: automatic speech recognition, information retrieval, and machine translation.

For ASR, both the studies reviewed and included in this thesis suggest a simple approach for the unit selection problem. First, one should predict as small units as it is possible to do without increasing acoustic confusability. Morphs are good candidates for agglutinative languages, as demonstrated in various studies (see Hirsimäki et al., 2009). Even smaller units may be useful in order to construct unseen morphs in the case that there is enough information from the acoustic model. For phonemic writing systems, the units may simply be letters. As argued by Heeman and Damnati (1997), larger units are useful only if they help in acoustic modeling. An interesting direction is parallel learning of morphological segmentation and acoustic units (Deligne and Bimbot, 1997).

Second, while there seems to be no reason to use complex units in the target distribution, the situation is different for context units. In fact, the language model should base its predictions on as large context units as possible, as long as they are frequent enough for reliable estimates. The methods for growing, pruning, and clustering the histories of the n-gram models introduced in this thesis provide practical tools for this task. As such, they are especially suitable for morph-based models—there is no sense in growing the n-gram model one *letter* at a time—but they could as well be extended to have different predictive

units and predicted units.

While the n-gram models and their simple extensions, such as varigram models, are still *de facto* standard in any application that requires statistical language modeling, it seems that new methods are gradually outperforming them by larger and larger margins. The review on state-of-the-art language modeling techniques reveals two promising approaches: hierarchical Pitman-Yor (HPY) language models (Teh, 2006)—especially if extended to infinite Markov models (Mochihashi and Sumita, 2008) and domain adaptation (Wood and Teh, 2009)—and recurrent neural network (RNN) language models (Mikolov et al., 2010).[3] While especially the RNN model differs substantially from the standard n-gram model, the unit selection approach sketched above should still work fine. An additional benefit for the neural network models would be that predicting units from a smaller lexicon would make the normalization of the probabilities quicker.

In spite of the promising results, it is worth noticing that the payback from the sophisticated models and smoothing techniques depends on the size of the training data. Brants et al. (2007) have compared interpolated Kneser-Ney smoothing with "stupid back-off", a very simple smoothing scheme that uses no discounting, fixed back-off weight, and unnormalized scores. Their task was re-scoring the output of a machine translation system. The size of the language model training data was up to 2 trillion (2×10^{12}) tokens. For 5-gram models trained on 10^{10} words and above, there was no statistically significant difference between KN smoothing and stupid back-off. For 10^{11} words and above, KN models were too expensive to train, while the BLEU scores of the "stupid back-off" models continued to grow.

As both the amount of data and the computational power grows, it is not clear whether the best approach will be to use as much data as possible with very simple models, or moderate amounts of in-domain data with sophisticated modeling techniques. Maybe the answer is still in between—using models that are simple and quick to train but that use clever approximations from theoretically well-grounded models. Such approximations include the Kneser-Ney smoothing (Kneser and Ney, 1995) as well as the power law discounting (Huang and Renals, 2010b) techniques. The revised Kneser pruning and growing algorithms of Publication I should be relatively simple to extend to apply the power law discounting that has given as good results as the HPY language models.

While coverage of the lexical units seems to be the essential criterion for speech recognition, the problem of unit selection is more intricate for information retrieval and machine translation. In these applications, the semantic relevance of the units is more crucial.

While IR has not been in the center of this thesis, there are a few points to be made. First, the Morpho Challenge evaluations show that traditional rule-based stemmers and analyzers may already be replaced by unsupervised learning of morphological segmentation or clustering without essentially hurting the performance. While it seems to be hard to obtain improvements from constructions that encompass several words, some recent studies (e.g., Shen et al., 2006; Koster et al., 2011) have shown that it is still possible. This is one clear application for unsupervised learning of phrasal constructions.

In statistical machine translation, the processing of morphologically complex

[3] Apparently, HPY and RNN language models have not so far been experimentally compared.

languages is still an unsolved problem. This thesis has proposed using morpheme-like units induced in an unsupervised manner instead of words as a general and language-independent solution. Even if the current machine translation systems are not particularly developed for morpheme-like units, significant improvements could be obtained using the MBR system combination for word and morpheme based models. As illustrated by the Morpho Challenge experiments, the methods of the Morfessor family seem to be well-suited for this application.

Also the learning of phrasal constructions is relevant for SMT. For example, a recent study by Turchi et al. (2012) suggests that the factor limiting the performance of an SMT system is not in the numeric parameters, but the entries of the phrase translation table. The current phrase extraction algorithms are often based only on the word alignments. An algorithm for learning phrases from monolingual corpora may provide additional information for the extraction.

As statistical machine translation anyway requires a bilingual corpus, a promising approach is to learn suitable constructions from bilingual data. Bilingual unit selection for SMT has already been tested for both morphological segmentation (e.g., Fishel, 2009; Mermer and Akın, 2010) and word segmentation (e.g., Chung and Gildea, 2009; Paul et al., 2010).

While the use of multilingual data is especially relevant for machine translation, it is likely to be useful for the learning problems of NLP in general. Parallel sentences or documents, as different views for the same underlying semantics, provide an indirect way to take the meanings of the constructions into account. For example, if a particular construction is used in one language, another construction with a similar meaning (but potentially a very different form) should be observed in the other language. Different learning settings can be considered depending on whether there is annotated data available in one language (e.g., Yarowsky et al., 2001) or not (e.g., Snyder and Barzilay, 2010). Essential prerequisites for this very promising line of research are efficient monolingual models and evaluations suitable for a wide variety of languages—such as those developed in this thesis.

A. Appendices

A.1 Proof for optimal feature generators in linear bilingual document model

This appendix gives the proof that justifies the CCA-based evaluation method presented in Section 5.4 for the bilingual document model in the case of full-rank and non-singular projections. This is a slightly elaborated version of the argumentation presented in Publication III.

From Section 5.4.1, recall that $\mathbf{U}$ and $\mathbf{V}$ are matrices containing the canonical variates found by the canonical correlation analysis, and $\mathbf{A}$ and $\mathbf{B}$ the respective projections. For generating processes G_s and G_t and feature generators $\mathfrak{F}_s$ and $\mathfrak{F}_t$,

$$\mathbf{U}\mathbf{V}^{\mathrm{T}} = \mathbf{A}^{\mathrm{T}}\mathfrak{F}_s(\mathrm{G}_s(\mathbf{W}_s\mathbf{Z}))\mathfrak{F}_t(\mathrm{G}_t(\mathbf{W}_t\mathbf{Z}))^{\mathrm{T}}\mathbf{B}. \tag{A.1}$$

By definition, CCA finds $\mathbf{A}$ and $\mathbf{B}$ that maximize the correlations of the canonical variates with orthogonality constraints. Let $\mathbf{Z} \in \mathbb{R}^{D_z \times N}$, $\mathbf{W}_s \in \mathbb{R}^{D_z \times D_z}$ and $\mathbf{W}_t \in \mathbb{R}^{D_z \times D_z}$ be non-singular. Our claim is that the highest possible correlations are obtained in the case that $\mathfrak{F}_s(\mathrm{G}_s(\mathbf{Z}_s)) = \mathbf{Z}_s$ and $\mathfrak{F}_t(\mathrm{G}_t(\mathbf{Z}_t)) = \mathbf{Z}_t$.

Proof. Consider the choice of $\mathbf{A}$ and $\mathbf{B}$ in

$$\mathbf{U}\mathbf{V}^{\mathrm{T}} = \mathbf{A}^{\mathrm{T}}\mathbf{W}_s\mathbf{Z}\mathbf{Z}^{\mathrm{T}}{\mathbf{W}_t}^{\mathrm{T}}\mathbf{B}. \tag{A.2}$$

Because $\mathbf{C}_z = \mathbf{Z}\mathbf{Z}^{\mathrm{T}}$ is non-singular and symmetric, there exists an eigenvalue decomposition $\mathbf{C}_z = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{\mathrm{T}}$, where $\mathbf{\Lambda}$ is a diagonal matrix of eigenvalues $\lambda_1, \ldots, \lambda_{D_z}$ and $\mathbf{Q}$ is an orthogonal matrix of eigenvectors. Thus $\mathbf{Z} = \mathbf{Q}\mathbf{\Lambda}^{\frac{1}{2}}$. Setting $\mathbf{A} = ({\mathbf{W}_s}^{-1})^{\mathrm{T}}\mathbf{Q}\mathbf{\Lambda}^{-\frac{1}{2}}$ and $\mathbf{B} = ({\mathbf{W}_t}^{-1})^{\mathrm{T}}\mathbf{Q}\mathbf{\Lambda}^{-\frac{1}{2}}$ gives:

$$\mathbf{U}\mathbf{V}^{\mathrm{T}} = \mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{Q}^{\mathrm{T}}{\mathbf{W}_s}^{-1}\mathbf{W}_s\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{\mathrm{T}}{\mathbf{W}_t}^{\mathrm{T}}({\mathbf{W}_t}^{-1})^{\mathrm{T}}\mathbf{Q}\mathbf{\Lambda}^{-\frac{1}{2}} = \mathbf{I}. \tag{A.3}$$

This choice of $\mathbf{A}$ and $\mathbf{B}$ also fulfills the orthogonality constraints of $\mathbf{U}$ and $\mathbf{V}$:

$$\mathbf{U} = \mathbf{A}^{\mathrm{T}}\mathbf{W}_s\mathbf{Z} = \mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{Q}^{\mathrm{T}}{\mathbf{W}_s}^{-1}\mathbf{W}_s\mathbf{Q}\mathbf{\Lambda}^{\frac{1}{2}} = \mathbf{I}; \tag{A.4}$$

$$\mathbf{V} = \mathbf{B}^{\mathrm{T}}\mathbf{W}_t\mathbf{Z} = \mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{Q}^{\mathrm{T}}{\mathbf{W}_t}^{-1}\mathbf{W}_t\mathbf{Q}\mathbf{\Lambda}^{\frac{1}{2}} = \mathbf{I}. \tag{A.5}$$

Thus $\rho_i = \mathrm{corr}(u_i, v_i) = 1$ for all $i = 1, \ldots, D_z$ and there exists no feature generator that could give higher correlations.

A.2 Morpho Challenge evaluation results

The tables of this section show part of the results calculated for the algorithms submitted to Morpho Challenge and used in the meta-evaluation of Publication VI. The full result tables are available from `http://research.ics.tkk.fi/events/morphochallenge/`.

Table A.1 shows all algorithms submitted to Morpho Challenges 2007–2010 and some statistics of their analysis for the English data set. The algorithms are categorized to three types: unsupervised (U), semi-supervised (S), and unsupervised with supervised parameter tuning (P). The difference between the last two is that the algorithms marked as semi-supervised used the provided annotated training data directly, not only for optimizing (few) free parameters for the evaluation metric in use.

Tables A.2–A.5 show the results of linguistic evaluation using the EMMA evaluation method by Spiegler and Monson (2010). Tables A.6–A.8 show the results of the information retrieval experiments, and Tables A.9 and A.10 show the results of the statistical machine translation experiments.

The application evaluations include some reference methods (Kurimo et al., 2010b). For IR, "dummy" means that no segmentation or analysis was performed and words were used as index terms as such. Hyphens were replaced by spaces so that hyphenated words were indexed as separate words. In "grammatical morphemes", the words were analyzed using the reference analyses of the linguistic evaluation. For ambiguous analyses, either only the first analysis was used (first), all of them were used (all), or the analysis with least number of morphemes was used (min). Words that were not in the gold standard were used as such. The IR experiments included also two language-specific methods: "Snowball" means that the words were stemmed by language specific stemming algorithms provided by Snowball libstemmer library 3. Hyphenated words were first split to parts and then stemmed separately. In "TWOL" the two-level morphological analyzer TWOL from Lingsoft was used to find the lemmas of the words. Some words may have several alternative interpretations and either all alternatives were used ("TWOL all") or only the first one ("TWOL first"). Compound words were split to parts. Words not recognized by the analyzer were indexed as such.

Morfessor Categories-MAP occur in the tables several times, because if was trained (on slightly different data sets) both by Monson et al. (2008, 2009) and the Challenge organizers. Algorithms developed in this thesis (Publications VIII, IX, and X) are highlighted.

Table A.1. Algorithms submitted to Morpho Challenges and their statistics for English. The type of the algorithm is either semi-supervised (S), unsupervised (U), or unsupervised with supervised parameter tuning (P). #a/w is the average number of analyses per word, #m/w is the the average number of morphemes per word, and #lexicon is the size of the morpheme lexicon.

Method	Author	Type	#a/w	#m/w	#lexicon
Allomorfessor 2008	Kohonen et al. (2009a)	U	1.00	1.62	180813
Allomorfessor 2009	Publication VIII	**U**	**1.00**	**2.59**	**23741**
Bernhard 1 2007	Bernhard (2008)	U	1.00	2.61	55490
Bernhard 2 2007	Bernhard (2008)	U	1.00	2.90	52582
Bordag 5 2007	Bordag (2008)	U	1.00	1.97	190094
Bordag 5a 2007	Bordag (2008)	U	1.00	1.97	189568
Can 2009	Can and Manandhar (2010)	U	1.00	2.09	150097
DEAP MDL-CAT 2010	Spiegler et al. (2010a)	S	4.20	3.41	418915
DEAP MDL-NOCAT 2010	Spiegler et al. (2010a)	S	1.71	3.42	216528
DEAP PROB-CAT 2010	Spiegler et al. (2010a)	S	4.63	2.79	790355
DEAP PROB-NOCAT 2010	Spiegler et al. (2010a)	S	1.69	2.91	301086
Lignos Aggressive Comp. 2010	Lignos (2010)	U	1.00	2.10	153688
Lignos Base Inference 2010	Lignos (2010)	U	1.00	1.92	168208
Lignos Iterative Comp. 2010	Lignos (2010)	U	1.00	1.91	171964
Lignos 2009	Lignos et al. (2010a)	U	1.00	1.74	198546
MAGIP 2010	Golénia et al. (2010a)	S	1.00	3.76	93086
McNamee 3 2007	McNamee (2008)	U	1.00	1.00	15212
McNamee 4 2007	McNamee (2008)	U	1.00	1.00	98475
McNamee 5 2007	McNamee (2008)	U	1.00	1.00	243578
MetaMorph 2009	Tchoukalov et al. (2010)	U	1.00	1.44	330990
Morfessor Baseline 2010	Creutz and Lagus (2005b)	U	1.00	2.17	61508
Morfessor Categories-MAP 2007	Monson et al. (2008)	U	1.00	2.07	137973
Morfessor Categories-MAP 2008	Monson et al. (2009)	U	1.00	2.07	137973
Morfessor Categories-MAP 2010	Creutz and Lagus (2007)	U	1.00	2.01	166078
Morfessor S+W 2010	Kohonen et al. (2010)	**S**	**1.00**	**2.65**	**32196**
Morfessor S+W+L 2010	Kohonen et al. (2010)	**S**	**1.00**	**2.59**	**45427**
Morfessor U+W 2010	Kohonen et al. (2010)	**P**	**1.00**	**2.80**	**14555**
MorphAcq 2010	Nicolas et al. (2010)	U	1.00	1.72	227131
MorphoNet 2009	Bernhard (2010b)	U	1.00	1.75	211439
ParaMor 2007	Monson et al. (2008)	U	2.42	1.88	233981
ParaMor 2008	Monson et al. (2009)	U	1.27	1.75	252997
ParaMor-Morfessor 2007	Monson et al. (2008)	U	3.42	1.93	386257
ParaMor-Morfessor 2008	Monson et al. (2009)	U	2.27	1.89	378364
ParaMor Mimic 2009	Monson et al. (2010)	P	1.00	3.04	188716
ParaMor-Morfessor Mimic 2009	Monson et al. (2010)	P	1.00	2.96	166310
ParaMor-Morfessor Union 2009	Monson et al. (2010)	P	1.00	2.87	120148
Promodes 2 2009	Spiegler et al. (2010b)	U	1.00	3.63	47456
Promodes 2009	Spiegler et al. (2010b)	U	1.00	3.28	107111
Promodes 2010	Spiegler et al. (2010c)	P	1.00	3.80	72811
Promodes committee 2009	Spiegler et al. (2010b)	U	1.00	3.63	47456
Promodes-E 2010	Spiegler et al. (2010c)	P	1.00	3.09	109050
Promodes-H 2010	Spiegler et al. (2010c)	P	1.00	4.39	45043
RALI-ANA 2009	Lavallée and Langlais (2010)	U	1.00	2.10	166826
RALI-COF 2009	Lavallée and Langlais (2010)	U	1.00	1.91	145733
RePortS 2007	Keshava and Pitler (2006)	U	1.00	1.57	211475
UNGRADE 2009	Golénia et al. (2010b)	U	1.00	3.87	123634
Zeman 1 2008	Zeman (2009)	U	3.18	1.74	905251
Zeman 2007	Zeman (2008)	U	3.18	1.74	905251
Zeman 3 2008	Zeman (2009)	U	1.08	1.37	319982

Table A.2. Precision, recall, and F-measure for the linguistic evaluation with EMMA for English.

Rank	Method	Type	Precision	Recall	F-measure
1	**Morfessor S+W+L 2010**	**S**	82.08	84.50	83.27
2	Lignos Base Inference 2010	U	87.44	76.82	81.78
3	Lignos 2009	U	92.40	73.33	81.77
4	Lignos Iterative Comp. 2010	U	87.36	76.41	81.52
5	Bernhard 1 2007	U	83.67	79.32	81.44
6	**Morfessor S+W 2010**	**S**	79.39	81.76	80.56
7	Bernhard 2 2007	U	79.09	80.93	80.00
8	Lignos Aggressive Comp. 2010	U	82.17	75.88	78.90
9	Morfessor Baseline 2010	U	86.86	72.26	78.89
10	Morfessor Categories-MAP 2010	U	92.09	68.15	78.33
11	MorphAcq 2010	U	78.51	77.59	78.04
12	Morfessor Categories-MAP 2007	U	86.55	70.62	77.78
13	Morfessor Categories-MAP 2008	U	86.55	70.62	77.78
14	**Allomorfessor 2009**	**U**	79.01	76.48	77.72
15	RALI-COF 2009	U	79.31	73.28	76.17
16	Bordag 5a 2007	U	82.78	70.11	75.91
17	Bordag 5 2007	U	82.72	70.09	75.88
18	MetaMorph 2009	U	84.48	68.05	75.38
19	ParaMor-Morfessor Mimic 2009	P	69.62	81.05	74.90
20	**Morfessor U+W 2010**	**P**	73.13	76.73	74.89
21	ParaMor-Morfessor Union 2009	P	69.62	80.95	74.86
22	ParaMor 2008	U	73.93	75.63	74.77
23	MorphoNet 2009	U	72.03	74.30	73.15
24	Allomorfessor 2008	U	87.22	62.38	72.74
25	ParaMor Mimic 2009	P	67.93	78.27	72.73
26	Can 2009	U	72.10	73.35	72.72
27	RALI-ANA 2009	U	76.94	67.92	72.15
28	Zeman 3 2008	U	88.39	58.28	70.24
29	RePortS 2007	U	76.83	64.13	69.90
30	McNamee 5 2007	U	98.81	51.92	68.07
31	Promodes-E 2010	P	65.73	69.77	67.69
32	Promodes 2009	U	61.18	75.61	67.64
33	McNamee 4 2007	U	97.73	51.27	67.26
34	Promodes 2010	P	56.58	73.82	64.06
35	DEAP MDL-NOCAT 2010	S	52.33	81.59	63.76
36	DEAP PROB-NOCAT 2010	S	56.18	73.14	63.55
37	McNamee 3 2007	U	87.05	45.77	59.99
38	MAGIP 2010	S	48.76	70.83	57.75
39	UNGRADE 2009	U	44.55	68.94	54.12
40	Promodes 2 2009	U	47.79	61.82	53.91
41	Promodes committee 2009	U	47.79	61.82	53.91
42	Zeman 2007	U	45.56	61.37	52.29
43	Zeman 1 2008	U	45.56	61.37	52.29
44	ParaMor-Morfessor 2008	U	40.08	68.24	50.40
45	Promodes-H 2010	P	40.16	62.98	49.04
46	ParaMor 2007	U	38.40	67.82	49.03
47	DEAP MDL-CAT 2010	S	25.79	85.03	39.57
48	ParaMor-Morfessor 2007	U	24.28	61.80	34.85
49	DEAP PROB-CAT 2010	S	22.29	77.30	34.60

Table A.3. Precision, recall, and F-measure for the linguistic evaluation with EMMA for Finnish.

Rank	Method	Type	Precision	Recall	F-measure
1	**Morfessor S+W+L 2010**	**S**	67.17	75.73	71.19
2	Lignos Base Inference 2010	U	79.90	56.05	65.88
3	Lignos Iterative Comp. 2010	U	77.33	56.92	65.57
4	RALI-COF 2009	U	74.61	55.96	63.94
5	Lignos Aggressive Comp. 2010	U	66.91	60.04	63.29
6	**Morfessor S+W 2010**	**S**	59.14	65.33	62.08
7	Morfessor Categories-MAP 2010	U	70.05	54.25	61.14
8	Bernhard 2 2007	U	61.74	60.51	61.11
9	Bernhard 1 2007	U	66.14	55.00	60.06
10	Morfessor Categories-MAP 2008	U	67.13	52.09	58.66
11	Morfessor Baseline 2010	U	76.14	47.42	58.44
12	Bordag 5a 2007	U	66.21	52.25	58.41
13	Bordag 5 2007	U	66.09	52.17	58.31
14	**Allomorfessor 2009**	**U**	71.69	49.05	58.25
15	ParaMor-Morfessor Mimic 2009	P	53.01	63.27	57.69
16	Allomorfessor 2008	U	79.78	43.87	56.61
17	MorphoNet 2009	U	59.53	53.27	56.22
18	ParaMor-Morfessor Union 2009	P	49.76	63.49	55.79
19	ParaMor 2008	U	56.24	54.95	55.58
20	ParaMor Mimic 2009	P	55.21	55.73	55.46
21	RALI-ANA 2009	U	69.56	45.24	54.82
22	Zeman 3 2008	U	82.19	41.10	54.80
23	**Morfessor U+W 2010**	**P**	53.58	55.71	54.62
24	Promodes 2010	P	44.93	59.47	51.18
25	MetaMorph 2009	U	69.59	40.27	51.02
26	McNamee 5 2007	U	99.77	33.97	50.68
27	McNamee 4 2007	U	98.41	33.51	50.00
28	Promodes 2009	U	43.37	58.69	49.88
29	Promodes-E 2010	P	44.23	55.55	49.25
30	MAGIP 2010	S	43.49	53.64	48.03
31	Promodes committee 2009	U	42.45	51.87	46.69
32	DEAP PROB-NOCAT 2010	S	39.36	56.22	46.29
33	UNGRADE 2009	U	41.48	51.64	46.00
34	Promodes 2 2009	U	37.91	53.17	44.26
35	McNamee 3 2007	U	86.21	29.45	43.90
36	Zeman 1 2008	U	45.24	41.57	43.32
37	Zeman 2007	U	44.52	41.51	42.96
38	Promodes-H 2010	P	35.89	52.73	42.71
39	DEAP MDL-NOCAT 2010	S	29.60	66.45	40.95
40	ParaMor-Morfessor 2008	U	30.56	48.71	37.53
41	DEAP MDL-CAT 2010	S	21.01	64.26	31.66
42	DEAP PROB-CAT 2010	S	21.06	50.95	29.79

Table A.4. Precision, recall, and F-measure for the linguistic evaluation with EMMA for German.

Rank	Method	Type	Precision	Recall	F-measure
1	Morfessor Categories-MAP 2007	U	76.51	57.00	65.33
2	Morfessor Categories-MAP 2008	U	76.51	57.00	65.33
3	Morfessor Categories-MAP 2010	U	78.26	55.83	65.17
4	Bordag 5a 2007	U	71.96	57.34	63.82
5	Bordag 5 2007	U	71.80	57.32	63.74
6	Bernhard 1 2007	U	71.64	54.68	62.02
7	**Allomorfessor 2009**	**U**	77.24	51.57	61.84
8	ParaMor-Morfessor Union 2009	P	62.75	60.30	61.50
9	ParaMor-Morfessor Mimic 2009	P	62.36	60.20	61.26
10	MorphAcq 2010	U	76.11	51.26	61.25
11	Bernhard 2 2007	U	64.27	57.65	60.78
12	Lignos Base Inference 2010	U	74.98	50.84	60.59
13	RALI-COF 2009	U	73.22	51.57	60.51
14	Morfessor Baseline 2010	U	81.28	48.06	60.40
15	MorphoNet 2009	U	71.95	51.93	60.32
16	**Morfessor U+W 2010**	**P**	65.71	55.60	60.23
17	Lignos 2009	U	85.88	45.45	59.44
18	Lignos Iterative Comp. 2010	U	71.55	50.35	59.10
19	Can 2 2009	U	63.56	55.01	58.97
20	Lignos Aggressive Comp. 2010	U	68.75	51.17	58.67
21	ParaMor 2008	U	68.34	49.69	57.54
22	Allomorfessor 2008	U	84.89	43.12	57.19
23	Promodes 2009	U	63.88	51.48	57.01
24	ParaMor Mimic 2009	P	65.21	50.24	56.75
25	Can 1 2009	U	71.92	46.68	56.61
26	RALI-ANA 2009	U	72.12	43.84	54.52
27	Promodes committee 2009	U	58.88	49.06	53.51
28	Zeman 3 2008	U	81.63	39.67	53.39
29	MetaMorph 2009	U	66.38	42.17	51.58
30	McNamee 5 2007	U	97.62	33.92	50.34
31	McNamee 4 2007	U	96.78	33.61	49.89
32	Promodes 2 2009	U	48.67	50.12	49.38
33	UNGRADE 2009	U	45.58	49.98	47.68
34	ParaMor 2007	U	48.24	46.60	47.40
35	McNamee 3 2007	U	87.60	30.39	45.12
36	ParaMor-Morfessor 2008	U	38.93	53.17	44.92
37	Zeman 2007	U	45.46	41.51	43.39
38	Zeman 1 2008	U	45.31	41.52	43.32
39	ParaMor-Morfessor 2007	U	28.68	49.59	36.34

Table A.5. Precision, recall, and F-measure for the linguistic evaluation with EMMA for Turkish.

Rank	Method	Type	Precision	Recall	F-measure
1	**Morfessor S+W+L 2010**	**S**	75.32	57.79	65.39
2	**Morfessor S+W 2010**	**S**	57.34	44.25	49.94
3	Morfessor Categories-MAP 2010	U	64.68	40.04	49.46
4	Morfessor Categories-MAP 2008	U	63.07	40.36	49.22
5	ParaMor 2008	U	54.04	43.55	48.23
6	ParaMor-Morfessor Mimic 2009	P	49.19	45.84	47.45
7	Lignos Base Inference 2010	U	71.31	35.30	47.22
8	ParaMor Mimic 2009	P	49.87	44.86	47.22
9	MorphAcq 2010	U	63.10	37.44	46.99
10	ParaMor-Morfessor Union 2009	P	48.19	45.21	46.65
11	Lignos Iterative Comp. 2010	U	66.07	36.02	46.62
12	Bernhard 2 2007	U	63.02	36.97	46.60
13	Bordag 5 2007	U	63.87	36.53	46.47
14	Bordag 5a 2007	U	63.66	36.40	46.31
15	RALI-ANA 2009	U	64.35	35.08	45.41
16	Lignos Aggressive Comp. 2010	U	57.09	37.33	45.13
17	Morfessor Baseline 2010	U	65.43	34.30	45.00
18	Bernhard 1 2007	U	63.16	34.76	44.84
19	Allomorfessor 2008	U	71.39	32.69	44.84
20	**Allomorfessor 2009**	**U**	63.07	34.63	44.71
21	RALI-COF 2009	U	46.93	42.50	44.61
22	Can 1 2009	U	70.05	32.36	44.27
23	Zeman 3 2008	U	76.95	30.72	43.91
24	MorphoNet 2009	U	45.46	42.46	43.90
25	DEAP MDL-NOCAT 2010	S	35.90	55.54	43.60
26	McNamee 5 2007	U	95.93	27.66	42.93
27	McNamee 4 2007	U	95.96	27.63	42.91
28	Promodes 2010	P	44.71	41.17	42.86
29	Promodes committee 2009	U	52.48	35.45	42.31
30	Zeman 2007	U	46.68	37.92	41.84
31	Zeman 1 2008	U	46.68	37.92	41.84
32	Can 2 2009	U	50.37	35.03	41.32
33	ParaMor-Morfessor 2008	U	36.23	47.06	40.93
34	McNamee 3 2007	U	88.91	25.77	39.95
35	DEAP PROB-NOCAT 2010	S	32.03	52.23	39.71
36	UNGRADE 2009	U	41.38	37.01	39.07
37	MetaMorph 2009	U	54.86	30.28	39.01
38	Promodes-H 2010	P	41.23	35.81	38.33
39	Promodes-E 2010	P	37.22	36.64	36.93
40	Promodes 2009	U	34.66	37.91	36.21
41	MAGIP 2010	S	34.79	35.41	35.09
42	Promodes 2 2009	U	33.96	33.74	33.84
43	DEAP MDL-CAT 2010	S	23.90	55.64	33.44
44	DEAP PROB-CAT 2010	S	24.17	52.06	33.01
45	**Morfessor U+W 2010**	**P**	29.59	26.27	27.83

Table A.6. The mean average precision (MAP) in the information retrieval task for English.

Rank	Method	Type	MAP
1	Snowball Porter	-	40.92
2	ParaMor-Morfessor 2008	U	40.43
3	TWOL first	-	40.20
4	TWOL all	-	39.94
5	ParaMor 2008	U	39.42
6	Bernhard 2 2007	U	39.39
7	Bernhard 1 2007	U	38.88
8	ParaMor-Morfessor Union 2009	P	38.79
9	Morfessor Baseline 2010	U	38.35
10	Lignos Base Inference 2010	U	38.32
11	**Allomorfessor 2009**	**U**	38.31
12	ParaMor Mimic 2009	P	38.21
13	**Morfessor S+W+L 2010**	**S**	38.20
14	Lignos Aggressive Comp. 2010	U	37.84
15	Grammatical morphemes (first)	-	37.77
16	**Morfessor S+W 2010**	**S**	37.76
17	Lignos Iterative Comp. 2010	U	37.76
18	Morfessor Categories-MAP 2010	U	37.54
19	Morfessor Categories-MAP 2007	U	37.50
20	ParaMor-Morfessor Mimic 2009	P	36.90
21	**Morfessor U+W 2010**	**P**	36.90
22	Morfessor Categories-MAP 2008	U	36.80
23	MorphAcq 2010	U	36.80
24	RePortS 2007	U	36.03
25	McNamee 4 2007	U	35.72
26	Grammatical morphemes (all)	-	35.67
27	MorphoNet 2009	U	35.16
28	McNamee 5 2007	U	34.32
29	Bordag 5a 2007	U	34.20
30	Bordag 5 2007	U	34.11
31	Promodes-E 2010	P	33.79
32	dummy	-	33.04
33	Promodes 2010	P	32.86
34	DEAP PROB-NOCAT 2010	S	31.41
35	McNamee 3 2007	U	30.35
36	DEAP MDL-NOCAT 2010	S	29.16
37	ParaMor 2007	U	28.19
38	ParaMor-Morfessor 2007	U	26.93
39	DEAP MDL-CAT 2010	S	24.09
40	MAGIP 2010	S	22.57
41	DEAP PROB-CAT 2010	S	21.86
42	Promodes-H 2010	P	20.34

Table A.7. The mean average precision (MAP) in the information retrieval task for Finnish.

Rank	Method	Type	MAP
1	TWOL first	-	49.73
2	Lignos Aggressive Comp. 2010	U	49.14
3	Bernhard 2 2007	U	49.07
4	TWOL all	-	48.01
5	Bernhard 1 2007	U	47.99
6	Morfessor Categories-MAP 2010	U	47.54
7	**Morfessor S+W 2010**	**S**	47.50
8	ParaMor-Morfessor Union 2009	P	47.13
9	Morfessor Categories-MAP 2008	U	46.74
10	**Allomorfessor 2009**	**U**	45.68
11	ParaMor-Morfessor 2008	U	45.42
12	**Morfessor S+W+L 2010**	**S**	44.65
13	ParaMor-Morfessor Mimic 2009	P	44.46
14	Grammatical morphemes (first)	-	43.17
15	Bordag 5 2007	U	43.10
16	Bordag 5a 2007	U	42.83
17	Snowball Finnish	-	42.75
18	Morfessor Baseline 2010	U	42.35
19	Lignos Iterative Comp. 2010	U	42.29
20	DEAP MDL-NOCAT 2010	S	41.60
21	Lignos Base Inference 2010	U	41.51
22	Grammatical morphemes (all)	-	40.93
23	**Morfessor U+W 2010**	**P**	40.42
24	ParaMor Mimic 2009	P	39.05
25	MorphoNet 2009	U	38.75
26	MAGIP 2010	S	38.71
27	ParaMor 2008	U	38.28
28	DEAP MDL-CAT 2010	S	37.25
29	Promodes 2010	P	37.21
30	McNamee 5 2007	U	37.00
31	McNamee 4 2007	U	36.04
32	dummy	-	35.29
33	McNamee 3 2007	U	32.48
34	Promodes-E 2010	P	32.13
35	DEAP PROB-NOCAT 2010	S	31.74
36	Promodes-H 2010	P	31.09
37	DEAP PROB-CAT 2010	S	28.43

Table A.8. The mean average precision (MAP) in the information retrieval task for German.

Rank	Method	Type	MAP
1	TWOL first	-	48.36
2	TWOL all	-	47.45
3	ParaMor-Morfessor 2008	U	46.90
4	**Morfessor U+W 2010**	**P**	46.66
5	Morfessor Categories-MAP 2010	U	46.57
6	Morfessor Categories-MAP 2008	U	46.50
7	Bernhard 1 2007	U	45.01
8	ParaMor-Morfessor Mimic 2009	P	44.97
9	Bernhard 2 2007	U	44.93
10	Lignos Base Inference 2010	U	44.34
11	Morfessor Categories-MAP 2007	U	44.25
12	ParaMor-Morfessor Union 2009	P	44.02
13	**Allomorfessor 2009**	**U**	43.93
14	Morfessor Baseline 2010	U	43.91
15	Lignos Aggressive Comp. 2010	U	43.79
16	Lignos Iterative Comp. 2010	U	42.00
17	Bordag 5 2007	U	41.87
18	Bordag 5a 2007	U	41.47
19	ParaMor-Morfessor 2007	U	38.98
20	MorphAcq 2010	U	38.62
21	Snowball German	-	38.59
22	ParaMor Mimic 2009	P	37.45
23	ParaMor 2008	U	36.24
24	dummy	-	35.08
25	McNamee 5 2007	U	33.97
26	Grammatical morphemes (first)	-	33.50
27	MorphoNet 2009	U	33.12
28	McNamee 4 2007	U	32.60
29	ParaMor 2007	U	31.42
30	Grammatical morphemes (all)	-	30.08
31	McNamee 3 2007	U	27.82

Table A.9. BLEU scores of the minimum Bayes risk combination for the Finnish-to-English machine translation task.

Rank	Method	Type	BLEU
1	**Allomorfessor 2009**	**U**	26.80
2	Morfessor Baseline 2010	U	26.65
3	Morfessor Categories-MAP 2010	U	26.34
4	Grammatical morphemes (min)	-	26.23
5	Lignos Base Inference 2010	U	26.19
6	Lignos Iterative Comp. 2010	U	26.05
7	Lignos Aggressive Comp. 2010	U	25.97
8	**Morfessor S+W 2010**	**S**	25.94
9	DEAP MDL-CAT 2010	S	25.90
10	DEAP PROB-CAT 2010	S	25.89
11	**Morfessor S+W+L 2010**	**S**	25.82
12	Promodes-E 2010	P	25.79
13	MetaMorph 2009	U	25.72
14	Promodes 2010	P	25.64
15	DEAP MDL-NOCAT 2010	S	25.62
16	**Morfessor U+W 2010**	**P**	25.61
17	MorphoNet 2009	U	25.56
18	DEAP PROB-NOCAT 2010	S	25.55
19	ParaMor-Morfessor Mimic 2009	P	25.54
20	ParaMor Mimic 2009	P	25.53
21	ParaMor-Morfessor Union 2009	P	25.44
22	Promodes-H 2010	P	25.35
23	MAGIP 2010	S	25.33
24	words	-	25.28

Table A.10. BLEU scores of the minimum Bayes risk combination for the German-to-English machine translation task.

Rank	Method	Type	BLEU
1	**Allomorfessor 2009**	**U**	30.09
2	Morfessor Categories-MAP 2010	U	30.08
3	Morfessor Baseline 2010	U	29.94
4	**Morfessor U+W 2010**	**P**	29.86
5	MorphAcq 2010	U	29.86
6	Lignos Iterative Comp. 2010	U	29.85
7	Lignos Aggressive Comp. 2010	U	29.80
8	ParaMor Mimic 2009	P	29.77
9	Grammatical morphemes (min)	-	29.70
10	ParaMor-Morfessor Union 2009	P	29.64
11	Lignos Base Inference 2010	U	29.52
12	ParaMor-Morfessor Mimic 2009	P	29.48
13	MorphoNet 2009	U	29.46
14	MetaMorph 2009	U	29.44
15	words	-	29.43

A.3 Examples of transformations extracted by Allomorfessor

Tables A.11 and A.12 show the analyses selected by Allomorfessor in the experiments of Publication VIII for English and Finnish data sets. The examples include both linguistically sensible and erroneous analyses. If there are no comments, the use of transformation is considered very successful. For example, in English, the transformation `(-e)` is applied correctly over almost all of its occurrences.

Table A.11. The 15 most frequent transformation extracted by Allomorfessor for English.

Count	Transf.	Example analyses		Comment
1182	`(-e)`	adhering	`adhere (-e) ing`	
300	`(-y)`	vulnerabilities	`vulnerability (-y) ies`	
		temporarily	`temporary (-y) ily`	
		technologist	`technology (-y) ist`	
120	`(-t)`	affluence	`affluent (-t) ce`	
		bankrupcy	`bankrupt (-t) cy`	misspelled
66	`(-a)`	encyclopedic	`encyclopedia (-a) c`	
		diplomtic	`diplomat (-a) ic`	misspelled
		hemophilic	`hemophilia (-a) c`	
41	`(-i)`	publshed	`publish (-i) ed`	misspelled
35	`(-s)`	euripidean	`euripides (-s) a () n`	
		diocletian	`diocles (-s) tian`	
27	`(-o)`	aspirating	`aspiration (-o) g`	`-ing` not segmented
27	`(-n)`	proletariat	`proletarian (-n) t`	
		restauration	`restaurant (-n) ion`	wrong lemma
		superstitious	`superstition (-n) us`	
20	`(-c)`	paraplegia	`paraplegic (-c) a`	
8	`(t|c)`	excellencies	`excellent (t|c) ies`	adjective as lemma
		inconveniencies	`in () convenient (t|c) ies`	adjective as lemma
2	`(a|s)`	ljubljanska	`ljubljana (a|s) ka`	proper name
1	`(-g)`	licensintorg	`licensing (-g) torg`	proper name
1	`(s|n)`	sclerosing	`sclerosis (s|n) g`	`-ing` not segmented
1	`(-h)`	thorougbred	`thorough (-h) bred`	misspelled
1	`(-a -y)`	bulathkopitiya	`bulathkopitya (-a -y) iya`	proper name

Table A.12. The 22 most frequent transformations extracted by Allomorfessor for Finnish.

Count	Transf.	Example analyses		Comment
7771	(-n)	ahdingolla	ahdingon (-n) lla	genitive as lemma
		aikojemme	aikojen (-n) mme	plural genitive as lemma
4096	(-i)	anakronismeille	anakronismi (-i) e () ille	(i|e) preferable
		desibelejä	desibeli (-i) ejä	(i|e) preferable
2598	(-a)	diakonissoja	diakonissa (-a) oja	(a|o) preferable
		eufemismi	eufemia (-a) smi	
2507	(-t)	fagotisti	fagotti (-t) sti	
		haltuunoton	haltuunotto (-t) n	
1114	(-s)	harvennuksen	harvennus (-s) ksen	
		yliherkkyydet	yliherkkyys (-s) det	
939	(-e)	vuosituhantista	vuosituhantiset (-e) a	plural as lemma
		viikattein	viikate (-e) tein	
675	(i|e)	videoprojektoreina	video () projektori (i|e) ina	
		transistoreita	transistori (i|e) ita	
532	(-ä)	tulennielijöitä	tulennielijä (-ä) öitä	
		tulokertymien	tulokertymä (-ä) ien	
430	(a|i)	kaavailemia	kaavailema (a|i) a	
		juurevia	juureva (a|i) a	
428	(n|s)	hankkeeseesi	hankkeeseen (n|s) i	undersegmented
		diabeteksesi	diabeteksen (n|s) i	undersegmented
322	(a|e)	emigranttien	emigranttia (a|e) n	partitive as lemma
		hajuharhojen	haju () harhoja (a|e) n	plural partitive as lemma
311	(-k)	agnostikoksi	agnostikko (-k) ksi	
		haaksirikossa	haaksirikko (-k) ssa	
232	(-a -t)	murhissa	murhista (-a -t) sa	elative as lemma
		varainhankinnalla	varainhankinta (-a -t) na () lla	(t|n) preferable
183	(-n -i)	barrikadeja	barrikadin (-n -i) eja	genitive as lemma
		kursseihen	kursseihin (-n -i) en	misspelled
143	(n|i -e)	aivotärähdyksiä	aivo () tärähdyksen (n|i -e) ä	genitive as lemma
		hoplofoobisia	hoplofoobisen (n|i -e) a	genitive as lemma
138	(-n n|s)	aivokurkiaisen	aivokurkiainen (-n n|s) n	
		mustapukuiset	mustapukuinen (-n n|s) t	
97	(t|d)	häädöt	häätö (t|d) t	
		kursivoidun	kursivoitu (t|d) n	
83	(a|s -t)	amppeleissa	amppeleita (a|s -t) sa	plural partitive as lemma
		elintarvikkeissa	elintarvikkeita (a|s -t) sa	plural partitive as lemma
82	(ä|t -l)	näöltään	näöllä (ä|t -l) ään	adessive as lemma
		piirtoheittimeltä	piirtoheittimellä (ä|t -l) ä	adessive as lemma
77	(-e -s)	esoteerinen	esoteerisen (-e -s) en	genitive as lemma
		teksasilainen	teksasilaisen (-e -s) en	
75	(t|n)	abstrahoinnin	abstrahointi (t|n) n	
		vektorikvantisoinnin	vektori () kvantisointi (t|n) n	
75	(a|t -l)	matkapuhelimeltaan	matka () puhelimella (a|t -l) aan	adessive as lemma
		pankkiautomaateilta	pankki () automaateilla (a|t -l) a	adessive as lemma

Bibliography

Abney, S. P. (1991). Parsing by chunks. In Berwick, R., Abney, S., and Tenny, C., editors, *Principle-Based Parsing*, pages 257–278. Kluwer Academic Publishers, Dordrecht, The Netherlands.

Achlioptas, D. (2001). Database-friendly random projections. In Aref, W. G., editor, *Proceedings of the 20th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, pages 274–281, Santa Barbara, California, USA. ACM.

Adriaans, P., Trautwein, M., and Vervoort, M. (2000). Towards high speed grammar induction on large text corpora. In *SOFSEM 2000: Theory and practice of Informatics*, volume 1963 of *Lecture Notes in Computer Science*, pages 173–186. Springer Verlag, Berlin, Germany.

Ailon, N. and Chazelle, B. (2006). Approximate nearest neighbors and the fast Johnson-Lindenstrauss transform. In Kleinberg, J. M., editor, *The 38th ACM Symposium on Theory of Computing (STOC 2006)*, pages 557–563, Seattle, WA, USA. ACM.

Akaho, S. (2001). A kernel method for canonical correlation analysis. In Yanai, H., Okada, A., Shigemasu, K., Kano, Y., and Meulman, J. J., editors, *Proceedings of the International Meeting of the Psychometric Society (IMPS2001)*, Osaka, Japan. Springer-Verlag.

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723.

Albright, A. and Hayes, B. (2003). Rules vs. analogy in English past tenses: A computational/experimental study. *Cognition*, 90:119–161.

Alegre, M. and Gordon, P. (1999). Frequency effects and the representational status of regular inflections. *Journal of Memory and Language*, 40:41–61.

Alexandrescu, A. and Kirchhoff, K. (2006). Factored neural language models. In Moore, R. C., Bilmes, J., Chu-Carroll, J., and Sanderson, M., editors, *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 1–4, New York, NY, USA. Association for Computational Linguistics.

Alkula, R. (2001). From plain character strings to meaningful words: Producing better full text databases for inflectional and compounding languages with morphological analysis software. *Information Retrieval*, 4(3):195–208.

Alpaydin, E. (2004). *Introduction to Machine Learning*. The MIT Press, Cambridge, MA, USA.

Alumäe, T. and Kurimo, M. (2010). Domain adaptation of maximum entropy language models. In Hajič, J., Carberry, S., Clark, S., and Nivre, J., editors, *Proceedings of the ACL 2010 Conference Short Papers*, pages 301–306, Uppsala, Sweden. Association for Computational Linguistics.

Alumäe, T. and Kurimo, M. (2010). Efficient estimation of maximum entropy language models with n-gram features: an SRILM extension. In Kobayashi, T., Hirose, K., and Nakamura, S., editors, *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010)*, pages 1820–1823, Makuhari, Chiba, Japan. ISCA.

Arthur, D. and Vassilvitskii, S. (2007). K-means++: The advantages of careful seeding. In Bansal, N., Pruhs, K., and Stein, C., editors, *SODA '07: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1027–1035, New Orleans, LA, USA. Society for Industrial and Applied Mathematics.

Baayen, R. H. (2007). Storage and computation in the mental lexicon. In Jarema, G. and Libben, G., editors, *The Mental Lexicon: Core Perspectives*, pages 81–104. Elsevier, Oxford, UK.

Baayen, R. H., Dijkstra, T., and Schreuder, R. (1997). Singulars and plurals in Dutch: Evidence for a parallel dual route model. *Journal of Memory and Language*, 37:94–117.

Baayen, R. H., Milin, P., Filipović Đurđević, D., Hendrix, P., and Marelli, M. (2011). An amorphous model for morphological processing in visual comprehension based on naive discriminative learning. *Psychological Review*, 118:438–482.

Bacchiani, M. and Roark, B. (2003). Unsupervised language model adaptation. In *Proceedings of the 2003 IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, pages 224–227, Hong Kong, China. IEEE.

Bach, F. R. and Jordan, M. I. (2003). Kernel independent component analysis. *Journal of Machine Learning Research*, 3:1–48.

Bagga, A. and Baldwin, B. (1998). Entity-based cross-document coreferencing using the vector space model. In *Proceedings of the 17th International Conference on Computational Linguistics (COLING'98)*, volume 1, pages 79–85, Montreal, Canada. Association for Computational Linguistics / Morgan Kaufmann Publishers.

Bahl, L. R., Brown, P. F., de Souza, P. V., and Mercer, R. L. (1989). A tree-based statistical language model for natural language speech recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37(7):1001–1008.

Baroni, M., Matiasek, J., and Trost, H. (2002). Unsupervised discovery of morphologically related words based on orthographic and semantic similarity. In *Proceedings of the ACL-02 Workshop on Morphological and Phonological Learning*, pages 48–57, Philadelphia, PA, USA. Association for Computational Linguistics.

Barron, A., Rissanen, J., and Yu, B. (1998). The minimum description length principle in coding and modeling. *IEEE Transactions on Information Theory*, 44(6):2743–2760.

Baum, L. E. (1972). An inequality and an associated maximization technique in statistical estimation of probabilistic functions of a Markov process. *Inequalities*, 3(1):1–8.

Beal, M. J., Ghahramani, Z., and Rasmussen, C. E. (2002). The infinite hidden Markov model. In *Advances in Neural Information Processing Systems 14*, pages 577–584, Cambridge, MA, USA. MIT Press.

Beesley, K. (1988). Language identifier: A computer program for automatic natural-language identification of on-line text. In Hammond, D. L., editor, *Proceedings of the 29th Annual Conference of the American Translators Association*, pages 47–54, Seattle, WA, USA. Information Today, Inc.

Bellegarda, J. R. (2000). Exploiting latent semantic information in statistical language modeling. *Proceedings of the IEEE*, 88(8):1279–1296.

Bengio, Y., Ducharme, R., and Vincent, P. (2001). A neural probabilistic language model. In Leen, T. K., Dietterich, T. G., and Tresp, V., editors, *Advances in Neural Information Processing Systems 13*, pages 932–938. MIT Press, Cambridge, MA, USA.

Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Benzécri, J.-P. (1973). *L'Analyse des Données. Volume II. L'Analyse des Correspondances*. Dunod, Paris, France.

Berant, J., Gross, Y., Mussel, M., Sandbank, B., Ruppin, E., and Edelman, S. (2007). Boosting unsupervised grammar induction by splitting complex sentences on function words. In Caunt-Nulton, H., Kulatilake, S., and hao Woo, I., editors, *Proceedings of the 31st Annual Boston University Conference on Language Development (BUCLD)*, pages 93–104, Boston, MA, USA. Cascadilla Press.

Bernhard, D. (2006). Unsupervised morphological segmentation based on segment predictability and word segments alignment. In Kurimo, M., Creutz, M., and Lagus, K., editors, *Proceedings of the PASCAL Challenge Workshop on Unsupervised segmentation of words into morphemes*, pages 18–22, Venice, Italy. PASCAL European Network of Excellence.

Bernhard, D. (2008). Simple morpheme labelling in unsupervised morpheme analysis. In *Advances in Multilingual and Multimodal Information Retrieval, 8th Workshop of the Cross-Language Evaluation Forum, CLEF 2007, Budapest, Hungary, September 19–21, 2007, Revised Selected Papers*, volume 5152 of *Lecture Notes in Computer Science*, pages 873–880. Springer, Berlin / Heidelberg, Germany.

Bernhard, D. (2010a). Apprentissage non supervisé de familles morphologiques : comparaison de méthodes et aspects multilingues. *Traitement Automatique des Langues*, 52(2):11–39.

Bernhard, D. (2010b). Morphonet: Exploring the use of community structure for unsupervised morpheme analysis. In *Multilingual Information Access Evaluation I. Text Retrieval Experiments: 10th Workshop of the Cross-Language Evaluation Forum, CLEF 2009, Corfu, Greece, September 30 – October 2, 2009, Revised Selected Papers*, volume 6241 of *Lecture Notes in Computer Science*, pages 598–608. Springer, Berlin / Heidelberg, Germany.

Bertram, R., Baayen, R. H., and Schreuder, R. (2000). Effects of family size for complex words. *Journal of Memory and Language*, 42:390–405.

Besançon, R. and Rajman, M. (2002). Evaluation of a vector space similarity measure in a multilingual framework. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC 2002)*, volume 1252, pages 1537–1542, Las Palmas, Spain. European Language Resources Association.

Bharati, A., Sangal, R., Bendre, S., Kumar, P., and Aishwarya (2001). Unsupervised improvement of morphological analyzer for inflectionally rich languages. In *Proceedings of the Sixth Natural Language Processing Pacific Rim Symposium*, pages 685–692, Tokyo, Japan. Asian Federation of Natural Language Processing.

Bilmes, J. A. (1998). A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. Technical Report TR-97-021, International Compute Science Institute, Berkeley, CA, USA.

Bilmes, J. A. and Kirchhoff, K. (2003). Factored language models and generalized parallel backoff. In *Companion Volume of the Proceedings of HLT-NAACL 2003 — Short Papers*, pages 4–6, Edmonton, Canada. Association for Computational Linguistics.

Bingham, E. and Mannila, H. (2001). Random projection in dimensionality reduction: applications to image and text data. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2001)*, pages 245–250, San Francisco, CA, USA. ACM.

Bird, S. (2009). Natural language processing and linguistic fieldwork. *Computational Linguistics*, 35(3):469–474.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer Science+Business Media, New York, NY, USA.

Blanken, H. and Hiemstra, D. (2007). Searching for text documents. In Blanken, H. M., de Vries, A. P., Blok, H. E., and Feng, L., editors, *Multimedia retrieval*, pages 97–124. Springer-Verlag, Berlin / Heidelberg, Germany.

Blasig, R. (1999). Combination of words and word categories in varigram histories. In *Proceedings of the 1999 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1, pages 529–532, Phoenix, AZ, USA. IEEE.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Blitzer, J., Globerson, A., and Pereira, F. (2005). Distributed latent variable models of lexical co-occurrences. In Cowell, R. G. and Ghahramani, Z., editors, *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics (AISTATS 2005)*, pages 25–32, Barbados. Society for Artificial Intelligence and Statistics.

Bloomfield, L. (1935). *Language*. George Allen & Unwin Ltd., London, UK.

Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In Bartlett, P. L. and Mansour, Y., editors, *Proceedings of the Eleventh Annual Conference on Computational Learning Theory (COLT 1998)*, pages 92–100, Madison, WI, USA. Morgan Kaufmann.

Bod, R. (2006). An all-subtrees approach to unsupervised parsing. In Calzolari, N., Cardie, C., and Isabelle, P., editors, *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 865–872, Sydney, Australia. Association for Computational Linguistics.

Bod, R. (2007). Is the end of supervised parsing in sight? In Zaenen, A. and van den Bosch, A., editors, *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 400–407, Prague, Czech Republic. Association for Computational Linguistics.

Bod, R. (2009a). Constructions at work or at rest? *Cognitive Linguistics*, 20(1):129–134.

Bod, R. (2009b). From exemplar to grammar: A probabilistic analogy-based model of language learning. *Cognitive Science*, 33(5):752–793.

Bonafonte, A. and Mariño, J. B. (1996). Language modeling using x-grams. In Bunnel, H. T. and Idsardi, W., editors, *Proceedings of Fourth International Conference Spoken Language Processing (ICSLP)*, pages 394–397, Philadelphia, PA, USA. ISCA.

Bookstein, A. and Swanson, D. R. (1974). Probabilistic models for automatic indexing. *Journal of the American Society for Information Science*, 25(5):312–316.

Bordag, S. (2006). Two-step approach to unsupervised morpheme segmentation. In Kurimo, M., Creutz, M., and Lagus, K., editors, *Proceedings of the PASCAL Challenge Workshop on Unsupervised segmentation of words into morphemes*, pages 23–27, Venice, Italy. PASCAL European Network of Excellence.

Bordag, S. (2008). Unsupervised and knowledge-free morpheme segmentation and analysis. In *Advances in Multilingual and Multimodal Information Retrieval, 8th Workshop of the Cross-Language Evaluation Forum, CLEF 2007, Budapest, Hungary, September 19–21, 2007, Revised Selected Papers*, volume 5152 of *Lecture Notes in Computer Science*, pages 881–891. Springer, Berlin / Heidelberg, Germany.

Borensztajn, G., Zuidema, W., and Bod, R. (2009). Children's grammars grow more abstract with age. *Topics in Cognitive Science*, 1:175–188.

Borga, M. (1998). *Learning Multidimensional Signal Processing*. PhD thesis, Linköping University, Sweden.

Branavana, S., Chen, H., Zettlemoyer, S., and Barzilay, R. (2009). Reinforcement learning for mapping instructions to actions. In Su, K.-Y., Su, J., Wiebe, J., and Li, H., editors, *Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP*, pages 82–90, Suntec, Singapore. Association for Computational Linguistics.

Brants, T. and Franz, A. (2006). *Web 1T 5-gram Version 1*. Linguistic Data Consortium, Philadelphia, PA, USA.

Brants, T., Popat, A. C., Xu, P., Och, F. J., and Dean, J. (2007). Large language models in machine translation. In Eisner, J., editor, *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 858–867, Prague, Czech Republic. Association for Computational Linguistics.

Brent, M. (1999). An efficient, probabilistically sound algorithm for segmentation and word discovery. *Machine Learning*, 34:71–105.

Broman, S. and Kurimo, M. (2005). Methods for combining language models in speech recognition. In *Proceedings of the 9th European Conference on Speech Communication and Technology (Interspeech 2005)*, pages 1317–1320, Lisbon, Portugal. ISCA.

Brown, P. F., DellaPietra, V. J., deSouza, P. V., Lai, J. C., and Mercer, R. L. (1992). Class-based *n*-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

Brown, P. F., Pietra, V. J. D., Pietra, S. A. D., and Mercer, R. L. (1993). The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Callison-Burch, C., Koehn, P., Monz, C., and Zaidan, O. (2011). Findings of the 2011 workshop on statistical machine translation. In Callison-Burch, C., Koehn, P., Monz, C., and Zaidan, O. F., editors, *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 22–64, Edinburgh, Scotland. Association for Computational Linguistics.

Callison-Burch, C., Osborne, M., and Koehn, P. (2006). Re-evaluating the role of BLEU in machine translation research. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2006)*, pages 249–256, Trento, Italy. Association for Computational Linguistics.

Can, B. and Manandhar, S. (2010). Clustering morphological paradigms using syntactic categories. In *Multilingual Information Access Evaluation I. Text Retrieval Experiments: 10th Workshop of the Cross-Language Evaluation Forum, CLEF 2009, Corfu, Greece, September 30 – October 2, 2009, Revised Selected Papers*, volume 6241 of *Lecture Notes in Computer Science*, pages 641–648. Springer, Berlin / Heidelberg, Germany.

Candès, E. J., Romberg, J., and Tao, T. (2006). Robust uncertainty principles: Exact signal reconstruction from highly incomplete Fourier information. *IEEE Transactions on Information Theory*, 52(2):489–509.

Candès, E. J. and Wakin, M. B. (2008). An introduction to compressive sampling. *IEEE Signal Processing Magazine*, 21(2):21–30.

Cardoso, J.-F. (1998). Blind signal separation: statistical principles. *Proceedings of the IEEE*, 86(10):2009–2025.

Caropreso, M. F., Matwin, S., and Sebastiani, F. (2001). A learner-independent evaluation of the usefulness of statistical phrases for automated text categorization. In Chin, A. G., editor, *Text Databases & Document Management: Theory & Practice*, pages 78–102. IGI Publishing, Hershey, PA, USA.

Caruana, R. (1997). Multitask learning. *Machine Learning*, 28:41–75.

Chaitin, G. J. (1969). On the length of programs for computing finite binary sequences: statistical considerations. *Journal of the ACM*, 16:145–159.

Chan, E. (2006). Learning probabilistic paradigms for morphology in a latent class model. In Wicentowski, R. and Kondrak, G., editors, *Proceedings of the Eighth Meeting of the ACL Special Interest Group on Computational Phonology and Morphology at HLT-NAACL 2006*, pages 69–78, New York, NY, USA. Association for Computational Linguistics.

Charniak, E. (2001). Immediate-head parsing for language models. In *Proceedings of 39th Annual Meeting of the Association for Computational Linguistics*, pages 124–131, Toulouse, France. Association for Computational Linguistics.

Chelba, C. and Acero, A. (2004). Adaptation of maximum entropy capitalizer: Little data can help a lot. In Lin, D. and Wu, D., editors, *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 285–292, Barcelona, Spain. Association for Computational Linguistics.

Chelba, C. and Jelinek, F. (2000). Structured language modeling. *Computer Speech and Language*, 14(4):283–332.

Chen, H., Houston, A. L., Sewell, R. R., and Schatz, B. R. (1998). Internet browsing and searching: User evaluations of category map and concept space techniques. *Journal of the American Society for Information Science*, 49(7):582–603.

Chen, S. F. (1995). Bayesian grammar induction for language modeling. In Uszkoreit, H., editor, *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 228–235, Cambridge, MA, USA. Association for Computational Linguistics.

Chen, S. F. and Goodman, J. (1999). An empirical study of smoothing techniques for language modeling. *Computer Speech and Language*, 13(4):359–393.

Chen, S. F. and Rosenfeld, R. (2000). A survey of smoothing techniques for me models. *IEEE Transactions on Speech and Audio Processing*, 8(1):37–50.

Chew, P. and Abdelali, A. (2007). Benefits of the 'massively parallel Rosetta stone': Cross-language information retrieval with over 30 languages. In Carroll, J. A., van den Bosch, A., and Zaenen, A., editors, *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL 2007)*, pages 872–879, Prague, Czech Republic. Association for Computational Linguistics.

Chiang, D. (2007). Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Chiang, D., Graehl, J., Knight, K., Pauls, A., and Ravi, S. (2010). Bayesian inference for finite-state transducers. In Kaplan, R., Burstein, J., Harper, M., and Penn, G., editors, *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 447–455, Los Angeles, California. Association for Computational Linguistics.

Chomsky, N. (1957). *Syntactic Structures*. Mouton, The Hague / Paris.

Chomsky, N. (1959). Review of B.F. Skinner's Verbal Behavior. *Language*, 35(1):26–58.

Chomsky, N. (1965). *Aspects of the Theory of Syntax*. MIT Press, Cambridge, MA, USA.

Chomsky, N. and Halle, M. (1968). *The Sound Pattern of English*. Harper & Row, New York, NY, USA.

Christodoulopoulos, C., Goldwater, S., and Steedman, M. (2010). Two decades of unsupervised POS induction: How far have we come? In Li, H. and M'arquez, L., editors, *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 575–584, Cambridge, MA. Association for Computational Linguistics.

Chrupała, G., Dinu, G., and van Genabith, J. (2008). Learning morphology with Morfette. In Calzolari, N., Choukri, K., Maegaard, B., Mariani, J., Odjik, J., Piperidis, S., and Tapias, D., editors, *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco. European Language Resources Association.

Chueh, C.-H. and Chien, J.-T. (2010). Topic cache language model for speech recognition. In *Proceedings of the 2010 IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, pages 5194–5197, Dallas, TX, USA. IEEE.

Chung, T. and Gildea, D. (2009). Unsupervised tokenization for machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 718–726, Singapore. Association for Computational Linguistics.

Clark, A. (2001). *Unsupervised Language Acquisition: Theory and Practice*. PhD thesis, School of Cognitive and Computing Sciences, Brighton, UK.

Clark, A. (2002). Memory-based learning of morphology with stochastic transducers. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 513–520, Philadelphia, PA, USA. Association for Computational Linguistics.

Clark, A., Eyraud, R., and Habrard, A. (2008). A polynomial algorithm for the inference of context free languages. In Clark, A., Coste, F., and Miclet, L., editors, *Grammatical Inference: Algorithms and Applications, 9th International Colloquium on Grammatical Inference (ICGI 2008), Saint-Malo, France, September 22–24, 2008, Proceedings*, volume 5278 of *Lecture Notes in Computer Science*, pages 29–42. Springer.

Clark, A., Eyraud, R., and Habrard, A. (2010). Using contextual representations to efficiently learn context-free languages. *Journal of Machine Learning Research*, 11:2707–2744.

Clarkson, P. and Robinson, A. (1997). Language model adaptation using mixtures and an exponentially decaying cache. In *Proceedings of the 1997 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 2, pages 799–802, Munich, Germany. IEEE.

Coccaro, N. and Jurafsky, D. (1998). Towards better integration of semantic predictors in statistical language modeling. In Mannell, R. H. and Robert-Ribes, J., editors, *Proceedings of the 5th International Conference on Spoken Language Processing (ICSLP-1998)*, volume 6, pages 2403–2406, Sydney, Australia. Australian Speech Science and Technology Association.

Coenen, F., Leng, P., Sanderson, R., and Wang, Y. J. (2007). Statistical identification of key phrases for text classification. In Perner, P., editor, *Proceedings of the 5th International Conference on Machine Learning and Data Mining in Pattern Recognition (MLDM '07), Leipzig, Germany, July 18–20, 2007*, volume 4571 of *Lecture Notes in Computer Science*, pages 838–853. Springer-Verlag, Berlin / Heidelberg, Germany.

Çöltekin, Ç. (2010). A freely available morphological analyzer for Turkish. In Calzolari, N., Choukri, K., Maegaard, B., Mariani, J., Odjik, J., Piperidis, S., Rosner, M., and Tapias, D., editors, *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC'10)*, pages 280–287, Valletta, Malta. European Language Resources Association.

Corston-Oliver, S. and Gamon, M. (2004). Normalizing German and English inflectional morphology to improve statistical word alignment. In Frederking, R. E. and Taylor, K., editors, *Machine Translation: From Real Users to Research, 6th Conference of the Association for Machine Translation in the Americas (AMTA 2004), Washington, DC, USA, September 28–October 2, 2004, Proceedings*, volume 3265 of *Lecture Notes in Computer Science*, pages 48–57, Berlin / Heidelberg, Germany. Springer.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297.

Cover, T. M. and Thomas, J. A. (2006). *Elements of information theory*. John Wiley & Sons, Inc., Hoboken, NJ, USA, 2nd edition.

Cramer, B. (2007). Limitations of current grammar induction algorithms. In Biemann, C., Seretan, V., and Riloff, E., editors, *Proceedings of the ACL 2007 Student Research Workshop*, pages 43–48, Prague, Czech Republic. Association for Computational Linguistics.

Creutz, M. (2003). Unsupervised segmentation of words using prior distributions of morph length and frequency. In Hinrichs, E. W. and Roth, D., editors, *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 280–287, Sapporo, Japan. Association for Computational Linguistics.

Creutz, M. (2006). *Induction of the Morphology of Natural Language: Unsupervised Morpheme Segmentation with Application to Automatic Speech Recognition*. PhD thesis, Helsinki University of Technology.

Creutz, M., Hirsimäki, T., Kurimo, M., Puurula, A., Pylkkönen, J., Siivola, V., Varjokallio, M., Arisoy, E., Saraçlar, M., and Stolcke, A. (2007). Morph-based speech recognition and modeling of out-of-vocabulary words across languages. *ACM Transactions on Speech and Language Processing*, 5(1):3:1–3:29.

Creutz, M. and Lagus, K. (2002). Unsupervised discovery of morphemes. In Maxwell, M., editor, *Proceedings of the ACL-02 Workshop on Morphological and Phonological Learning*, pages 21–30, Philadelphia, PA, USA. Association for Computational Linguistics.

Creutz, M. and Lagus, K. (2004). Induction of a simple morphology for highly-inflecting languages. In *Proceedings of the Seventh Meeting of the ACL Special Interest Group in Computational Phonology*, pages 43–51, Barcelona, Spain. Association for Computational Linguistics.

Creutz, M. and Lagus, K. (2005a). Inducing the morphological lexicon of a natural language from unannotated text. In Honkela, T., Könönen, V., Pöllä, M., and Simula, O., editors, *Proceedings of AKRR'05, International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning*, pages 106–113, Espoo, Finland. Helsinki University of Technology, Laboratory of Computer and Information Science.

Creutz, M. and Lagus, K. (2005b). Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0. Technical Report A81, Publications in Computer and Information Science, Helsinki University of Technology.

Creutz, M. and Lagus, K. (2007). Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing*, 4(1):3:1–3:34.

Creutz, M. and Lindén, K. (2004). Morpheme segmentation gold standards for Finnish and English. Technical Report A77, Publications in Computer and Information Science, Helsinki University of Technology.

Croft, W. (2001). *Radical Construction Grammar — Syntactic theory in typological perspective*. Oxford University Press, Oxford, UK.

Croft, W. and Cruse, D. A. (2004). *Cognitive Linguistics*. Cambridge Textbooks in Linguistics. Cambridge University Press, Cambridge, UK.

Culy, C. and Riehemann, S. Z. (2003). The limits of n-gram translation evaluation metrics. In *Proceedings of the Machine Translation Summit IX*, New Orleans, LA, USA. Association for Machine Translation in the Americas.

Curran, J. R. and Moens, M. (2002). Scaling context space. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 231–238, Philadelphia, PA, USA. Association for Computational Linguistics.

Daelemans, W. and van den Bosch, A. (2005). *Memory-Based Language Processing*. Cambridge University Press, Cambridge, UK.

Daelemans, W., van den Bosch, A., and Zavrel, J. (1999). Forgetting exceptions is harmful in language learning. *Machine Learning*, 34(1):11–43.

Darroch, J. N. and Ratcliff, D. (1972). Generalized iterative scaling for log-linear models. *Annals of Mathematical Statistics*, 43(5):1470–1480.

Dasgupta, S., Littman, M. L., and McAllester, D. A. (2001). PAC generalization bounds for co-training. In Dietterich, T. G., Becker, S., and Ghahramani, Z., editors, *Advances in Neural Information Processing Systems 14*, pages 375–382, Cambridge, MA, USA. MIT Press.

Dasgupta, S. and Ng, V. (2007). High-performance, language-independent morphological segmentation. In Sidner, C., Schultz, T., Stone, M., and Zhai, C., editors, *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 155–163, Rochester, NY, USA. Association for Computational Linguistics.

Daumé III, H. (2007). Frustratingly easy domain adaptation. In Zaenen, A. and van den Bosch, A., editors, *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263, Prague, Czech Republic. Association for Computational Linguistics.

De Bie, T. and De Moor, B. (2003). On the regularization of canonical correlation analysis. In *Proceedings of the Fourth International Symposium on Independent Component Analysis and Blind Source Separation (ICA2003)*, pages 785–790, Nara, Japan. NTT Communication Science Laboratories.

de Marcken, C. G. (1996). *Unsupervised Language Acquisition*. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology.

Deerwester, S. C., Dumais, S. T., Landauer, T. K., Furnas, G. W., and Harshman, R. A. (1990). Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407.

Deligne, S. and Bimbot, F. (1995). Language modeling by variable length sequences: Theoretical formulation and evaluation of multigrams. In *Proceedings of the 1995 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 1, pages 169–172, Detroit, MI, USA. IEEE.

Deligne, S. and Bimbot, F. (1997). Inference of variable-length linguistic and acoustic units by multigrams. *Speech Communication*, 23(3):223–241.

Demberg, V. (2007). A language-independent unsupervised model for morphological segmentation. In Zaenen, A. and van den Bosch, A., editors, *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 920–927, Prague, Czech Republic. Association for Computational Linguistics.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38.

Department of General Linguistics, University of Helsinki and Research Institute for the Languages of Finland (gatherers) (1996–1998). Finnish Parole Corpus. Available through CSC, http://www.csc.fi/.

Donoho, D. L. (2006). Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306.

Duh, K. and Kirchhoff, K. (2004). Automatic learning of language model structure. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)*, pages 148–154, Geneva, Switzerland. COLING.

Dumais, S. T. (1991). Improving the retrieval of information from external sources. *Behavior Research Methods, Instruments, & Computers*, 23(2):229–236.

Dunning, T. (1994). Statistical identification of language. Technical Report MCCS-94-273, Computing Research Lab, New Mexico State University.

Dyer, C., Muresan, S., and Resnik, P. (2008). Generalizing word lattice translation. In Moore, J. D., Teufel, S., Allan, J., and Furui, S., editors, *Proceedings of ACL-08: HLT*, pages 1012–1020, Columbus, OH, USA. Association for Computational Linguistics.

Emami, A., Xu, P., and Jelinek, F. (2003). Using a connectionist model in a syntactical based language model. In *Proceedings of the 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 372–375, Hong Kong, China. IEEE.

Fagan, J. L. (1988). The effectiveness of a nonsyntactic approach to automatic phrase indexing for document retrieval. *Journal of the American Society for Information Science*, 40(2):115–132.

Feldman, J. A. (2006). *From Molecule to Metaphor: A Neural Theory of Language*. MIT Press, Cambridge, MA, USA.

Fellbaum, C., editor (1998). *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA, USA.

Ferguson, T. S. (1973). A Bayesian analysis of some nonparametric problems. *Annals of Statistics*, 1(2):209–230.

Fillmore, C., Kay, P., and O'Connor, C. (1988). Regularity and idiomaticity in grammatical constructions: The case of *let alone*. *Language*, 64:501–538.

Finkel, J. R. and Manning, C. D. (2009). Hierarchical Bayesian domain adaptation. In Ostendorf, M., Collins, M., Narayanan, S., Oard, D. W., and Vanderwende, L., editors, *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 602–610, Boulder, Colorado. Association for Computational Linguistics.

Finn, A. and Kushmerick, N. (2006). Learning to classify documents according to genre. *Journal of the American Society for Information Science and Technology*, 57(11):1506–1518.

Fishel, M. (2009). Deeper than words: Morph-based alignment for statistical machine translation. In *Proceedings of the Conference of the Pacific Association for Computational Linguistics (PacLing 2009)*, Sapporo, Japan.

Florian, R. and Wicentowski, R. (2002). Unsupervised italian word sense disambiguation using wordnets and unlabeled corpora. In Edmonds, P., Mihalcea, R., and Saint-Dizier, P., editors, *Proceedings of the ACL-02 Workshop on Word Sense Disambiguation: Recent Successes and Future Directions*, pages 67–73. Association for Computational Linguistics.

Forney, Jr., G. D. (1973). The Viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278.

Fried, M. and Östman, J.-O., editors (2004). *Construction Grammar in a cross-language perspective*, volume 2 of *Constructional Approaches to Language*. John Benjamins, Amsterdam, Netherlands & Philadelphia, PA, USA.

Gale, W. A. and Sampson, G. (1995). Good-turing frequency estimation without tears. *Journal of Quantitative Linguistics*, 2:217–237.

Gao, J., Goodman, J. T., Cao, G., and Li, H. (2002). Exploring asymmetric clustering for statistical language modeling. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 183–190, Philadelphia, PA, USA. Association for Computational Linguistics.

Gärdenfors, P. (2000). *Conceptual Spaces — The Geometry of Thought*. MIT Press, Cambridge, MA, USA.

Gaussier, É., Renders, J.-M., Matveeva, I., Goutte, C., and Déjean, H. (2004). A geometric view on bilingual lexicon extraction from comparable corpora. In Scott, D., Daelemans, W., and Walker, M. A., editors, *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL 2004)*, pages 526–533, Barcelona, Spain. Association for Computational Linguistics.

Ghahramani, Z. (2004). Unsupervised learning. In Bousquet, O., von Luxburg, U., and Rätsch, G., editors, *Advanced Lectures on Machine Learning*, volume 3176 of *Lecture Notes in Computer Science*, pages 72–112. Springer, Berlin / Heidelberg, Germany.

Giachin, E. (1995). Phrase bigrams for continuous speech recognition. In *Proceedings of the 1995 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 1, pages 225–228, Detroit, MI, USA. IEEE.

Gildea, D. and Hofmann, T. (1999). Topic based language models using EM. In *Proceedings of 6th European Conference on Speech Communication and Technology (Eurospeech'99)*, pages 2167–2170, Budapest, Hungary. ISCA.

Gold, E. M. (1967). Language identification in the limit. *Information and Control*, 10(5):447–474.

Goldberg, A. E. (1995). *Constructions — A Construction Grammar Approach to Argument Structure*. The Chigago University Press, Chigago, IL, USA.

Goldberg, A. E. (2003). Constructions: A new theoretical approach to language. *Trends in Cognitive Sciences*, 7(5):219–224.

Goldberg, A. E. (2006). *Constructions at Work — The nature of generalization in language*. Oxford University Press, Oxford, UK.

Goldsmith, J. (2001). Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27(2):153–189.

Goldsmith, J. A. (2010). Segmentation and morphology. In Clark, A., Fox, C., and Lappin, S., editors, *The Handbook of Computational Linguistics and Natural Language Processing*, pages 364–393. Wiley-Blackwell, West Sussex, UK.

Goldwater, S. (2006). *Nonparametric Bayesian Models of Lexical Acquisition*. PhD thesis, Brown University, Providence, RI, USA.

Goldwater, S., Griffiths, T., and Johnson, M. (2006). Interpolating between types and tokens by estimating power-law generators. In Weiss, Y., Schölkopf, B., and Platt, J., editors, *Advances in Neural Information Processing Systems 18*, pages 459–466. MIT Press, Cambridge, MA, USA.

Goldwater, S., Griffiths, T. L., and Johnson, M. (2011). Producing power-law distributions and damping word frequencies with two-stage language models. *Journal of Machine Learning Research*, 12:2335–2382.

Goldwater, S. and McClosky, D. (2005). Improving statistical MT through morphological analysis. In Mooney, R., Brew, C., Chien, L.-F., and Kirchhoff, K., editors, *Proceedings of the Human Language Technology Conference / Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP 2005)*, pages 676–683, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Golénia, B., Spiegler, S., and Flach, P. A. (2010a). Unsupervised morpheme discovery with ungrade. In *Multilingual Information Access Evaluation I. Text Retrieval Experiments: 10th Workshop of the Cross-Language Evaluation Forum, CLEF 2009, Corfu, Greece, September 30 – October 2, 2009, Revised Selected Papers*, volume 6241 of *Lecture Notes in Computer Science*, pages 633–640. Springer, Berlin / Heidelberg, Germany.

Golénia, B., Spiegler, S., Ray, O., and Flach, P. (2010b). Morphological analysis using morpheme graph and mixed-integer computation of stable models. In Kurimo, M., Virpioja, S., and Turunen, V. T., editors, *Proceedings of the Morpho Challenge 2010 Workshop*, pages 25–29, Espoo, Finland. Aalto University School of Science and Technology, Department of Information and Computer Science. Technical Report TKK-ICS-R37.

Good, I. J. (1953). The population frequencies of species and the estimation of population parameters. *Biometrika*, 40(3/4):237–264.

Goodman, J. (2001a). Classes for fast maximum entropy training. In *Proceedings of the 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 561–564, Salt Lake City, UT, USA. IEEE.

Goodman, J. (2004). Exponential priors for maximum entropy models. In Susan Dumais, D. M. and Roukos, S., editors, *HLT-NAACL 2004: Main Proceedings*, pages 305–312, Boston, Massachusetts, USA. Association for Computational Linguistics.

Goodman, J. and Gao, J. (2000). Language model size reduction by pruning and clustering. In *Proceedings of the Sixth International Conference on Spoken Language Processing (ICSLP 2000)*, volume 3, pages 110–113, Beijing, China. ISCA.

Goodman, J. T. (2001b). A bit of progress in language modeling — extended version. Technical Report MSR-TR-2001-72, Microsoft Research.

Gretton, A., Herbrich, R., Smola, A., Bousquet, O., and Schölkopf, B. (2005). Kernel methods for measuring independence. *Journal of Machine Learning Research*, 6:2075–2129.

Grünwald, P. (2005). A tutorial introduction to the Minimum Description Length principle. In *Advances in Minimum Description Length: Theory and Applications*. MIT Press.

Hafer, M. A. and Weiss, S. F. (1974). Word segmentation by letter successor varieties. *Information Storage and Retrieval*, 10(11–12):371–385.

Haghighi, A., Liang, P., Berg-Kirkpatrick, T., and Klein, D. (2008). Learning bilingual lexicons from monolingual corpora. In Moore, J. D., Teufel, S., Allan, J., and Furui, S., editors, *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-08: HLT)*, pages 771–779, Columbus, OH, USA. Association for Computational Linguistics.

Hammarström, H. (2006). A naive theory of morphology and an algorithm for extraction. In Wicentowski, R. and Kondrak, G., editors, *Proceedings of the Eighth Meeting of the ACL Special Interest Group on Computational Phonology (SIGPHON)*, pages 79–88, New York, NY, USA. Association for Computational Linguistics.

Hammarström, H. (2006). Poor man's stemming: Unsupervised recognition of same-stem words. In Ng, H., Leong, M.-K., Kan, M.-Y., and Ji, D., editors, *Information Retrieval Technology*, volume 4182 of *Lecture Notes in Computer Science*, pages 323–337. Springer, Berlin / Heidelberg, Germany.

Hammarström, H. and Borin, L. (2011). Unsupervised learning of morphology. *Computational Linguistics*, 37(2):309–350.

Hardoon, D. R. and Shawe-Taylor, J. (2007). Sparse canonical correlation analysis. Technical report, University College London, London, UK.

Hardoon, D. R., Szedmak, S., and Shawe-Taylor, J. (2004). Canonical correlation analysis: An overview with application to learning methods. *Neural Computation*, 16(12):2639–2664.

Harley, T. A. (1995). *The Psychology of Language*. Psychology Press Ltd., East Sussex, UK.

Harman, D. (1992). Overview of the first text retrieval conference (trec-1). In *The First Text REtrieval Conference (TREC-1)*, pages 1–20. National Institute of Standards and Technology. NIST Special Publication 500-207.

Harris, Z. S. (1955). From phoneme to morpheme. *Language*, 31(2):190–222. Reprinted 1970 in Papers in Structural and Transformational Linguistics, Reidel Publishing Company, Dordrecht, Holland.

Hearne, M. and Way, A. (2006). Disambiguation strategies for data-oriented translation. In Hansen, V. and Maegaard, B., editors, *Proceedings of the 11th Conference of the European Association for Machine Translation (EAMT-2006)*, pages 59–68, Oslo, Norway. European Association for Machine Translation.

Heeman, P. A. and Damnati, G. (1997). Deriving phrase-based language models. In *Proceedings of the 1997 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 41–48, Waikoloa, HI, USA. IEEE.

Hinton, G. and Salakhutdinov, R. (2011). Discovering binary codes for documents by learning deep generative models. *Topics in Cognitive Science*, 3(1):74–91.

Hinton, G. E. (2009). Deep belief networks. *Scholarpedia*, 4(5):5947.

Hirsimäki, T., Creutz, M., Siivola, V., Kurimo, M., Virpioja, S., and Pylkkönen, J. (2006). Unlimited vocabulary speech recognition with morph language models applied to Finnish. *Computer Speech & Language*, 20(4):515–541.

Hirsimäki, T., Pylkkönen, J., and Kurimo, M. (2009). Importance of high-order n-gram models in morph-based speech recognition. *IEEE Transactions on Audio, Speech and Language Processing*, 17(4):724–732.

Hockett, C. F. (1954). Two models of grammatical description. *Word*, 10:210–234.

Hoeting, J. A., Madigan, D., Raftery, A., and Volinsky, C. T. (1999). Bayesian model averaging: A tutorial. *Statistical Science*, 14(4):382–417.

Hofmann, T. (1999a). Probabilistic latent semantic analysis. In Laskey, K. B. and Prade, H., editors, *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI'99)*, pages 289–296. Morgan Kaufmann.

Hofmann, T. (1999b). Probabilistic latent semantic indexing. In *Proceedings of the 22nd Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 50–57, Berkeley, CA, USA. ACM.

Hofmann, T. (2001). Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning Journal*, 42(1):177–196.

Honkela, T. (1997). *Self-Organizing Maps in Natural Language Processing*. PhD thesis, Helsinki University of Technology, Espoo, Finland.

Honkela, T., Hyvärinen, A., and Väyrynen, J. J. (2010). WordICA — emergence of linguistic representations for words by independent component analysis. *Natural Language Engineering*, 16:277–308.

Honkela, T., Pulkki, V., and Kohonen, T. (1995). Contextual relations of words in Grimm tales, analyzed by self-organizing map. In Fogelman-Soulié, F. and Gallinari, P., editors, *Proceedings of ICANN'95, International Conference on Artificial Neural Networks*, volume 2, pages 3–7, Paris, France. EC2 & Cie.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2:359–366.

Hotelling, H. (1936). Relations between two sets of variates. *Biometrika*, 28(3):321–377.

Hsu, A. S., Chater, N., and Vitanyi, P. M. B. (2011). The probabilistic analysis of language acquisition: Theoretical, computational, and experimental analysis. *Cognition*, 120:380–390.

Huang, S. and Renals, S. (2010a). Hierarchical Bayesian language models for conversational speech recognition. *IEEE Transactions on Audio, Speech and Language Processing*, 18(8):1941–1954.

Huang, S. and Renals, S. (2010b). Power law discounting for n-gram language models. In *Proceedings of the IEEE International Conference on Acoustic, Speech, and Signal Processing (ICASSP)*, pages 5178–5181, Dallas, Texas, USA. IEEE.

Huffman, D. A. (1952). A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101.

Hull, D. A. (1996). Stemming algorithms: A case study for detailed evaluation. *Journal of the American Society for Information Science*, 47(1):70–84.

Hyvärinen, A., Karhunen, J., and Oja, E. (2001). *Independent Component Analysis*. John Wiley & Sons, New York, NY, USA.

Hyvärinen, A. and Oja, E. (1996). Simple neuron models for independent component analysis. *International Journal of Neural Systems*, 7:671–687.

Hyvärinen, A. and Oja, E. (1997). A fast fixed-point algorithm for independent component analysis. *Neural Computation*, 9:1483–1492.

Isbell, C. L. and Viola, P. (1999). Restructuring sparse high dimensional data for effective information retrieval. In Kearns, M. S., Solla, S. A., and Cohn, D. A., editors, *Advances in Neural Information Processing Systems 11*, pages 480–486. The MIT Press, Cambridge, MA, USA.

Iyer, R. M. and Ostendorf, M. (1999). Modeling long distance dependence in language: topic mixtures versus dynamic cache models. *IEEE Transactions on Speech and Audio Processing*, 7(1):30–39.

Järvelin, K. and Pirkola, A. (2005). Morphological processing in mono- and cross-lingual information retrieval. In Arppe, A., Carlson, L., Lindén, K., Piitulainen, J., Suominen, M., Vainio, M., Westerlund, H., and Yli-Jyrä, A., editors, *Inquiries into Words, Constraints and Contexts. Festschrift for Kimmo Koskenniemi on his 60th Birthday*, CSLI Studies in Computational Linguistics ONLINE, pages 214–226. CSLI Publications.

Jaynes, E. T. (1957). Information theory and statistical mechanics. *The Physical Review*, 106(4):260–630.

Johnson, W. B. and Lindenstrauss, J. (1984). Extensions of Lipschitz maps into a Hilbert space. *Contemporary Mathematics*, 26:189–206.

Jones, M. N., Kintsch, W., and Mewhort, D. J. (2006). High-dimensional semantic space accounts of priming. *Journal of Memory and Language*, 55(4):534–552.

Jurafsky, D. and Martin, J. H. (2008). *Speech and Language Processing*. Prentice Hall, Upper Saddle River, New Jersey 07458, USA, 2nd edition.

Jurafsky, D., Wooters, C., Segal, J., Stolcke, A., Fosler, E., Tajchaman, G., and Morgan, N. (1995). Using a stochastic context-free grammar as a language model for speech recognition. In *Proceedings of the 1995 International Conference on Acoustics, Speech, and Signal Processing (ICASSP-95)*, volume 1, pages 189–192, Detroit, MI, USA. IEEE.

Kando, N., Kuriyama, K., Nozue, T., Eguchi, K., Kato, H., and Hidaka, S. (1999). Overview of IR tasks at the first NTCIR workshop. In Kando, N., editor, *Proceedings of the First NTCIR Workshop on Research in Japanese Text Retrieval and Term Recognition*, pages 11–44, Tokyo, Japan. National Center for Science Information Systems.

Kanerva, P., Kristoferson, J., and Holst, A. (2000). Random indexing of text samples for latent semantic analysis. In Gleitman, L. R. and Joshi, A. K., editors, *Proceedings of the 22nd Annual Conference of the Cognitive Science Society (CogSci 2000)*, page 1036, Philadelphia, PA, USA. Institute for Research in Cognitive Science, University of Pennsylvania.

Kaplan, R. M. and Kay, M. (1994). Regular models of phonological rule systems. *Computational Linguistics*, 20(3):331–378.

Karlsson, F. (2007). Constraints on multiple center-embedding of clauses. *Journal of Linguistics*, 43(2):365–392.

Karlsson, F. (2008). *Yleinen kielitiede*. Gaudeamus Helsinki University Press, Helsinki, Finland, 3rd edition.

Karttunen, L. (1998). The proper treatment of optimality in computational phonology: plenary talk. In Karttunen, L. and Oflazer, K., editors, *Proceedings of the International Workshop on Finite State Methods in Natural Language Processing (FSMNLP)*, pages 1–12, Ankara, Turkey. Association for Computational Linguistics.

Karttunen, L. and Beesley, K. R. (2005). Twenty-five years of finite-state morphology. In Arppe, A., Carlson, L., Lindén, K., Piitulainen, J., Suominen, M., Vainio, M., Westerlund, H., and Yli-Jyrä, A., editors, *Inquiries into Words, Constraints and Contexts. Festschrift for Kimmo Koskenniemi on his 60th Birthday*, CSLI Studies in Computational Linguistics ONLINE, pages 71–83. CSLI Publications.

Karttunen, L., Kaplan, R. M., and Zaenen, A. (1992). Two-level morphology with composition. In Boitet, C., editor, *The 15th International Conference on Computational Linguistics (COLING 1992)*, volume 1, pages 141–148, Nantes, France. International Committee on Computational Linguistics.

Kasami, T. (1965). An efficient recognition and syntax-analysis algorithm for context-free languages. Technical Report AFCRL-65-758, Air Force Cambridge Research Lab, Bedford, MA, USA.

Kaski, S. (1998). Dimensionality reduction by random mapping: Fast similarity computation for clustering. In *Proceedings of 1998 IEEE International Joint Conference on Neural Networks (IJCNN)*, volume 1, pages 413–418, Anchorage, AK, USA. IEEE.

Kaski, S., Honkela, T., Lagus, K., and Kohonen, T. (1998). WEBSOM—self-organizing maps of document collections. *Neurocomputing*, 21:101–117.

Kay, J. (1992). Feature discovery under contextual supervision using mutual information. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN 1992)*, volume 4, pages 79–84. IEEE, Baltimore, MD, USA.

Keshava, S. and Pitler, E. (2006). A simpler, intuitive approach to morpheme induction. In Kurimo, M., Creutz, M., and Lagus, K., editors, *Proceedings of the PASCAL Challenge Workshop on Unsupervised Segmentation of Words into Morphemes*, pages 28–32, Venice, Italy. PASCAL European Network of Excellence.

Kirchhoff, K., Vergyri, D., Bilmes, J., Duh, K., and Stolcke, A. (2006). Morphology-based language modeling for conversational Arabic speech recognition. *Computer Speech & Language*, 20(4):589–608.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680.

Klakow, D. (1998). Language-model optimization by mapping of corpora. In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 2, pages 701–704, Seattle, WA, USA. IEEE.

Klami, M. (2005). Unsupervised discovery of morphs in children's stories and their use in self-organizing map -based analysis. Master's thesis, University of Helsinki, Department of General Linguistics, Helsinki, Finland.

Klein, D. and Manning, C. (2004). Corpus-based induction of syntactic structure: Models of dependency and constituency. In Scott, D., Daelemans, W., and Walker, M. A., editors, *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 478–485, Barcelona, Spain. Association for Computational Linguistics.

Klein, D. and Manning, C. D. (2002). A generative constituent-context model for improved grammar induction. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 128–135, Philadelphia, PA, USA. Association for Computational Linguistics.

Kneser, R. (1996). Statistical language modeling using a variable context length. In *Proceedings of Fourth International Conference Spoken Language Processing (ICSLP)*, volume 1, pages 494–497, Philadelphia, PA, USA. ISCA.

Kneser, R. and Ney, H. (1995). Improved backing-off for *m*-gram language modeling. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 1:181–184.

Knight, K. and Graehl, J. (1998). Machine transliteration. *Computational Linguistics*, 24(4):599–612.

Koehn, P. (2005). Europarl: A parallel corpus for statistical machine translation. In *Proceedings of the 10th Machine Translation Summit*, pages 79–86, Phuket, Thailand. Asia-Pacific Association for Machine Translation.

Koehn, P. and Hoang, H. (2007). Factored translation models. In Eisner, J., editor, *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 868–876, Prague, Czech Republic. Association for Computational Linguistics.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. (2007). Moses: Open source toolkit for statistical machine translation. In Ananiadou, S., editor, *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume, Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

Koehn, P. and Monz, C. (2005). Shared task: Statistical machine translation between European languages. In Koehn, P., Martin, J., Mihalcea, R., Monz, C., and Pedersen, T., editors, *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, pages 119–124, Ann Arbor, MI, USA. Association for Computational Linguistics.

Koehn, P., Och, F. J., and Marcu, D. (2003). Statistical phrase-based translation. In Hearst, M. and Ostendorf, M., editors, *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (HLT-NAACL)*, pages 48–54, Edmonton, Canada. Association for Computational Linguistics.

Kohonen, O., Virpioja, S., and Klami, M. (2009a). Allomorfessor: Towards unsupervised morpheme analysis. In *Evaluating Systems for Multilingual and Multimodal Information Access: 9th Workshop of the Cross-Language Evaluation Forum, CLEF 2008, Aarhus, Denmark, September 17–19, 2008, Revised Selected Papers*, volume 5706 of *Lecture Notes in Computer Science*, pages 975–982. Springer, Berlin / Heidelberg, Germany.

Kohonen, O., Virpioja, S., and Lagus, K. (2009b). A constructionist approach to grammar inference. In *NIPS Workshop on Grammar Induction, Representation of Language and Language Learning*, Whistler, Canada. On-line: `http://www.cs.ucl.ac.uk/staff/rmartin/grll09/`. Retrieved 2012-06-01.

Kohonen, O., Virpioja, S., Leppänen, L., and Lagus, K. (2010). Semi-supervised extensions to Morfessor Baseline. In Kurimo, M., Virpioja, S., and Turunen, V. T., editors, *Proceedings of the Morpho Challenge 2010 Workshop*, pages 30–34, Espoo, Finland. Aalto University School of Science and Technology, Department of Information and Computer Science. Technical Report TKK-ICS-R37.

Kohonen, T. (1982). Self-organizing formation of topologically correct features maps. *Biological Cybernetics*, 43(1):59–69.

Kohonen, T. (1995). *Self-Organizing Maps*. Springer.

Kohonen, T., Kaski, S., Lagus, K., Salojärvi, J., Honkela, J., Paatero, V., and Saarela, A. (2000). Self organization of a massive document collection. *IEEE Transactions on Neural Networks*, 11(3):574–585.

Kolmogorov, A. (1965). Three approaches to the quantitative definition of information. *Problems of Information Transmission*, 1(1):1–7.

Koskenniemi, K. (1983). *Two-level morphology: A general computational model for word-form recognition and production*. PhD thesis, University of Helsinki.

Koster, C. H. A., Beney, J., Verberne, S., and Vogel, M. (2011). Phrase-based document categorization. In Lupu, M., Mayer, K., Tait, J., and Trippe, A. J., editors, *Current Challenges in Patent Information Retrieval*, volume 29 of *The Information Retrieval Series*, pages 263–286. Springer-Verlag, Berlin / Heidelberg, Germany.

Koster, C. H. A. and Seutter, M. (2003). Taming wild phrases. In Sebastiani, F., editor, *Proceedings of the 25th European Conference on Information Retrieval Research (ECIR 2003), Pisa, Italy, April 14–16, 2003*, volume 2633 of *Lecture Notes in Computer Science*, pages 161–176. Springer-Verlag, Berlin / Heidelberg, Germany.

Kostić, A. (1991). Informational approach to processing inflected morphology: Standard data reconsidered. *Psychological Research*, 53(1):62–70.

Kraft, L. G. (1949). A device for quantizing, grouping, and coding amplitude modulated pulses. Master's thesis, Electrical Engineering Department, Massachusetts Institute of Technology, Cambridge, MA, USA.

Kübler, S., McDonald, R., and Nivre, J. (2009). *Dependency Parsing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool.

Kuhn, H. W. (1955). The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97.

Kuhn, R. and De Mori, R. (1990). A cache-based natural language model for speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(6):570–583.

Kukich, K. (1992). Techniques for automatically correcting words in text. *ACM Computing Surveys*, 24(4):377–439.

Kumar, S. and Byrne, W. (2004). Minimum Bayes-Risk decoding for statistical machine translation. In Dumais, S., Marcu, D., and Roukos, S., editors, *HLT-NAACL 2004: Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, Proceedings of the Main Conference*, pages 169–176, Boston, MA, USA. Association for Computational Linguistics.

Kumar, S., Deng, Y., and Byrne, W. (2006). A weighted finite state transducer translation template model for statistical machine translation. *Journal of Natural Language Engineering*, 12(1):35–75.

Kuo, H.-K. J. and Reichl, W. (1999). Phrase-based language models for speech recognition. In *Proceedings of 6th European Conference on Speech Communication and Technology (Eurospeech'99)*, pages 1595–1598, Budapest, Hungary. ISCA.

Kurimo, M. (1997). Training mixture density HMMs with SOM and LVQ. *Computer Speech and Language*, 11(4):321–343.

Kurimo, M. (2002). Thematic indexing of spoken documents by using self-organizing maps. *Speech Communication*, 38(1-2):29–44.

Kurimo, M., Creutz, M., and Lagus, K. (2006a). Unsupervised segmentation of words into morphemes — Challenge 2005, an introduction and evaluation report. In Kurimo, M., Creutz, M., and Lagus, K., editors, *Proceedings of the PASCAL Challenge Workshop on Unsupervised segmentation of words into morphemes*, pages 1–11, Venice, Italy. PASCAL European Network of Excellence.

Kurimo, M., Creutz, M., and Varjokallio, M. (2008). Morpho Challenge evaluation using a linguistic Gold Standard. In *Advances in Multilingual and Multimodal Information Retrieval, 8th Workshop of the Cross-Language Evaluation Forum, CLEF 2007, Budapest, Hungary, September 19–21, 2007, Revised Selected Papers*, volume 5152 of *Lecture Notes in Computer Science*, pages 864–873. Springer, Berlin / Heidelberg, Germany.

Kurimo, M. and Lagus, K. (2002). An efficiently focusing large vocabulary language model. In *International Conference on Artificial Neural Networks (ICANN'02)*, pages 1068–1073, Madrid, Spain.

Kurimo, M., Puurula, A., Arisoy, E., Siivola, V., Hirsimäki, T., Pylkkönen, J., Alumäe, T., and Saraçlar, M. (2006b). Unlimited vocabulary speech recognition for agglutinative languages. In Moore, R. C., Bilmes, J., Chu-Carroll, J., and Sanderson, M., editors, *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 487–494, New York, NY, USA. Association for Computational Linguistics.

Kurimo, M., Turunen, V., and Varjokallio, M. (2009). Overview of Morpho Challenge 2008. In *Evaluating Systems for Multilingual and Multimodal Information Access, 9th Workshop of the Cross-Language Evaluation Forum, CLEF 2008, Aarhus, Denmark, September 17–19, 2008, Revised Selected Papers*, volume 5706 of *Lecture Notes in Computer Science*, pages 951–966. Springer, Berlin / Heidelberg, Germany.

Kurimo, M., Virpioja, S., Turunen, V., and Lagus, K. (2010a). Morpho challenge 2005-2010: Evaluations and results. In Heinz, J., Cahill, L., and Wicentowski, R., editors, *Proceedings of the 11th Meeting of the ACL Special Interest Group on Computational Morphology and Phonology*, pages 87–95, Uppsala, Sweden. Association for Computational Linguistics.

Kurimo, M., Virpioja, S., and Turunen, V. T. (2010b). Overview and results of Morpho Challenge 2010. In *Proceedings of the Morpho Challenge 2010 Workshop*, pages 7–24, Espoo, Finland. Aalto University School of Science and Technology, Department of Information and Computer Science. Technical Report TKK-ICS-R37.

Kurimo, M., Virpioja, S., Turunen, V. T., Blackwood, G. W., and Byrne, W. (2010c). Overview and results of Morpho Challenge 2009. In *Multilingual Information Access Evaluation I. Text Retrieval Experiments: 10th Workshop of the Cross-Language Evaluation Forum, CLEF 2009, Corfu, Greece, September 30 – October 2, 2009, Revised Selected Papers*, volume 6241 of *Lecture Notes in Computer Science*, pages 578–597. Springer, Berlin / Heidelberg, Germany.

Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In Brodley, C. E. and Danyluk, A. P., editors, *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001)*, pages 282–289, Williamstown, MA, USA. Morgan Kaufmann.

Lagus, K. (2000). Text mining with the WEBSOM. *Acta Polytechnica Scandinavica, Mathematics and Computing Series*, No. 110. D.Sc. (Tech.) Thesis, Helsinki University of Technology, Finland.

Lagus, K., Kaski, S., and Kohonen, T. (2004). Mining massive document collections by the WEBSOM method. *Information Sciences*, 163(1–3):135–156.

Lagus, K. and Kurimo, M. (2002). Language model adaptation in speech recognition using document maps. In *IEEE Workshop on Neural Networks for Signal Processing (NNSP'02)*, pages 627–636, Martigny, Switzerland. IEEE.

Lai, P. L. and Fyfe, C. (2000). Kernel and nonlinear canonical correlation analysis. *International Journal of Neural Systems*, 10(5):365–377.

Laine, M., Niemi, J., Koivuselkä-Sallinen, P., and Hyönä, J. (1995). Morphological processing of polymorphemic words in a highly inflecting language. *Cognitive Neuropsychology*, 12:457–502.

Laine, M. and Virtanen, P. (1996). *WordMill Lexical Search Program*. Centre for Cognitive Neuroscience, University of Turku.

Lakoff, G. (1987). *Women, Fire, and Dangerous Things: What Categories Reveal about the Mind*. University of Chicago Press, Chicago, IL, USA.

Lakoff, G. and Johnson, M. (2003). *Metaphors We Live By*. Chigago University Press, Chigago, IL, USA.

Lamar, M., Maron, Y., Johnson, M., and Bienenstock, E. (2010). SVD and clustering for unsupervised POS tagging. In Hajič, J., Carberry, S., Clark, S., and Nivre, J., editors, *Proceedings of the ACL 2010 Conference Short Papers*, pages 215–219, Uppsala, Sweden. Association for Computational Linguistics.

Landauer, T. K. and Dumais, S. T. (1997). A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 104(2):211–240.

Langacker, R. W. (1987). *Theoretical Prerequisites*, volume 1 of *Foundations of Construction Grammar*. Stanford University Press, Stanford, CA, USA.

Langacker, R. W. (1991). *Descriptive Application*, volume 2 of *Foundations of Construction Grammar*. Stanford University Press, Stanford, CA, USA.

Lappin, S. and Shieber, S. M. (2007). Machine learning theory and practice as a source of insight into universal grammar. *Journal of Linguistics*, 43:393–427.

Lavallée, J.-F. and Langlais, P. (2010). Unsupervised morphological analysis by formal analogy. In *Multilingual Information Access Evaluation I. Text Retrieval Experiments: 10th Workshop of the Cross-Language Evaluation Forum, CLEF 2009, Corfu, Greece, September 30 - October 2, 2009, Revised Selected Papers*, volume 6241 of *Lecture Notes in Computer Science*, pages 617–624. Springer, Berlin / Heidelberg, Germany.

Lee, D. D. and Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791.

Lee, Y.-S. (2004). Morphological analysis for statistical machine translation. In *Proceedings of the HLT-NAACL 2004*, pages 57–60, Boston, USA.

Lehtonen, M., Cunillera, T., Rodríguez-Fornells, A., Hultén, A., Tuomainen, J., and Laine, M. (2007). Recognition of morphologically complex words in Finnish: evidence from event-related potentials. *Brain Research*, 1148:123–137.

Lehtonen, M. and Laine, M. (2003). How word frequency affects morphological processing in monolinguals and bilinguals. *Bilingualism: Language and Cognition*, 6:213–225.

Leino, P. (1999). *Suomen kielen kognitiivista kielioppia. 1, Polysemia - kielen moniselitteisyys*. Helsingin yliopiston suomen kielen laitos, Helsinki, Finland, 2nd edition.

Leurgans, S. E., Moyeed, R. A., and Silverman, B. W. (1993). Canonical correlation analysis when the data are curves. *Journal of the Royal Statistical Society. Series B (Methodological)*, 55(3):725–740.

Levinson, S. C. (1983). *Pragmatics*. Cambridge Textbooks in Linguistics. Cambridge University Press, Cambridge, UK.

Lewis, D. D. (1992). An evaluation of phrasal and clustered representations on a text categorization task. In Belkin, N. J., Ingwersen, P., and Pejtersen, A. M., editors, *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '92)*, pages 37–50, Copenhagen, Denmark. ACM.

Lewis, D. D. and Tong, R. M. (1992). Text filtering in MUC-3 and MUC-4. In *Proceedings of the 4th Conference on Message Understanding (MUC-4)*, pages 51–66, McLean, VA, USA. Morgan Kaufmann Publishers.

Li, M., Chen, X., Li, X., Ma, B., and Vitanyi, P. (2004). The similarity metric. *IEEE Transactions on Information Theory*, 50(12):3250–3264.

Li, W. and McCallum, A. (2005). Semi-supervised sequence modeling with syntactic topic models. In Veloso, M. M. and Kambhampati, S., editors, *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI-05)*, pages 813–818, Pittsburgh, PA, USA. AAAI Press.

Li, Y. and Shawe-Taylor, J. (2007). Advanced learning algorithms for cross-language patent retrieval and classification. *Information Processing and Management*, 43(5):1183–1199.

Lignos, C. (2010). Learning from unseen data. In Kurimo, M., Virpioja, S., and Turunen, V. T., editors, *Proceedings of the Morpho Challenge 2010 Workshop*, pages 35–38, Espoo, Finland. Aalto University School of Science and Technology, Department of Information and Computer Science. Technical Report TKK-ICS-R37.

Lignos, C., Chan, E., Marcus, M. P., and Yang, C. (2010a). A rule-based acquisition model adapted for morphological analysis. In *Multilingual Information Access Evaluation I. Text Retrieval Experiments: 10th Workshop of the Cross-Language Evaluation Forum, CLEF 2009, Corfu, Greece, September 30 – October 2, 2009, Revised Selected Papers*, volume 6241 of *Lecture Notes in Computer Science*, pages 658–665. Springer, Berlin / Heidelberg, Germany.

Lignos, C., Chan, E., Yang, C., and Marcus, M. P. (2010b). Evidence for a morphological acquisition model from development data. In Franich, K., Iserman, K. M., and Keil, L. L., editors, *Proceedings of the 34th Annual Boston University Conference on Language Development (BUCLD 34)*, volume 2, pages 269–280. Cascadilla Press.

Lim, H. S., Nam, K., and Hwang, Y. (2005). A computational model of korean mental lexicon. In Gervasi, O., Gavrilova, M., Kumar, V., Laganà, A., Lee, H., Mun, Y., Taniar, D., and Tan, C., editors, *Computational Science and Its Applications — ICCSA 2005*, volume 3480 of *Lecture Notes in Computer Science*, pages 17–26. Springer, Berlin / Heidelberg, Germany.

Lindén, K. (2003). Word sense disambiguation with THESSOM. In *Intelligent Systems and Innovational Computing — Proceedings of the Workshop on Self-Organizing Networks (WSOM 2003)*, Kitakuyshu, Japan. Kyushu Institute of Technology.

Lindén, K. (2008). A probabilistic model for guessing base forms of new words by analogy. In Gelbukh, A. F., editor, *Proceedings of the 9th international conference on Computational Linguistics and Intelligent Text Processing (CICLing 2008), Haifa, Israel, February 17–23, 2008*, volume 4919 of *Lecture Notes on Computer Science*, pages 106–116. Springer-Verlag, Berlin / Heidelberg, Germany.

Lindén, K. (2009). Guessers for finite-state transducer lexicons. In Gelbukh, A. F., editor, *Proceedings of the 10th International Conference on Computational Linguistics and Intelligent Text Processing (CICLing 2009), Mexico City, Mexico, March 1–7, 2009*, volume 5449 of *Lecture Notes in Computer Science*, pages 158–169. Springer-Verlag, Berlin / Heidelberg, Germany.

Lindén, K. and Lagus, K. (2002). Word sense disambiguation in document space. In *Proceedings of the 2002 IEEE International Conference on Systems, Man and Cybernetics*, Hammamet, Tunisia. IEEE.

Lloyd, S. P. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–136.

Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., and Watkins, C. (2002). Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–444.

Lund, K. and Burgess, C. (1996). Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments & Computers*, 28(2):203–208.

MacKay, D. J. (2003). *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, Cambridge, UK.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In Le Cam, L. M. and Neyman, J., editors, *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability. Volume I: Statistics*, pages 281–297, Berkeley and Los Angeles, CA, USA. University of California Press.

MacWhinney, B. (2000). *The CHILDES Project: Tools for analyzing talk*. Lawrence Erlbaum Associates, Mahwah, NJ, USA, 3rd edition.

Magdy, W. and Darwish, K. (2006). Arabic OCR error correction using character segment correction, language modeling, and shallow morphology. In Jurafsky, D. and Gaussier, E., editors, *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 408–414, Sydney, Australia. Association for Computational Linguistics.

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

Manning, C. D. and Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, MA, USA.

Mathias, L. and Byrne, W. (2006). Statistical phrase-based speech translation. In *Proceedings of the 31st IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Toulouse, France. IEEE.

Matthews, P. H. (1991). *Morphology*. Cambridge Textbooks in Linguistics. Cambridge University Press, Cambridge, UK, 2nd edition.

McCallum, A., Freitag, D., and Pereira, F. (2000). Maximum entropy Markov models for information extraction and segmentation. In Langley, P., editor, *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, pages 591–598, Stanford, CA, USA. Morgan Kaufmann.

McClelland, J. L. and Patterson, K. (2002). Rules or connections in past-tense inflections: what does the evidence rule out? *Trends in cognitive sciences*, 6(11):465–472.

McNamee, P. (2008). Retrieval experiments at Morpho Challenge 2008. In Peters, C., editor, *Working Notes for the CLEF 2008 Workshop*, Aarhus, Denmark. Cross-Language Evaluation Forum.

Meila, M. (2003). Comparing clusterings by the variation of information. In Schölkopf, B. and Warmuth, M. K., editors, *Learning Theory and Kernel Machines*, volume 2777 of *Lecture Notes in Computer Science*, pages 173–187. Springer Berlin / Heidelberg.

Melzer, T., Reiter, M., and Bischof, H. (2001). Nonlinear feature extraction using generalized canonical correlation analysis. In Dorffner, G., Bischof, H., and Hornik, K., editors, *Proceedings of the International Conference on Artificial Neural Networks (ICANN '01)*, volume 2130 of *Lecture Notes in Computer Science*, pages 353–360. Springer-Verlag, Berlin / Heidelberg, Germany.

Menon, A. K. and Elkan, C. (2011). Fast algorithms for approximating the singular value decomposition. *ACM Transactions on Knowledge Discovery from Data*, 5(2):13:1–13:36.

Mermer, C. and Akın, A. A. (2010). Unsupervised search for the optimal segmentation for statistical machine translation. In *Proceedings of the ACL 2010 Student Research Workshop*, pages 31–36, Uppsala, Sweden. Association for Computational Linguistics.

Mihajlik, P., Tüske, Z., Tarján, B., Németh, B., and Fegyó, T. (2010). Improved recognition of spontaneous Hungarian speech — morphological and acoustic modeling techniques for a less resourced task. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(6):1588–1600.

Mihalcea, R. and Simard, M. (2005). Parallel texts. *Natural Language Engineering*, 11(3):239–246.

Mikolov, T., Deoras, A., Kombrink, S., Burget, L., and Černocký, J. (2011). Empirical evaluation and combination of advanced language modeling techniques. In *Proceedings of the 12th Annual Conference of the International Speech Communication Association (INTERSPEECH 2011)*, pages 605–608, Florence, Italy. ISCA.

Mikolov, T., Karafiát, M., Burget, L., Černocký, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In Kobayashi, T., Hirose, K., and Nakamura, S., editors, *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010)*, pages 1045–1048, Makuhari, Chiba, JP. ISCA.

Milin, P., Kuperman, V., Kostic, A., and Baayen, R. H. (2009). Paradigms bit by bit: an information theoretic approach to the processing of paradigmatic structure in inflection and derivation. In Blevins, J. and Blevins, J., editors, *Analogy in grammar: Form and acquisition*, pages 214–252. Oxford University Press, Oxford, UK.

Miller, G. A. (1995). WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41.

Minier, Z., Bodó, Z., and Csató, L. (2007). Wikipedia-based kernels for text categorization. In Negru, V., Jebelean, T., Petcu, D., and Zaharie, D., editors, *Proceedings of the 9th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC'07)*, pages 157–164, Timisoara, Romania. IEEE Computer Society.

Mitchell, J. and Lapata, M. (2008). Vector-based models of semantic composition. In Moore, J. D., Teufel, S., Allan, J., and Furui, S., editors, *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-08:HLT)*, pages 236–244, Columbus, OH, USA. Association for Computational Linguistics.

Mitchell, T. M. (1997). *Machine Learning*. McGraw Hill, New York, NY, USA.

Mnih, A. and Hinton, G. (2007). Three new graphical models for statistical language modelling. In Ghahramani, Z., editor, *Proceedings of the Twenty-Fourth International Conference on Machine Learning (ICML 2007), Corvallis, Oregon, USA, June 20–24, 2007*, volume 227 of *ACM International Conference Proceeding Series*, pages 641–648. ACM, New York, NY, USA.

Mnih, A. and Hinton, G. (2008). A scalable hierarchical distributed language model. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 21*, pages 1081–1088. MIT Press, Cambridge, MA, USA.

Mnih, A., Yuecheng, Z., and Hinton, G. ((2009). Improving a statistical language model through non-linear prediction. *Neurocomputing*, 72(7–9):1414–1418.

Mochihashi, D. and Sumita, E. (2008). The infinite Markov model. In Platt, J., Koller, D., Singer, Y., and Roweis, S., editors, *Advances in Neural Information Processing Systems 20*, pages 1017–1024. MIT Press, Cambridge, MA, USA.

Mohri, M., Pereira, F., and Riley, M. (2002). Weighted finite-state transducers in speech recognition. *Computer Speech & Language*, 16(1):69–88.

Monson, C., Carbonell, J., Lavie, A., and Levin, L. (2008). ParaMor: Finding paradigms across morphology. In *Advances in Multilingual and Multimodal Information Retrieval, 8th Workshop of the Cross-Language Evaluation Forum, CLEF 2007, Budapest, Hungary, September 19–21, 2007, Revised Selected Papers*, volume 5152 of *Lecture Notes in Computer Science*, pages 900–907. Springer, Berlin / Heidelberg, Germany.

Monson, C., Carbonell, J., Lavie, A., and Levin, L. (2009). ParaMor and Morpho Challenge 2008. In *Evaluating Systems for Multilingual and Multimodal Information Access, 9th Workshop of the Cross-Language Evaluation Forum, CLEF 2008, Aarhus, Denmark, September 17–19, 2008, Revised Selected Papers*, volume 5706 of *Lecture Notes in Computer Science*, pages 967–974. Springer, Berlin / Heidelberg, Germany.

Monson, C., Hollingshead, K., and Roark, B. (2010). Simulating morphological analyzers with stochastic taggers for confidence estimation. In *Multilingual Information Access Evaluation I. Text Retrieval Experiments: 10th Workshop of the Cross-Language Evaluation Forum, CLEF 2009, Corfu, Greece, September 30 – October 2, 2009, Revised Selected Papers*, volume 6241 of *Lecture Notes in Computer Science*, pages 649–657. Springer, Berlin / Heidelberg, Germany.

Morin, F. and Bengio, Y. (2005). Hierarchical probabilistic neural network language model. In Cowell, R. G. and Ghahramani, Z., editors, *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics (AISTATS 2005)*, pages 246–252, Barbados. Society for Artificial Intelligence and Statistics.

Moscoso del Prado Martín, F., Kostić, A., and Baayen, R. H. (2004). Putting the bits together: An information theoretical perspective on morphological processing. *Cognition*, 94:1–18.

Müller, P. and Quintana, F. A. (2004). Nonparametric Bayesian data analysis. *Statistical science*, 19(1):95–110.

Munkres, J. (1957). Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38.

Nadeau, D. and Sekine, S. (2007). A survey of named entity recognition and classification. *Linguisticae Investigationes*, 30(1):3–26.

Nakov, P., Popova, A., and Mateev, P. (2001). Weight functions impact on LSA performance. In *Proceedings of the EuroConference on Recent Advances in Natural Language Processing (RANLP 2001)*, pages 187–193, Tzigov Chark, Bulgaria. Bulgarian Academy of Sciences.

Nelson, D. L., McEvoy, C. L., and Schreiber, T. A. (1998). The University of South Florida word association, rhyme, and word fragment norms. On-line: `http://web.usf.edu/FreeAssociation/`. Retrieved 2010-10-07. University of South Florida, Tampa, FL, USA.

New, B., Brysbaert, M., Segui, J., Ferrand, L., and Rastle, K. (2004). The processing of singular and plural nouns in French and English. *Journal of Memory and Language*, 51:568–585.

Ney, H., Essen, U., and Kneser, R. (1994). On structuring probabilistic dependencies in stochastic language modeling. *Computer Speech and Language*, 8:1–28.

Nicolas, L., Farré, J., and Molinero, M. A. (2010). Unsupervised learning of concatenative morphology based on frequency-related form occurrence. In Kurimo, M., Virpioja, S., and Turunen, V. T., editors, *Proceedings of the Morpho Challenge 2010 Workshop*, pages 39–43, Espoo, Finland. Aalto University School of Science and Technology, Department of Information and Computer Science. Technical Report TKK-ICS-R37.

Niemi, J., Laine, M., and Tuominen, J. (1994). Cognitive morphology in Finnish: foundations of a new model. *Language and Cognitive Processes*, 9:423–446.

Niesler, T. R., Whittaker, E. W. D., and Woodland, P. C. (1998). Comparison of part-of-speech and automatically derived category-based language models for speech recognition. In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1, pages 177–180, Seattle, WA, USA. IEEE.

Niesler, T. R. and Woodland, P. C. (1996a). Combination of word-based and category-based language models. In *Proceedings of Fourth International Conference Spoken Language Processing (ICSLP)*, volume 1, pages 220–223, Philadelphia, PA, USA. ISCA.

Niesler, T. R. and Woodland, P. C. (1996b). A variable-length category-based n-gram language model. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1, pages 164–167, Atlanta, GA, USA. IEEE.

Niesler, T. R. and Woodland, P. C. (1999). Variable-length category n-gram language models. *Computer Speech and Language*, 13(1):99–124.

Nießen, S. and Ney, H. (2004). Statistical machine translation with scarce resources using morpho-syntactic information. *Computational Linguistics*, 30(2):181–204.

Nigam, K., McCallum, A. K., Thrun, S., and Mitchell, T. (2000). Text classification from labeled and unlabeled documents using em. *Machine Learning*, 39(2–3):103–134.

Norris, D. (2005). How do computational models help us build better theories? In Cutler, A., editor, *Twenty-First Century Psycholinguistics: Four Cornerstones*, pages 331–346. Lawrence Erlbaum Associates, Inc., Mahwah, NJ, USA.

Norris, D. (2006). The Bayesian reader: Explaining word recognition as an optimal Bayesian decision process. *Psychological Review*, 113(2):327–357.

Novak, M. and Mammone, R. (2001). Use of non-negative matrix factorization for language model adaptation in a lecture transcription task. In *Proceedings of 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 541–544, Salt Lake City, UT, USA. IEEE.

Oflazer, K. and El-Kahlout, İ. D. (2007). Exploring different representational units in English-to-Turkish statistical machine translation. In Callison-Burch, C., Koehn, P., Fordyce, C. S., and Monz, C., editors, *Proceedings of the Statistical Machine Translation Workshop at ACL 2007*, pages 25–32, Prague, Czech Republic. Association for Computational Linguistics.

Oflazer, K., McShane, M., and Nirenburg, S. (2001). Bootstrapping morphological analyzers by combining human elicitation and machine learning. *Computational Linguistics*, 27(1):59–85.

Oja, E. (1982). A simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, 15:267–273.

Paatero, P. and Tapper, U. (1994). Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5:111–126.

Palmer, A., Moon, T., and Baldridge, J. (2009). Evaluating automation strategies in language documentation. In Ringger, E., Haertel, R., and Tomanek, K., editors, *Proceedings of the NAACL HLT 2009 Workshop on Active Learning for Natural Language Processing*, pages 36–44, Boulder, CO, USA. Association for Computational Linguistics.

Pang, B., Lee, L., and Vaithyanathan, S. (2002). Thumbs up? Sentiment classification using machine learning techniques. In Hajič, J. and Matsumoto, Y., editors, *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 79–86, Philadelphia, PA, USA. Association for Computational Linguistics.

Papadimitriou, C. H., Raghavan, P., Tamaki, H., and Vempala, S. (2000). Latent semantic indexing: A probabilistic analysis. *Journal of Computer and System Sciences*, 61(2):217–235.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, PA, USA. Association for Computational Linguistics.

Paul, M., Finch, A., and Sumita, E. (2010). Integration of multiple bilingually-learned segmentation schemes into statistical machine translation. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 400–408, Uppsala, Sweden. Association for Computational Linguistics.

Pearson, K. (1896). Mathematical contributions to the theory of evolution. iii. regression, heredity and panmixia. *Philosophical Transactions of the Royal Society A*, 187:253–318.

Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. *Philosophical Magazine 2*, pages 559–572.

Peng, F., Feng, F., and McCallum, A. (2004). Chinese segmentation and new word detection using conditional random fields. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)*, pages 562–568, Geneva, Switzerland. COLING.

Peters, C., editor (2001). *Cross-Language Information Retrieval and Evaluation, Workshop of Cross-Language Evaluation Forum, CLEF 2000, Lisbon, Portugal, September 21-22, 2000, Revised Papers*, volume 2069 of *Lecture Notes in Computer Science*. Springer, Berling / Heidelberg, Germany.

Pinker, S. and Ullman, M. T. (2002). The past and future of the past tense. *Trends in Cognitive Sciences*, 6(11):456–463.

Pirkola, A. (2001). Morphological typology of language for IR. *Journal of Documentation*, 57(3):330–348.

Pitman, J. and Yor, M. (1997). The two-parameter Poisson–Dirichlet distribution derived from a stable subordinator. *Annals of Probability*, 25(2):855–900.

Ponte, J. M. and Croft, W. B. (1998). A language modeling approach to information retrieval. In Croft, W. B., Moffat, A., Rijsbergen, C. J. V., Wilkinson, R., and Zobel, J., editors, *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 275–281, Melbourne, Australia. ACM.

Ponvert, E., Baldridge, J., and Erk, K. (2010). Simple unsupervised identification of low-level constituents. In *Proceedings of the Fourth International Conference on Semantic Computing (ICSC 2010)*, pages 24–31, Pittsburgh, PA, USA. IEEE.

Ponvert, E., Baldridge, J., and Erk, K. (2011). Simple unsupervised grammar induction from raw text with cascaded finite state models. In Matsumoto, Y. and Mihalcea, R., editors, *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1077–1086, Portland, OR, USA. Association for Computational Linguistics.

Poon, H., Cherry, C., and Toutanova, K. (2009). Unsupervised morphological segmentation with log-linear models. In Ostendorf, M., Collins, M., Narayanan, S., Oard, D. W., and Vanderwende, L., editors, *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL '09)*, pages 209–217, Boulder, CO, USA. Association for Computational Linguistics.

Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, 14(3):130–137.

Potamianos, G. and Jelinek, F. (1998). A study of n-gram and decision tree letter language modeling methods. *Speech Communication*, 24(3):171–192.

Poutsma, A. (2000). Data-oriented translation. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000)*, volume 2, pages 635–641, Saarbrücken, Germany. Morgan Kaufmann.

Prince, A. and Smolensky, P. (1993). Optimality theory: Constraint interaction in generative grammar. Technical Report RuCCS-TR-2, Rutgers Cognitive Science Center, Rutgers University. ROA Version, 8/2002, on-line: `roa.rutgers.edu/files/537-0802/537-0802-PRINCE-0-0.PDF`. Retrieved 2011-12-12.

Pullum, G. and Rawlins, K. (2007). Argument or no argument? *Linguistics and Philosophy*, 30(2):277–287.

Pullum, G. K. and Gazdar, G. (1982). Natural languages and context-free languages. *Linguistics and Philosophy*, 4(4):471–504.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1(1):81–106.

Ramshaw, L. A. and Marcus, M. P. (1995). Text chunking using transformation-based learning. In *Proceedings of the Third ACL Workshop on Very Large Corpora*, pages 82–94, Cambridge, MA, USA. Association for Computational Linguistics.

Rapp, R. (2002). The computation of word associations: Comparing syntagmatic and paradigmatic approaches. In *Proceedings of the COLING 2002: The 19th International Conference on Computational Linguistics*, Taipei, Taiwan. Association for Computational Linguistics.

Rapp, R. (2004). A freely available automatically generated thesaurus of related words. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC 2004)*, pages 395–398, Lisbon, Portugal. European Language Resources Association.

Rasmussen, C. (2000). The infinite Gaussian mixture model. In Solla, S. A., Leen, T. K., and Müller, K.-R., editors, *Advances in Neural Information Processing Systems 12*, pages 554–560. MIT Press, Cambridge, MA, USA.

Ratnaparkhi, A. (1996). A maximum entropy model for part-of-speech tagging. In Brill, E. and Church, K., editors, *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-96)*, pages 133–142, Philadelphia, PA, USA. Association for Computational Linguistics.

Rescorla, R. (2007). Rescorla–Wagner model. *Scholarpedia*, 2(3):2237.

Ries, K., Buo, F. D., and Wang, Y.-Y. (1995). Improved language modelling by unsupervised acquisition of structure. In *Proceedings of the 1995 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 1, pages 193–196, Detroit, MI, USA. IEEE.

Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, 14:465–471.

Rissanen, J. (1983). Universal prior for integers and estimation by minimum description length. *The Annals of Statistics*, 11:416–431.

Rissanen, J. (1989). *Stochastic Complexity in Statistical Inquiry*. World Scientific Publishing, New Jersey.

Rissanen, J. (1996). Fisher information and stochastic complexity. *IEEE Transactions on Information Theory*, 42:40–47.

Ristad, E. S. and Thomas, R. G. (1995). New techniques for context modeling. In Uszkoreit, H., editor, *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 220–227, Cambridge, MA, USA. Morgan Kaufmann Publishers / Association for Computational Linguistics.

Ritter, H. and Kohonen, T. (1989). Self-organizing semantic maps. *Biological Cybernetics*, 61:241–254.

Roark, B. (2001). Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2):249–276.

Roark, B., Saraçlar, M., and Collins, M. (2007). Discriminative n-gram language modeling. *Computer Speech and Language*, 21(2):373–392.

Roark, B., Saraçlar, M., Collins, M., and Johnson, M. (2004). Discriminative language modeling with conditional random fields and the perceptron algorithm. In Scott, D., Daelemans, W., and Walker, M. A., editors, *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 47–54, Barcelona, Spain.

Roark, B. and Sproat, R. (2007). *Computational approaches to morphology and syntax*. Oxford surveys in syntax and morphology. Oxford University Press, New York, NY, USA.

Robertson, S. and Zaragoza, H. (2009). The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389.

Rosch, E. (1973). Natural categories. *Cognitive Psychology*, 4:328–350.

Rosch, E. (1975). Cognitive representations of semantic categories. *Journal of Experimental Psychology: General*, 104(3):192–233.

Rosenberg, A. and Hirschberg, J. (2007). V-measure: A conditional entropy-based external cluster evaluation measure. In Eisner, J., editor, *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 410–420, Prague, Czech Republic. Association for Computational Linguistics.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408.

Rosenfeld, R. (1994). A hybrid approach to adaptive statistical language modeling. In *Proceedings of the ARPA workshop on human language technology*, pages 76–81.

Rosenfeld, R. (1996). A maximum entropy approach to adaptive statistical language modelling. *Computer Speech and Language*, 10(3):187–228.

Rosenfeld, R., Chen, S. F., and Zhu, X. (2001). Whole-sentence exponential language models: a vechile for linguistic-statistical integration. *Computer Speech and Language*, 15:55–73.

Rumelhart, D. E. and McClelland, J. H. (1986). On learning past tenses of English verbs. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 2, pages 216–271. MIT Press, Cambridge, CA, USA.

Sadeniemi, M., Kettunen, K., Lindh-Knuutila, T., and Honkela, T. (2008). Complexity of European Union languages: A comparative approach. *Journal of Quantitative Linguistics*, 15(2):185–211.

Saeed, J. I. (1997). *Semantics*. Blackwell Publishers Ltd, Oxford, UK.

Sahlgren, M. (2006a). Towards pertinent evaluation methodologies for word-space models. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC 2006)*, pages 821–824, Genoa, Italy. European Language Resources Association.

Sahlgren, M. (2006b). *The Word-Space Model*. PhD thesis, Department of Linguistics, Stockholm University, Stockholm, Sweden.

Sahlgren, M. and Karlgren, J. (2005). Automatic bilingual lexicon acquisition using random indexing of parallel corpora. *Natural Language Engineering*, 11(3):327–341.

Salton, G. and Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Prosessing and Management*, 24(5):513–523.

Sang, E. F. T. K. and Buchholz, S. (2000). Introduction to the CoNLL-2000 shared task. In Cardie, C., Daelemans, W., Nédellec, C., and Sang, E. T. K., editors, *Proceedings of the 2nd Workshop on Learning Language in Logic and the 4th Conference on Computational Natural Language Learning*, pages 127–132, Lisbon, Portugal. Association for Computational Linguistics.

Sapir, E. (1921). *Language: an introduction to the study of speech*. Harcourt, Brace; Bartleby.com (2000), New York, NY, USA. On-line: `www.bartleby.com/186/`. Retrieved 2011-07-08.

Sarikaya, R., Afify, M., Deng, Y., Erdoğan, H., and Gao, Y. (2008). Joint morphological-lexical language modeling for processing morphologically rich languages with application to dialectal Arabic. *IEEE Transactions on Audio Speech and Language Processing*, 16(7):1330–1340.

Saul, L. and Pereira, F. (1997). Aggregate and mixed-order markov models for statistical language processing. In Cardie, C. and Weischedel, R., editors, *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing (EMNLP-97)*, pages 81–89, New Providence, RI, USA. Association for Computational Linguistics.

Schone, P. and Jurafsky, D. (2000). Knowledge-free induction of morphology using latent semantic analysis. In Cardie, C., Daelemans, W., Nédellec, C., and Sang, E. T. K., editors, *Proceedings of the 2nd Workshop on Learning Language in Logic and the 4th Conference on Computational Natural Language Learning*, pages 67–72, Lisbon, Portugal. Association for Computational Linguistics.

Schone, P. and Jurafsky, D. (2001). Knowledge-free induction of inflectional morphologies. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics*, Pittsburgh, PA, USA. Association for Computational Linguistics.

Schönefeld, D. (2006). Constructions. *Constructions*, SV1(1):39.

Schütze, H. (1992). Dimensions of meaning. In *Proceedings of the 1992 ACM/IEEE Conference on Supercomputing (SC 1992)*, pages 787–796, Minneapolis, MN, USA. IEEE Computer Society.

Schütze, H. (1995). Distributional part-of-speech tagging. In *Proceedings of the 7th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 141–148, Dublin, Ireland. Association for Computational Linguistics.

Schütze, H., Hull, D. A., and Pedersen, J. O. (1995). A comparison of classifiers and document representations for the routing problem. In Fox, E. A., Ingwersen, P., and Fidel, R., editors, *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '95)*, pages 229–237, Seattle, WA, USA. ACM.

Schwarz, G. E. (1978). Estimating the dimension of a model. *Annals of Statistics*, 6(2):461–464.

Schwenk, H. (2007). Continuous space language models. *Computer Speech and Language*, 21:492–518.

Schwenk, H. and Gauvain, J.-L. (2002). Connectionist language modeling for large vocabulary continuous speech recognition. In *Proceedings of the 2002 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 765–768, Orlando, FL, USA. IEEE.

Scott, S. and Matwin, S. (1999). Feature engineering for text classification. In Bratko, I. and Dzeroski, S., editors, *Proceedings of the Sixteenth International Conference on Machine Learning (ICML 1999)*, pages 379–388, Bled, Slovenia. Morgan Kaufmann Publishers.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47.

Seginer, Y. (2007). Fast unsupervised incremental parsing. In Zaenen, A. and van den Bosch, A., editors, *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 384–391, Prague, Czech Republic. Association for Computational Linguistics.

Sereewattana, S. (2003). Unsupervised segmentation for statistical machine translation. Master's thesis, University of Edinburgh, Edinburgh, UK.

Sereno, J. A. and Jongman, A. (1997). Processing of English inflectional morphology. *Memory & Cognition*, 25:425–437.

Seshadri, N. and Sundberg, C.-E. W. (1994). List Viterbi decoding algorithms with applications. *IEEE Transactions on Communications*, 42(234):313 –323.

Seymore, K. and Rosenfeld, R. (1996). Scalable backoff language models. In *Proceedings of Fourth International Conference Spoken Language Processing (ICSLP)*, volume 1, pages 232–235, Philadelphia, PA, USA. ISCA.

Sha, F. and Pereira, F. (2003). Shallow parsing with conditional random fields. In Hearst, M. and Ostendorf, M., editors, *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2003)*, pages 134–141, Edmonton, Canada. Association for Computational Linguistics.

Shalonova, K., Golenia, B., and Flach, P. (2009). Towards learning morphology for under-resourced fusional and agglutinating languages. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(5):956–965.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656.

Shawe-Taylor, J. and Cristianini, N. (2004). *Kernel Methods for Pattern Analysis*. Cambridge University Press.

Shen, D., Sun, J.-T., Yang, Q., and Chen, Z. (2006). Text classification improved through multigram models. In Yu, P. S., Tsotras, V. J., Fox, E. A., and Liu, B., editors, *Proceedings of the 15th ACM International Conference on Information and Knowledge Management (CIKM 2006)*, pages 672–681, Arlington, VA, USA. ACM.

Shieber, S. (1985). Evidence against the context-freeness of human language. *Linguistics and Philosophy*, 8:333–343.

Shtarkov, Y. M. (1987). Universal sequential coding of single messages. *Problemy Peredachi Informatsii*, 23(3):3–17. English translation: *Problems of Information Transmission*, 23(3):175–186.

Siivola, V. and Honkela, A. (2003). A state-space method for language modeling. In *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 548–553, Cancun, Mexico. IEEE.

Siivola, V. and Pellom, B. L. (2005). Growing an n-gram language model. In *Proceedings of the 9th European Conference on Speech Communication and Technology (INTERSPEECH'05)*, pages 1309–1312, Lisbon, Portugal. ISCA.

Sim, K. C., Byrne, W. J., Gales, M. J. F., Sahbi, H., and Woodland, P. C. (2007). Consensus network decoding for statistical machine translation. In *Proceesdings of the 2007 IEEE Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 4, pages 105–108, Honolulu, HI, USA. IEEE.

Singh, S. P., Kearns, M. J., Litman, D. J., and Walker, M. A. (1999). Reinforcement learning for spoken dialogue systems. In Tolla, S. A., Leen, T. K., and Müller, K.-R., editors, *Advances in Neural Information Processing Systems 12*, pages 956–962. The MIT Press, Cambridge, MA, USA.

Siu, M. and Ostendorf, M. (2000). Variable n-grams and extensions for conversational speech language modeling. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 8(1):63–75.

Snover, M. G., Jarosz, G. E., and Brent, M. R. (2002). Unsupervised learning of morphology using a novel directed search algorithm: Taking the first step. In Maxwell, M., editor, *Proceedings of the ACL-02 Workshop on Morphological and Phonological Learning*, pages 11–20, Philadelphia, PA, USA. Association for Computational Linguistics.

Snyder, B. and Barzilay, R. (2008a). Cross-lingual propagation for morphological analysis. In Fox, D. and Gomes, C. P., editors, *Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI 2008)*, pages 848–854, Chicago, IL, USA. AAAI Press.

Snyder, B. and Barzilay, R. (2008b). Unsupervised multilingual learning for morphological segmentation. In Moore, J. D., Teufel, S., Allan, J., and Furui, S., editors, *Proceedings of ACL-08: HLT*, pages 737–745, Columbus, OH, USA. Association for Computational Linguistics.

Snyder, B. and Barzilay, R. (2010). Climbing the Tower of Babel: Unsupervised multilingual learning. In Fürnkranz, J. and Joachims, T., editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 29–36, Haifa, Israel. Omnipress.

Solan, Z., Horn, D., Ruppin, E., and Edelman, S. (2005). Unsupervised learning of natural languages. *Proceedings of the National Academy of Sciences*, 102(33):11639–11634.

Solomonoff, R. (1964a). A formal theory of inductive inference part i. *Information and Control*, 7(1):1–22.

Solomonoff, R. (1964b). A formal theory of inductive inference part ii. *Information and Control*, 7(2):224–254.

Soong, F. K. and Huang, E.-F. (1991). A tree-trellis based fast search for finding the N-best sentence hypotheses in continuous speech recognition. In *Proceedings of the 1991 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1, pages 705–708, Toronto, Canada.

Soveri, A., Lehtonen, M., and Laine, M. (2007). Word frequency and morphological processing revisited. *The Mental Lexicon*, 2:359–385.

Spärck Jones, K. (1972). A statistical interpretation of term specifity and its application in retrieval. *Journal of Documentation*, 28(1):11–21.

Spearman, C. (1904). The proof and measurement of association between two things. *The American Journal of Psychology*, 15:72–101.

Spiegler, S. (2011). *Machine Learning for the Analysis of Morphologically Complex Languages*. PhD thesis, Merchant Venturers School of Engineering, University of Bristol.

Spiegler, S., Golénia, B., and Flach, P. A. (2010a). Deap: Deductive-abductive parsing for morphological analysis. In Kurimo, M., Virpioja, S., and Turunen, V. T., editors, *Proceedings of the Morpho Challenge 2010 Workshop*, pages 44–48, Espoo, Finland. Aalto University School of Science and Technology, Department of Information and Computer Science. Technical Report TKK-ICS-R37.

Spiegler, S., Golénia, B., and Flach, P. A. (2010b). Unsupervised word decomposition with the promodes algorithm. In *Multilingual Information Access Evaluation I. Text Retrieval Experiments: 10th Workshop of the Cross-Language Evaluation Forum, CLEF 2009, Corfu, Greece, September 30 – October 2, 2009, Revised Selected Papers*, volume 6241 of *Lecture Notes in Computer Science*, pages 625–632. Springer, Berlin / Heidelberg, Germany.

Spiegler, S., Golénia, B., and Flach, P. A. (2010c). Word decomposition with the promodes algorithm family bootstrapped on a small labelled dataset. In Kurimo, M., Virpioja, S., and Turunen, V. T., editors, *Proceedings of the Morpho Challenge 2010 Workshop*, pages 49–52, Espoo, Finland. Aalto University School of Science and Technology, Department of Information and Computer Science. Technical Report TKK-ICS-R37.

Spiegler, S. and Monson, C. (2010). EMMA: A novel evaluation metric for morphological analysis. In Huang, C.-R. and Jurafsky, D., editors, *Proceedings of the 23rd International Conference on Computational Linguistics (COLING)*, pages 1029–1037, Beijing, China. Coling 2010 Organizing Committee.

Spitkovsky, V. I., Alshawi, H., and Jurafsky, D. (2010a). From baby steps to leapfrog: How "less is more" in unsupervised dependency parsing. In Kaplan, R., Burstein, J., Harper, M., and Penn, G., editors, *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 751–759, Los Angeles, CA, USA. Association for Computational Linguistics.

Spitkovsky, V. I., Alshawi, H., and Jurafsky, D. (2011). Lateen em: Unsupervised training with multiple objectives, applied to dependency grammar induction. In Barzilay, R. and Johnson, M., editors, *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1269–1280, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Spitkovsky, V. I., Alshawi, H., Jurafsky, D., and Manning, C. D. (2010b). Viterbi training improves unsupervised dependency parsing. In Lapata, M. and Sarkar, A., editors, *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 9–17, Uppsala, Sweden. Association for Computational Linguistics.

Sproat, R. and Emerson, T. (2003). The first international Chinese word segmentation bakeoff. In Ma, Q. and Xia, F., editors, *Proceedings of the Second SIGHAN Workshop on Chinese Language Processing*, volume 17, pages 133–143, Sapporo, Japan. Association for Computational Linguistics.

Sproat, R., Gales, W., Shih, C., and Chang, N. (1996). A stochastic finite-state word-segmentation algorithm for Chinese. *Computational Linguistics*, 22(3):377–404.

Steels, L. (2004). Constructivist development of grounded construction grammar. In Scott, D., Daelemans, W., and Walker, M. A., editors, *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 9–16, Barcelona, Spain. Association for Computational Linguistics.

Steyvers, M., Shiffrin, R. M., and Nelson, D. L. (2005). Word association spaces for predicting semantic similarity effects in episodic memory. In Healy, A. F., editor, *Experimental Cognitive Psychology and Its Applications*, pages 237–249. American Psychological Association, Washington, DC, USA.

Stolcke, A. (1995). An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics*, 21(2):165–201.

Stolcke, A. (1998). Entropy-based pruning of backoff language models. In *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, pages 270–274.

Stolcke, A. and Segal, J. (1994). Precise n-gram probabilities from stochastic context-free grammars. In Pustejovsky, J., editor, *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 74–79, Las Cruces, NM, USA. Association for Computational Linguistics.

Stump, G. T. (2001). *Inflectional Morphology: A Theory of Paradigm Structure*. Cambridge University Press, Cambridge, UK.

Sun, W. and Xu, J. (2011). Enhancing Chinese word segmentation using unlabeled data. In Barzilay, R. and Johnson, M., editors, *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 970–979, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Sutton, C. and McCallum, A. (2005). Composition of conditional random fields for transfer learning. In Mooney, R., Brew, C., Chien, L.-F., and Kirchhoff, K., editors, *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 748–754, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Taft, M. (1979). Recognition of affixed words and the word frequency effect. *Memory and Cognition*, 7:263–272.

Tam, Y.-C. and Schultz, T. (2005). Dynamic language model adaptation using variational Bayes inference. In *Proceedings of the 9th European Conference on Speech Communication and Technology (Interspeech 2005)*, pages 5–8, Lisbon, Portugal. ISCA.

Tchoukalov, T., Monson, C., and Roark, B. (2010). Morphological analysis by multiple sequence alignment. In *Multilingual Information Access Evaluation I. Text Retrieval Experiments: 10th Workshop of the Cross-Language Evaluation Forum, CLEF 2009, Corfu, Greece, September 30 – October 2, 2009, Revised Selected Papers*, volume 6241 of *Lecture Notes in Computer Science*, pages 666–673. Springer, Berlin / Heidelberg, Germany.

Teahan, W. J., McNab, R., Wen, Y., and Witten, I. H. (2000). A compression-based algorithm for Chinese word segmentation. *Computational Linguistics*, 26(3):375–393.

Teh, Y. W. (2006). A hierarchical Bayesian language model based on Pitman-Yor processes. In Calzolari, N., Cardie, C., and Isabelle, P., editors, *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 985–992, Sydney, Australia. Association for Computational Linguistics.

Tepper, M. and Xia, F. (2008). A hybrid approach to the induction of underlying morphology. In Lee, J.-H., Copestake, A., and Matsumoto, Y., editors, *Proceedings of the Third International Joint Conference on Natural Language Processing (IJCNLP 2008)*, volume 1, Hyderabad, India. Asian Federation of Natural Language Processing.

Theron, P. and Cloete, I. (1997). Automatic acquisition of two-level morphological rules. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 103–110, Washington, DC, USA. Association for Computational Linguistics.

Thrun, S. (1995). Is learning the n-th thing any easier than learning the first? In Touretzky, D. S., Mozer, M., and Hasselmo, M. E., editors, *Advances in Neural Information Processing Systems 8*, pages 640–646. MIT Press, Cambridge, MA, USA.

Tiedemann, J. (2009). News from OPUS — A collection of multilingual parallel corpora with tools and interfaces. In Nicolov, N., Bontcheva, K., Angelova, G., and Mitkov, R., editors, *Recent Advances in Natural Language Processing V, Selected papers from RANLP 2007*, pages 237–248. John Benjamins, Amsterdam / Philadelphia.

Tripathi, A. (2011). *Data fusion and matching by maximizing statistical dependencies*. PhD thesis, University of Helsinki, Helsinki, Finland.

Tripathi, A., Klami, A., and Kaski, S. (2008). Using dependencies to pair samples for multi-view learning. TKK Reports in Information and Computer Science TKK-ICS-R8, Helsinki University of Technology, Espoo, Finland.

Tripathi, A., Klami, A., and Virpioja, S. (2010). Bilingual sentence matching using kernel CCA. In *Proceedings of the 2010 IEEE International Workshop on Machine Learning for Signal Processing (MLSP 2010)*, pages 130–135, Kittilä, Finland. IEEE.

Tromble, R., Kumar, S., Och, F., and Macherey, W. (2008). Lattice Minimum Bayes-Risk decoding for statistical machine translation. In Lapata, M. and Ng, H. T., editors, *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 620–629, Honolulu, HI, USA. Association for Computational Linguistics.

Tron, V., Gyepesi, G., Halácsky, P., Kornai, A., Németh, L., and Varga, D. (2005). Hunmorph: Open source word analysis. In Jansche, M., editor, *Proceedings of the ACL Workshop on Software*, pages 77–85, Ann Arbor, MI, USA. Association for Computational Linguistics.

Turchi, M., Bie, T. D., Goutte, C., and Cristianini, N. (2012). Learning to translate: A statistical and computational analysis. *Advances in Artificial Intelligence*, 2012(484580):15.

Turing, A. M. (1936). On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society. Second Series*, 42:230–265.

Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, 59:433–460.

Turney, P. (2002). Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 417–424, Philadelphia, PA, USA. Association for Computational Linguistics.

Turney, P. D. (2001). Mining the Web for synonyms: PMI-IR versus LSA on TOEFL. In Raedt, L. D. and Flach, P. A., editors, *Proceedings of the Twelth European Conference on Machine Learning (ECML-2001)*, volume 2167 of *Lecture Notes in Computer Science*, pages 491–502. Springer-Verlag, Berlin / Heidelberg, Germany.

Turney, P. D. (2005). Measuring semantic similarity by latent relational analysis. In Kaelbling, L. P. and Saffiotti, A., editors, *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-05)*, pages 1136–1141, Edinburgh, UK. International Joint Conferences on Artificial Intelligence.

Turunen, V. T. and Kurimo, M. (2011). Speech retrieval from unsegmented Finnish audio using statistical morpheme-like units for segmentation, recognition, and retrieval. *ACM Transactions on Speech and Language Processing*, 8(1):1–25.

van den Bosch, A., Stroppa, N., and Way, A. (2007). A memory-based classification approach to marker-based ebmt. In Eynde, F. V., Vandeghinste, V., and Schuurman, I., editors, *Proceedings of the METIS-II Workshop on New Approaches to Machine Translation*, pages 63–72, Leuven, Belgium. Katholieke Universiteit Leuven. On-line: `http://www.ccl.kuleuven.be/ws-metis/program.php`. Retrieved 2012-06-01.

van Zaanen, M. (2000). ABL: Alignment-based learning. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000)*, volume 1, pages 961–967, Saarbrücken, Germany. DFKI GmbH.

Vapnik, V. N. (1999). *The nature of statistical learning theory*. Springer, New York, NY, USA, 2nd edition.

Vatanen, T., Väyrynen, J. J., and Virpioja, S. (2010). Language identification of short text segments with n-gram models. In Calzolari, N., Choukri, K., Maegaard, B., Mariani, J., Odjik, J., Piperidis, S., Rosner, M., and Tapias, D., editors, *Proceedings of the Seventh Conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association.

Väyrynen, J. J., Lindqvist, L., and Honkela, T. (2007). Sparse distributed representations for words with thresholded independent component analysis. In Si, J. and Sun, R., editors, *Proceedings of the International Joint Conference on Neural Networks (IJCNN 2007)*, pages 1031–1036, Orlando, FL, USA. IEEE.

Vigliocco, G., Vinson, D. P., Druks, J., Barber, H., and Cappa, S. F. (2011). Nouns and verbs in the brain: A review of behavioural, electrophysiological, neuropsychological and imaging studies. *Neuroscience and Biobehavioral Reviews*, 35:407–426.

Vinokourov, A., Shawe-Taylor, J., and Cristianini, N. (2003). Inferring a semantic representation of text via cross-language correlation analysis. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15*, pages 1497–1504. MIT Press, Cambridge, MA, USA.

Vitanyi, P. M. B. and Li, M. (2000). Minimum description length induction, Bayesianism, and Kolmogorov complexity. *IEEE Transactions on Information Theory*, 46(2):446–464.

Viterbi, A. J. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269.

Wallace, C. S. and Boulton, D. M. (1968). An information measure for classification. *Computer Journal*, 11(2):185–194.

Wallace, C. S. and Dowe, D. L. (1999). Minimum message length and kolmogorov complexity. *Computer Journal*, 42(4):270–283.

Wallace, C. S. and Freeman, P. R. (1987). Estimation and inference by compact coding. *Journal of the Royal Statistical Society. Series B (Methodological)*, 49(3):240–265.

Wang, M. Q. and Hirschberg, J. (1992). Automatic classification of intonational phrase boundaries. *Computer Speech and Language*, 6:175–196.

Ward, G. (2002). Moby thesaurus list. On-line: `http://www.gutenberg.org/ebooks/3202`. Retrieved 2012-02-02.

Watts, O., Yamagishi, J., and King, S. (2011). Unsupervised continuous-valued word features for phrase-break prediction without a part-of-speech tagger. In *Proceedings of the 12th Annual Conference of the International Speech Communication Association (INTERSPEECH 2011)*, pages 2157–2160, Florence, Italy. ISCA.

Wells, R. S. (1947). Immediate constituents. *Language*, 23(2):81–117.

Westerweld, T., de Vries, A., and de Jong, F. (2007). Generative probabilistic models. In Blanken, H. M., de Vries, A. P., Blok, H. E., and Feng, L., editors, *Multimedia retrieval*, pages 177–198. Springer-Verlag, Berlin / Heidelberg, Germany.

Wicentowski, R. (2004). Multilingual noise-robust supervised morphological analysis using the wordframe model. In *Current Themes in Computational Phonology and Morphology: Proceedings of Seventh Meeting of the ACL Special Interest Group on Computational Phonology (SIGPHON)*, pages 70–77, Barcelona, Spain. Association for Computational Linguistics.

Wilcox, B. (2011). Beyond façade: Pattern matching for natural language applications. On-line: `http://www.gamasutra.com/view/feature/6305/beyond_fa%C3%A7ade_pattern_matching_.php`. Retrieved 2011-06-21.

Wintner, S. (2009). What science underlies natural language engineering? *Computational Linguistics*, 35(4):641–644.

Witten, I. H. and Bell, T. C. (1991). The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory*, 37(4):1085–1094.

Wood, F. and Teh, Y. W. (2009). A hierarchical nonparametric Bayesian approach to statistical language model domain adaptation. In van Dyk, D. and Welling, M., editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS 2009)*, volume 5 of *JMLR: Workshop and Conference Proceedings*, pages 607–614. Journal of Machine Learning Research.

Wu, J. and Khudanpur, S. (2002). Building a topic-dependent maximum entropy model for very large corpora. In *Proceedings of the 2002 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1, pages 777–780, Orlando, FL, USA. IEEE.

Wu, Z. and Tseng, G. (1993). Chinese text segmentation for text retrieval: Achievements and problems. *Journal of the American Society for Information Science*, 44(9):532–542.

Xu, P. and Jelinek, F. (2004). Random forests in language modeling. In Lin, D. and Wu, D., editors, *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 325–332, Barcelona, Spain. Association for Computational Linguistics.

Yamamoto, H., Isogai, S., and Sagisaka, Y. (2003). Multi-class composite n-gram language model. *Speech Communication*, 41(2–3):369–379.

Yang, M. and Kirchhoff, K. (2006). Phrase-based backoff models for machine translation of highly inflected languages. In McCarthy, D. and Wintner, S., editors, *11th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2006)*, pages 41–48, Trento, Italy. Association for Computational Linguistics.

Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. In Uszkoreit, H., editor, *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, Cambridge, MA, USA. Association for Computational Linguistics.

Yarowsky, D. and Florian, R. (2002). Evaluating sense disambiguation across diverse parameter spaces. *Natural Language Engineering*, 8(4):293–310.

Yarowsky, D., Ngai, G., and Wicentowski, R. (2001). Inducing multilingual text analysis tools via robust projection across aligned corpora. In Allan, J., editor, *Proceedings of the First International Conference on Human Language Technology Research (HLT 2001)*, pages 161–168, San Diego, CA, USA. Morgan Kaufmann.

Yarowsky, D. and Wicentowski, R. (2000). Minimally supervised morphological analysis by multimodal alignment. In *Proceedings of the 38th Meeting of the ACL*, pages 207–216, Hong Kong, China. Association for Computational Linguistics.

Younger, D. H. (1967). Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10(2):189–208.

Zelikovitz, S. and Hirsh, H. (2001). Improving text classification with LSI using background knowledge. In Nebel, B., editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI01)*, pages 113–118, Seattle, WA, USA. International Joint Conferences on Artificial Intelligence Organization, Morgan Kaufmann Publishers.

Zeman, D. (2008). Unsupervised acquiring of morphological paradigms from tokenized text. In *Advances in Multilingual and Multimodal Information Retrieval, 8th Workshop of the Cross-Language Evaluation Forum, CLEF 2007, Budapest, Hungary, September 19–21, 2007, Revised Selected Papers*, volume 5152 of *Lecture Notes in Computer Science*, pages 892–899. Springer, Berlin / Heidelberg, Germany.

Zeman, D. (2009). Using unsupervised paradigm acquisition for prefixes. In *Evaluating Systems for Multilingual and Multimodal Information Access, 9th Workshop of the Cross-Language Evaluation Forum, CLEF 2008, Aarhus, Denmark, September 17–19, 2008, Revised Selected Papers*, volume 5706 of *Lecture Notes in Computer Science*, pages 983–990. Springer, Berlin / Heidelberg, Germany.

Zesch, T. and Gurevych, I. (2009). Wisdom of crowds versus wisdom of linguists — measuring the semantic relatedness of words. *Natural Language Engineering*, 16(1):25–59.

Zhang, D., Mei, Q., and Zhai, C. (2010). Cross-lingual latent topic extraction. In Hajič, J., Carberry, S., Clark, S., and Nivre, J., editors, *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1128–1137, Uppsala, Sweden. Association for Computational Linguistics.

Zhu, X. (2005). Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison.

Zipf, G. K. (1932). *Selective Studies and the Principle of Relative Frequency in Language*. Harvard University Press, Cambridge, MA.

Zipf, G. K. (1949). *Human Behavior and the Principle of Least Effort*. Addison-Wesley, Boston, MA, USA.

Zollmann, A., Venugopal, A., and Vogel, S. (2006). Bridging the inflection morphology gap for Arabic statistical machine translation. In Moore, R. C., Bilmes, J., Chu-Carroll, J., and Sanderson, M., editors, *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 201–204, New York, NY, USA. Association for Computational Linguistics.

DISSERTATIONS IN INFORMATION AND COMPUTER SCIENCE

Aalto-DD19/2012 Reunanen, Juha
Overfitting in Feature Selection: Pitfalls and Solutions. 2012.

Aalto-DD33/2012 Caldas, José
Graphical Models for Biclustering and Information Retrieval in Gene Expression Data. 2012.

Aalto-DD45/2012 Viitaniemi, Ville
Visual Category Detection: an Experimental Perspective. 2012.

Aalto-DD51/2012 Hanhijärvi, Sami
Multiple Hypothesis Testing in Data Mining. 2012.

Aalto-DD56/2012 Ramkumar, Pavan
Advances in Modeling and Characterization of Human Neuromagnetic Oscillations. 2012.

Aalto-DD97/2012 Turunen, Ville T.
Morph-Based Speech Retrieval: Indexing Methods and Evaluations of Unsupervised Morphological Analysis. 2012.

Aalto-DD115/2012 Vierinen, Juha
On statistical theory of radar measurements. 2012.

Aalto-DD117/2012 Huopaniemi, Ilkka
Multivariate Multi-Way Modelling of Multiple High-Dimensional Data Sources. 2012.

Aalto-DD137/2012 Paukkeri, Mari-Sanna
Language- and domain-independent text mining. 2012.

Aalto-DD133/2012 Ahlroth, Lauri
Online Algorithms in Resource Management and Constraint Satisfaction. 2012.

The digital revolution means that there are increasing amounts of text and speech material in electronic format. This calls for adaptive and data-driven approaches to automatically process the continuously accumulating data. A particular problem in language processing is how to select the lexical units (such as words, morphemes, and phrases) for further modeling in information retrieval, speech recognition, and machine translation systems. The systems themselves rely more and more on machine learning techniques. However, the unit selection problem is still commonly solved by traditional rule-based approaches that are limited in languages and domains. Building on statistical machine learning methods and the linguistic theory of construction grammars, this thesis presents new unsupervised and semi-supervised algorithms for selecting lexical units. It also presents new evaluation methods for the units learned and examines various approaches for utilizing them in speech recognition and statistical machine translation.

Aalto-DD 158/2012

ISBN 978-952-60-4882-6
ISBN 978-952-60-4883-3 (pdf)
ISSN-L 1799-4934
ISSN 1799-4934
ISSN 1799-4942 (pdf)

Aalto University
School of Science
Department of Information and Computer Science
www.aalto.fi