Teemu Hirsimäki, Mathias Creutz, Vesa Siivola, Mikko Kurimo, Sami Virpioja, and Janne Pylkkönen. 2006. Unlimited vocabulary speech recognition with morph language models applied to Finnish. Computer Speech and Language, volume 20, number 4, pages 515-541.

Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computer Speech and Language 20 (2006) 515–541
COMPUTER SPEECH AND LANGUAGE
www.elsevier.com/locate/csl

Unlimited vocabulary speech recognition with morph language models applied to Finnish

Teemu Hirsimäki *, Mathias Creutz, Vesa Siivola, Mikko Kurimo, Sami Virpioja, Janne Pylkkönen

Helsinki University of Technology, Neural Networks Research Centre, P.O. Box 5400, Konemiehentie 2, 02015 Espoo HUT, Finland

Received 18 June 2004; received in revised form 21 December 2004; accepted 31 July 2005
Available online 29 August 2005

Abstract

In the speech recognition of highly inflecting or compounding languages, the traditional word-based language modeling is problematic. As the number of distinct word forms can grow very large, it becomes difficult to train language models that are both effective and cover the words of the language well. In the literature, several methods have been proposed for basing the language modeling on sub-word units instead of whole words. However, to our knowledge, considerable improvements in speech recognition performance have not been reported.

In this article, we present a language-independent algorithm for discovering word fragments in an unsupervised manner from text. The algorithm uses the Minimum Description Length principle to find an inventory of word fragments that is compact but models the training text effectively. Language modeling and speech recognition experiments show that *n*-gram models built over these fragments perform better than *n*-gram models based on words. In two Finnish recognition tasks, relative error rate reductions between 12% and 31% are obtained. In addition, our experiments suggest that word fragments obtained using grammatical rules do not outperform the fragments discovered from text. We also present our recognition system and discuss how utilizing fragments instead of words affects the decoding process.

* Corresponding author. Tel.: +358 9 4513284; fax: +358 9 4513277.
E-mail address: teemu.hirsimaki@hut.fi (T. Hirsimäki).

0885-2308/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.csl.2005.07.002

1. Introduction

In certain natural languages, there are interesting key problems that have not been much studied in developing large vocabulary continuous speech recognition (LVCSR) for English. One major problem is related to the number of distinct word forms that appear in every-day use. The conventional way of building statistical language models has been to collect co-occurrence statistics on words, such as *n*-grams. If the language can be covered well by a lexicon of reasonable size, it is possible to train statistical models using available toolkits, given enough training data and computational resources.

In many languages, however, the word-based approach has some disadvantages. In highly inflecting languages, such as Finnish and Hungarian, there may be thousands of different word forms of the same root, which makes the construction of a fixed lexicon for any reasonable coverage hardly feasible. Also in compounding languages, such as German, Swedish, Greek and Finnish, complex concepts can be expressed in a single word, which considerably increases the number of possible word forms. This leads to data sparsity problems in *n*-gram language modeling.

1.1. Related work

During the recent years, some approaches have been proposed to deal with the problem of vocabulary growth in large vocabulary speech recognition for different languages. Geutner et al. (1998) presented a two-pass recognition approach for increasing the vocabulary adaptively. In the first pass, a traditional word lexicon was used to create a word lattice for each speech segment, and before the second pass, the inflectional forms of the words were added to the lattice. In a Serbo-Croatian task, they reported word accuracy improvement from 64.0% to 69.8%. McTait and Adda-Decker (2003), on the other hand, reported that the recognition performance in a German task could be improved by increasing the lexicon size. The use of a lexicon of 300,000 words instead of 60,000 words lowered the word error rate from 20.4% to 18.5%.

Factored language models (Bilmes and Kirchhoff, 2003; Kirchhoff et al., 2003) have recently been proposed for incorporating morphological knowledge in the modeling of inflecting languages. Instead of conditioning probabilities on a few preceding words, the probabilities are conditioned on sets of features derived from words. These features (or factors) can include, for example, morphological, syntactic and semantic information. Vergyri et al. (2004) present experiments on Arabic speech recognition and report minor word error rate reductions.

Another promising direction has been to abandon words as the basic units of language modeling and speech recognition. As prefixes, suffixes and compound words are the cause of the growth of the vocabulary in many languages, a logical idea is to split the words into shorter units. Then the language modeling and recognition can be based on these word fragments. Several approaches have been proposed for different languages, and perplexity reductions have been achieved, but few have reported clear recognition improvements. Byrne et al. (2000) used a morphological analyzer for Czech to split words in stems and endings. A language model based on a vocabulary of 9600 morphemes gave better results when compared to a model based on a vocabulary of 20,000 words. However, with larger vocabularies (61,000 words and 25,000 morphemes), the word based models performed better (Byrne et al., 2001). Kwon and Park (2003) also used a morphological analyzer

to obtain morphemes for a Korean recognition task. They reported that merging short morphemes together improved results. Szarvas and Furui (2003) used an analyzer to get morphemes for a Hungarian task. Additionally, morphosyntactic rules were incorporated into the model allowing only grammatical morpheme combinations. Relative morpheme error reductions between 1.7% and 7.2% were obtained.

In contrast to using a morphological analyzer, data-driven algorithms for splitting words in smaller units have also been investigated in speech recognition. Whittaker and Woodland (2000) proposed an algorithm for segmenting a text corpus into fragments that maximize the 2-gram likelihood of the segmented corpus. Small improvements in error rates (2.2% relative) were obtained in an English recognition task when the sub-word model was interpolated with a traditional word-based 3-gram model. Ordelman et al. (2003) presented a method for decomposing Dutch compound words automatically, and reported minor improvements in error rates.

To our knowledge, there is little previous work on basing the language modeling and recognition on sub-word units for Finnish LVCSR. Kneissler and Klakow (2001) segmented a corpus into word fragments that maximize the 1-gram likelihood of the corpus. Four different segmentation strategies were compared in a Finnish dictation task. The strategies required various amounts of input from an expert of the Finnish language. However, no comparisons to traditional word models were performed.

There are a number of works that aim at learning the morphology of a natural language in a fully unsupervised manner from data. Often words are assumed to consist of one stem typically followed by one suffix. Sometimes prefixes are possible. The work by Goldsmith (2001) exemplifies such an approach and gives a survey of the field. The morphologies discovered by these algorithms have not been applied in speech recognition. It seems that this kind of method is not suitable for agglutinative languages, such as Finnish, where words may consist of lengthy sequences of concatenated morphemes.

Morpheme-like units have also been discovered by algorithms for word segmentation, i.e., algorithms that discover word boundaries in text without blanks. Deligne and Bimbot (1997) derive a model structure that can be used both for word segmentation and for detecting variable-length acoustic units in speech data. Their data-driven units do not, however, produce as good results as conventional word models in recognizing the speech of French weather forecasts. Brent (1999) is mainly interested in the acquisition of a lexicon in an incremental fashion and applies his probabilistic model to the segmentation of transcripts of child-directed speech.

1.2. Contents of the article

In this work, we make use of word fragments in language modeling and speech recognition. To avoid using a huge word vocabulary consisting of hundreds of thousands of distinct word forms, we split the words into frequently occurring sub-word units. We present an algorithm that discovers such word fragments from a text corpus in a fully unsupervised manner. The fragment inventory, or lexicon, is optimized for the given corpus according to a model based on the information-theoretic Minimum Description Length (MDL) principle (Rissanen, 1989).[1] The resulting fragments are here

[1] For readers more familiar with probabilistic models, we note that our MDL model can also be formulated in the maximum a posteriori (MAP) framework. See Section 3.1.2.

referred to as *statistical morphs* as the boundaries of the fragments often coincide with grammatical morpheme boundaries.

The algorithm is motivated by the following features: The resulting model can cover the whole language obtaining a 0% out-of-vocabulary (OOV) rate by a reasonably sized but still apparently meaningful set of word fragments. The degree of word-splitting is influenced by the size of the training corpus, and foreign words are split as well, because no language-dependent assumptions are involved. A word can be split into a long sequence of fragments, which makes the model suitable for agglutinative languages. An earlier version of the method has already given good results in Finnish and Turkish recognition tasks (Siivola et al., 2003; Hacioglu et al., 2003).

In this article, we give a detailed description of the algorithm for segmenting a text corpus into statistical morphs, and compare the resulting language models with models based on two alternative methods. The other models are also capable of generating the whole language, albeit in a more simplistic manner: words augmented with phonemes, and fragments based on automatic grammatical analysis augmented with phonemes. The language modeling and recognition performance of *n*-gram models built using these units are evaluated in two Finnish tasks. We also discuss how the use of fragments affects the decoder of our speech recognition system.

In Section 2, we present the two Finnish tasks and performance measures used in the experiments. The central section of the paper is Section 3. It describes the statistical model and algorithm for segmenting a text corpus into word fragments. The alternative approaches for producing complete-coverage vocabularies are also presented with comparative cross-entropy experiments. Section 4 describes the recognition system. The acoustic models are presented briefly, the emphasis being on the duration models for Finnish phonemes, followed by the description of the decoder. The results of the experiments are given in Section 5 with discussion, and Section 6 concludes the work.

2. The Finnish evaluation task

This chapter describes the LVCSR task that we propose for evaluating the new language models for Finnish. The language research community for Finnish is rather small, and extensive text and speech corpora for language modeling and speech recognition research do not exist yet. However, the Finnish IT center for science has an ongoing project *Kielipankki* (*Language bank*) which collects Finnish and Swedish text and speech data that can be obtained for research purposes.[2] One of our aims was to use evaluations that could be easily utilized by other research groups to build and test comparable LVCSR systems.

2.1. Text and speech data

The large-vocabulary language models were trained using a text corpus of 40 million words. The main material is from the language bank described above, which was augmented by almost an equal amount of newswire text from the Finnish National News Agency.

[2] The project page of the Language bank is http://www.csc.fi/kielipankki/.

Two different speech data sets, not included in the language model training, were used for evaluating the recognition performance. For both data sets, acoustic models were trained and evaluated separately. As there is not yet enough Finnish speech data available to train proper speaker-independent models for this kind of tasks, we have considered only speaker-dependent tasks.

The first data set is a Finnish audio book[3] containing 12 h of speech from one female speaker. The first 11 h were used for training the acoustic models, and from the end of the book, 20 min were used for tuning the decoder parameters and 27 min for evaluation. The task is here referred to as BOOK.

The second speech data set, referred to as NEWS, consists of about 5 h of news reading by another female speaker. The content is divided into short newswire articles, where each article has its own characteristic topic. From this task, about 3.5 h were used for training the acoustic models, and 33 min for development and 49 min for evaluation.

In addition to training acoustic models, the reference transcriptions of the training portions of the BOOK and NEWS tasks were used to evaluate the cross-entropies of the language models reported in Section 3.6.

2.2. Phonetic transcriptions

The orthography of the Finnish language has a straightforward connection with the pronunciation. There is an almost one-to-one correspondence between letters and phonemes, with the exception of a few clusters of letters that correspond to one phoneme, such as double letters indicating a long sound. The splitting of words into fragments is thus rather unproblematic for speech recognition applications that need to reconstruct words from fragments and spell the words correctly.

However, foreign words, which are common especially in news data, are more problematic. In this work, we have utilized software to automatically produce satisfactory pronunciations for foreign names, and to expand numbers and abbreviations to complete written forms.[4] The reference transcriptions are also processed by the program, so currently foreign words are spelled as they would most likely be pronounced, and we leave it to a future version of our system to fully cope with the orthography.

2.3. Performance measures

2.3.1. Cross-entropy and perplexity

Because running extensive speech recognition tests to analyze the effect of all language model parameters takes too much time, it is common to first measure the language modeling accuracy for text data. This is often done by computing the probability of independent test data given by

[3] The audio version of the book *Syntymättömien sukupolvien Eurooppa* by Eero Paloheimo was kindly provided by the Finnish Federation of the Visually Impaired. In the near future, the book will be available in the Finnish language bank.

[4] We are grateful to Nicholas Volk from the University of Helsinki for kindly providing the software: http://www.ling.helsinki.fi/suopuhe/lavennin/.

the model, or derivative measures such as cross-entropy and perplexity (Chen and Goodman, 1999). It is important to notice that, if the language models operate on different units, such as different fragment inventories, cross-entropy and perplexity must be computed over a common symbol set, such as words for example, to allow for a fair comparison. See Section 3.6 for more details.

2.3.2. Recognition error rates

It is well known that a decrease in cross-entropy (or perplexity) does not necessarily lead to better recognition performance, so main conclusions should be based on the quality of the recognizer output. In speech recognition, the conventional error measure is the word error rate (WER), which is the proportion of all the deleted, added, and substituted words to the total number of words in the reference transcript. The WER is usually applied independent of the application for which the speech recognition is intended, although it obviously assumes that all the deletion, addition, and substitution errors are equally significant.

An example of more application-oriented recognition error measures is term error rate (TER). It is more suitable for measuring the quality of speech transcripts in speech retrieval applications (Johnson et al., 1999), because it concentrates on the more important content words. For dictation applications, the direct performance measure would be letter error rate (LER), because it best resembles the number of editing operations required to manually correct the text. In Finnish, the phoneme error rate (PHER) is close to LER as the letters in the written form of the words correspond almost directly to phonemes.

If the recognition is based on fragments of words, it is naturally possible to define an error rate for those units. However, if the evaluation concerns specifically the different ways of defining the fragments, it would not make sense to have separate measures for all of them. In this work, we use WER and PHER, but analyze mainly the latter which has a finer resolution than WER. Especially in Finnish, WER is perhaps not the most descriptive measure, since the words are often long, and making a tiny error in one of the suffixes of a long word renders the whole word erroneous.

3. Language modeling with data-driven units

In this section, we propose to solve the problem of large word vocabularies by producing a lexicon of word fragments and estimating n-gram language models over these fragments instead of entire words. Our algorithm learns a set of word fragments from a large text corpus or a corpus vocabulary in an unsupervised manner, and utilizes a model that is based on the MDL principle. The algorithm has obvious merits as it is not language-dependent and it relies on a model with a principled formulation, which takes the complexity of the model into account in addition to the likelihood of the data.[5] Furthermore, any word form can be constructed, which implies a 0% OOV rate for the model.

The word fragments produced by this algorithm will be referred to as *statistical morphs*. The choice of term reflects that the algorithm utilizes statistical criteria in selecting fragments into the lexicon. Moreover, when a word form is built by a concatenation of fragments, the boundaries

[5] Venkataraman (2001) exemplifies a *maximum likelihood* (ML) approach to word segmentation of transcribed speech. In ML estimation, the complexity of the model is not taken into account.

between the fragments frequently coincide with *morpheme boundaries*. The term *morph* is used in linguistics to denote a realization of a *morpheme*, which is the smallest meaningful unit of language. Morphs can be realizations of the same morpheme, e.g., English 'city' and 'citi' (in 'citi+es'), or the suffixes '-ssa' and '-ssä' in Finnish (having roughly the same meaning as the English preposition 'in'). Our algorithm does not, however, discover which morphs are realizations of the same morpheme.

To make comparisons, we have also constructed a recognition lexicon consisting of *grammatical morphs*. These morphs were produced with the help of software for automatic morphological analysis, based on a hand-made lexicon and rule-set. Additionally, we have tested traditional *word lexicons*. In these models, OOV words are problematic for different reasons. In the model based on grammatical morphs, some words are OOV, because they lack a morphological analysis in the hand-made description. In a model based on words, the lexicon can only hold a limited number of word forms. Thus, some words in the training corpus are left out and there will always be some perfectly valid word forms that were never observed in the available corpus. We suggest a simplistic solution to this problem by including all individual phonemes in the lexicon as word fragments allowing any OOV word to be composed of a concatenation of phonemes.

3.1. Statistical morphs

The original version of the algorithm for discovering statistical morphs was presented by Creutz and Lagus (2002), and it was called the *Recursive MDL* algorithm. The name reflects properties of the search heuristics and the model structure.[6] The algorithm, here slightly modified, learns a lexicon of morphs in an unsupervised manner from the same corpus that is used for training n-gram language models. The morph lexicon is constructed so that it is possible to form any word in the corpus by concatenating some morphs. We aim at finding an optimal lexicon and segmentation, i.e., a concise set of morphs giving a concise representation for the corpus. The source code for the algorithm is public (Creutz and Lagus, 2005).

3.1.1. Morph segmentation model

Following the MDL principle (Rissanen, 1989), the idea is to find an optimal encoding of a data set. That is, on the one hand, we choose a model and encode its parameters. On the other hand, we encode the data conditioned on the model. The optimal encoding is such that it gives the shortest total length, $L(x,\theta)$, of the code of the data, x, together with the code of the model parameters, θ

$$\arg\min_{\theta} L(x,\theta) = \arg\min_{\theta}[L(x|\theta) + L(\theta)]. \quad (1)$$

To be more concrete, in our case the data consist of a *corpus* or list of unsegmented words. The model is a *set of unique morphs* or a *morph lexicon*, where each morph is a string of characters and has a particular probability of occurrence. Now, each word in the corpus can be rewritten as a sequence of morph tokens. This can be thought of as encoding the corpus as a sequence of morph pointers, which point to entries in the morph lexicon. Our aim is to find the combination of a

[6] Argamon et al. (2004) propose another kind of recursive MDL-based algorithm for segmentation of words.

concise lexicon together with a concise representation of the corpus that yields the shortest total code length.

In an imagined scenario, we have a sending and receiving party. The sender is to transmit data (the corpus) to the receiver using the shortest possible code. Both sender and receiver are supposed to share some common knowledge, so that the receiver is able to decode the code used by the sender. Here, we assume that the sender first encodes the lexicon, which consists of the morphs spelled out. To spell out a morph there is a unique code for every character in the alphabet. The code length for each character α is derived from its probability $P(\alpha)$. We thus assume that there is a given probability distribution over the characters in the alphabet and a code for each character, which both parties have knowledge of. The code length of the entire lexicon is then

$$L(\text{lexicon}) = \sum_{j=1}^{M} \sum_{k=1}^{\text{length}(\mu_j)} -\log P(\alpha_{jk}), \tag{2}$$

where j runs over all morphs μ_j in the lexicon, which contains a total number of M morphs. The index k runs over the characters α_{jk} in each morph μ_j. The code length of an individual character is the negative logarithm of its probability. (All logarithms in this section have base 2, which means that code lengths are measured in number of *bits*.) To be able to distinguish where one morph ends and the next begins, every morph is assumed to end in a morph boundary character, which is part of the alphabet. Finally, the end of the lexicon is marked by appending an additional morph boundary character.

Next, the probability distribution of the morphs in the lexicon is transmitted. This probability distribution is used for creating codes for the morphs and is needed when the corpus is encoded (see Eq. (6)). The probabilities are estimated from the proposed segmentation of the corpus, such that the probability of each morph is its frequency (number of occurrences) in the corpus divided by the total number of morphs in the corpus. We denote the total number of morphs in the corpus by N, which is a *token count*, since the same morph may naturally occur many times in the segmented corpus. In contrast, the number of morphs in the lexicon, M, is a *type count*, since the lexicon contains no two identical morphs. In order to avoid sending floating-point numbers, which have to be truncated to some precision, the sender first encodes the total number of morphs, N, and then the frequencies of the M distinct morphs. This means that only positive integers need to be encoded. The receiver can compute the probability of a morph by dividing its frequency by the total number of morphs. The value of N can be encoded using the following number of bits (Rissanen, 1989, p. 34):

$$L(N) \approx \log c + \log N + \log\log N + \log\log\log N + \cdots, \tag{3}$$

where the sum includes all positive iterates, and c is a constant, about 2.865. The formula for $L(N)$ gives a code length for any positive integer and is related to the probability distribution $2^{-L(N+1)}$, which Rissanen calls a *universal prior* for non-negative integers.

The M individual morph frequencies could be encoded in the same way, but there exists a more compact code. As there are $\binom{N-1}{M-1}$ possibilities of choosing M positive integers (the frequencies) that sum up to N, approximately the following code length applies (Rissanen, 1989, pp. 35–37):

$$L(\text{morph frequencies}) = \log \binom{N-1}{M-1} = \log \frac{(N-1)!}{(M-1)!(N-M)!}. \quad (4)$$

Note that Stirling's approximation can be applied for large factorials: $n! \approx (n/e)^n \sqrt{2n\pi}$.

One way to derive Eq. (4) is to imagine that the N morph tokens are sorted into alphabetical order and each morph is represented by a binary digit. Since some morphs occur more than once, there will be sequences of several identical morphs in a row. Now, initialize all N bits to zero. Next, every location, where the morph *changes*, is switched to a one, whereas every location, where the morph is identical to the previous morph, is left untouched. There are $\binom{N}{M}$ possibilities of choosing M bits to switch in a string of N bits. However, as the value of the first bit is known to be one, it can be omitted, which leaves us with $\binom{N-1}{M-1}$ possible binary strings. These strings can be regarded as binary integers and ordered by magnitude. In the coding scheme, it is sufficient to tell *which* out of the $\binom{N-1}{M-1}$ strings is the actual one. Thus, the binary string itself is never transmitted.

At this stage the entire model has been encoded using the following code length:

$$L(\theta) = L(\text{lexicon}) + L(N) + L(\text{morph frequencies}). \quad (5)$$

The code length of the corpus (the data) is given by

$$L(x|\theta) = \sum_{i=1}^{N} -\log P(\mu_i|\theta), \quad (6)$$

where the corpus contains N morph tokens; μ_i is the ith token; and $P(\mu_i|\theta)$ is the probability of that morph, which can be calculated from the morph frequencies as described above. The length of each pointer is the negative logarithm of the probability of the morph it represents. Thus, frequently occurring morphs have shorter codes while rare morphs have longer codes. This implies that there is a tendency for frequent substrings of words to be selected as morphs, because they can be coded efficiently as a whole, whereas rare substrings are better coded in parts, as sequences of more common substrings.

3.1.2. Discussion of the formulation of the model

We have chosen to formulate the model in an MDL framework, because we find the interpretation offered by this framework instructive and rather intuitive. The task is to code, or compress, information into the smallest possible number of bits. This goal of minimizing the required storage capacity has practical implications, which are familiar to most people: Computer memory and disk space are limited and so is the capacity of the human memory. Saving storage capacity is thus valuable. In this respect, our model resembles text compression algorithms, which have also been proposed for word segmentation (e.g., Teahan et al., 2000).

However, the model could as well be expressed in a probabilistic, or Bayesian, framework. Instead of finding the parameter values that minimize the overall code length, the parameter values would be chosen that *maximize the overall probability* of the model and the data given the model. This is called MAP estimation. Conveniently, MDL and MAP are equivalent and produce the same result, as is demonstrated, e.g., in Chen (1996). A code length, $L(z)$, is transformed to the probability $P(z)$ through the simple formula: $P(z) = 2^{-L(z)}$. The sums of code lengths in Eqs. (1), (2), (5), and (6) can be rewritten as products of probabilities.

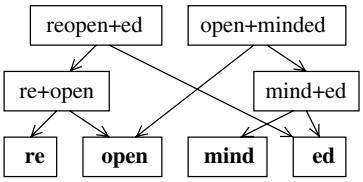

Fig. 1. Hypothetical splitting trees for two English words.

A model resembling the current one is expressed in a MAP framework in Creutz (2003). There, explicit prior distributions for morph length and morph frequency are utilized. As the beneficial effect of the priors diminishes when large training corpora are used (as is now the case), a simpler scheme was judged sufficient for the current work.

3.1.3. Search heuristics

The search algorithm utilizes a greedy search, where each word in the corpus is initially a morph of its own. Different morph segmentations are proposed and the segmentation yielding the shortest code length is selected. The procedure continues by modifying the segmentation, until no significant improvement is obtained.

The MDL framework is used as a theoretical basis for our model. Our intention is not to build real encoders and decoders. Therefore, in practice some minor simplifications to the overall code length as expressed in Section 3.1.1 are possible. As it is important to know the *difference* in code length between different segmentations, the absolute value is not crucial. The contribution of $L(N)$ in Eq. (3) is insignificant and is left out. We have also used a uniform distribution for the characters in the alphabet (see Eq. (2)), a solution chosen due to its simplicity.[7]

The search algorithm makes use of a data structure, where each distinct word form in the corpus has its own binary splitting tree. Fig. 1 shows the hypothetical splitting trees of the English words 'reopened' and 'openminded'. The leaf nodes of the structure are unsplit and they represent morphs that are present in the morph lexicon. The leaves are the only nodes that contribute to the overall code length of the model, whereas the higher-level nodes are used solely in the search. Each node is associated with an occurrence count indicating the number of times it occurs in the corpus. The occurrence count of a node always equals the sum of the counts of its parents. For instance, in Fig. 1 the count of the morph 'open' would equal the sum of the counts of 'reopen' and 'openminded'.

During the search process, modifications to the current morph segmentation are carried out through the operation `resplitnode` (see Algorithm 1). All distinct word forms in the corpus are sorted into a random order and each word in turn is fed to `resplitnode`, which produces a binary splitting tree for that word. First, the word as a whole is considered as a morph to be added to the lexicon. Then, every possible split of the word in two substrings is evaluated. The split (or no split) yielding the lowest code length is selected. In case of a split, splitting of the two parts continues recursively and stops when no more gains in overall code length can be

[7] Later experiments, where character probabilities were estimated from the corpus, have produced similar morph lexicons and segmentations, and the resulting n-gram models did not perform significantly better or worse when cross-entropies were calculated for different n-gram orders on two held-out test sets.

obtained by splitting a node into smaller parts. After all words have been processed once, they are again shuffled by random, and each word is reprocessed using `resplitnode`. This procedure is repeated until the overall code length of the model and corpus does not decrease significantly from one epoch to the next.

Every word is processed once in every epoch, but due to the random shuffling, the order in which the words are processed varies from one epoch to the next. It would be possible to utilize a deterministic approach, where all words would be processed in a predefined order, but the stochastic approach (random shuffling) was preferred, because deterministic approaches were suspected to cause unforeseen bias. If one were to employ a deterministic approach, it seems reasonable to sort the words in order of increasing or decreasing length, but even so, words of the same length ought to be ordered somehow, and for this purpose random shuffling seems much less prone to bias.

However, the stochastic nature of the algorithm means that the outcome depends on the series of random numbers produced by the random generator. The effect of this indeterminism was studied by running the morph segmentation algorithm with 11 different random seeds. For each outcome, n-gram language models were trained as described in Sections 3.1.4 and 3.5. The language models were tested as described in Section 3.6 on two different test sets. It was observed that for the n-gram orders 2, 3, and 4, the variation in cross-entropy due to different random seeds was always within the very small range of ± 0.01 bits. Therefore, we found it sufficient to use only the outcome of the first of the 11 runs in the speech recognition experiments.

Algorithm 1 resplitnote(*node*)

Require: *node* corresponds to an entire word or a substring of a word

//REMOVE THE CURRENT REPRESENTATION OF THE NODE//
if *node* is present in the data structure **then**
  for all nodes m in subtree rooted at *node* **do**
    decrease count(m) by count (*node*)
    if m is a leaf node, i.e., a morph **then**
      decrease $L(x|\theta)$ and L(morph frequencies) accordingly
  if count(m) $= 0$ **then**
    remove m from the data structure
    subtract contribution of m from L(lexicon) if m is a leaf node
//FIRST, TRY WITH THE NODE AS A MORPH OF ITS OWN//
restore *node* with count(*node*) into the data structure as a leaf node
increase $L(x|\theta)$ and L(morph frequencies) accordingly
add contribution of *node* to L(lexicon)
bestSolution $\leftarrow [L(x,\theta)$,*node*$]$

//THEN TRY EVERY SPLIT OF THE NODE INTO TWO SUBSTRINGS//
subtract contribution of *node* from $L(x,\theta)$,
but leave *node* in data structure
store current $L(x,\theta)$ and data structure
for all substrings *pre* and *suf* such that *pre* $\circ$ *suf* $=$ *node* **do**

```
  for subnode in [pre, suf] do
    if subnode is present in the data structure then
      for all nodes m in the subtree rooted at subnode do
        increase count(m) by count (node)
        increase L(x|θ) and L(morph frequencies) if m is a leaf node
    else
      add subnode with count (node) into the data structure
      increase L(x|θ) and L(morph frequencies) accordingly
      add contribution of subnode to L(lexicon)
  if L(x, θ) < code length stored in bestSolution then
    bestSolution ← [L(x, θ),pre,suf]
  restore stored data structure and L(x, θ)

//SELECT THE BEST SPLIT OR NO SPLIT//
select the split (or no split) yielding bestSolution
update the data structure and L(x, θ) accordingly
if a split was selected, such that pre ∘ suf = node then
  mark node as a parent node of pre and suf
  //PROCEED BY SPLITTING RECURSIVELY//
  resplitnode(pre)
  resplitnode(suf)
```

3.1.4. Language modeling with statistical morphs

The flow of operations for estimating a morph-based n-gram model is shown in Fig. 2. A corpus vocabulary is extracted from a text corpus, such that every distinct word form in the corpus occurs once in the vocabulary. This corpus vocabulary is used as input to the morph segmentation algorithm, which produces a morph lexicon, where every morph has a particular probability (see Eqs. (3)–(6)).

The morph segmentation algorithm also produces a segmentation of the words in the corpus vocabulary. However, this segmentation is not used as such, but the Viterbi algorithm is applied in order to produce the final morph segmentation of the words in the corpus. The Viterbi algorithm finds the most probable segmentation of a word given the morph lexicon and the morph

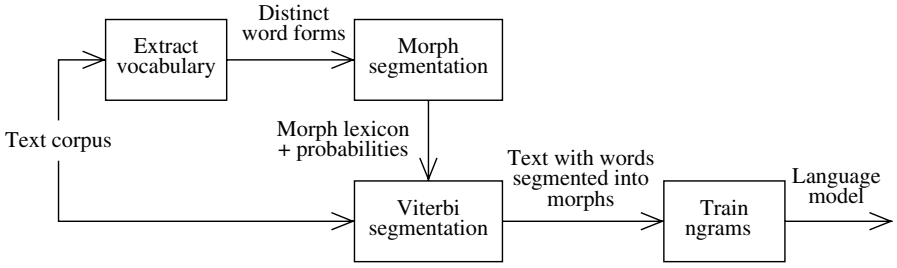

Fig. 2. The steps in the process of estimating a language model based on statistical morphs from a text corpus.

probabilities. The Viterbi search advances from left to right, or sequentially, in contrast to the recursive search heuristics described above. Therefore, the final segmentation differs, but only slightly, from the segmentation produced by recursive splitting. The rationale for utilizing one search algorithm when *estimating* the segmentation model and another one when *using* it is that the recursive search avoids local minima better than the Viterbi search, but once a model is estimated, the Viterbi algorithm can easily provide segmentations for new word forms that were not present in the training data. This means that it is possible to train morph segmentation models on only part of the words in the text corpus and yet obtain a morph segmentation for all words.[8]

In this work, two different sets of statistical morphs were trained. A morph lexicon containing 66,000 morphs was produced by extracting a corpus vocabulary containing all words in the corpus. Another smaller morph lexicon (26,000 morphs) was obtained by training the algorithm on a corpus vocabulary where word forms occurring less than three times in the corpus were filtered out. This approach is motivated by the fact that rare word forms might be noise (such as misspellings and foreign words) and their removal might increase the robustness of the algorithm.

Once the corpus has been segmented into morphs, *n*-gram language models are estimated over the morph sequences using Kneser-Ney smoothing. Word boundaries need to be modeled explicitly, as only part of the transitions between morphs occur at word boundaries. In our solution, word boundaries are realized as separate units, and occur in the *n*-grams as morphs among the others.

Note that the extraction of corpus vocabularies differs from the approach used in Creutz and Lagus (2002), where the whole corpus was used as training data for the algorithm. In the original approach, the amount of training data was much larger, because many word forms naturally occurred many times. Large training corpora lead to large morph lexicons, since the algorithm needs to find a balance between the two in its attempt to obtain the globally most concise model. By choosing only one occurrence of every word form as training data, the optimal balance occurs at a smaller morph lexicon, while still preserving the ability to recognize good morphs, which are common strings that occur in words in different combinations with other morphs.

It is possible to reduce the size of the morph lexicon also when training the model on a corpus instead of a corpus vocabulary. The rarest word forms can be filtered out from the corpus and the frequencies of the remaining morphs can be lowered, such that their relative weight in the corpus remains approximately the same as before. By filtering out words occurring less than 20 times in the corpus the resulting lexicon contained 35,000 morphs, which is comparable to the 26,000 morph lexicon obtained when a corpus vocabulary was used in the training. When *n*-gram language models were trained using both approaches, the two performed as well in terms of cross-entropy calculated on two separate test sets. Better results were not obtained for larger morph lexicons.

However, when comparing the obtained segmentations to a grammatical morph segmentation (see Section 3.2), the segments trained on the corpus vocabulary clearly matched the grammatical

[8] It is possible, although this occurs extremely rarely in our experiments, that there is no Viterbi parse in the model of a word form not observed in the training data. This might happen, e.g., if a word contains a new letter, which does not occur in any morph in the model. To make sure that every word obtains a segmentation, every individual letter, which does not already exists as a morph in the lexicon, can be suggested as a morph by the Viterbi search with a very low probability.

morphs better than the segments trained on the corpus itself. It sounds desirable to model language based on units with a close correspondence to actual morphemes, i.e., units associated with a meaning. Therefore, the *n*-gram language models used in our speech recognition experiments are based on morphs trained using a corpus vocabulary.

3.2. Grammatical morphs

To obtain a segmentation of words into grammatical morphs, each word form was run through a morphological analyzer[9] based on the two-level morphology of Koskenniemi (1983). The output of the analyzer consists of the base form of the word together with grammatical tags indicating, e.g., part-of-speech, number and case. Boundaries between the constituents of compound words are also marked. We have created a rule set that processes the output of the analyzer and produces a grammatical morph segmentation of the words in the corpus. The segmentation rules are derived from the morphological description for Finnish given by Hakulinen (1979).

Words not recognized by the morphological analyzer are treated as OOV words and split into individual phonemes, so that it is possible to construct any word form by a concatenation of phonemes. Such words make up 4.2% of all the words in the training corpus, and 0.3% and 3.8% of the words in the two test sets (BOOK and NEWS, respectively).

The *n*-gram probabilities are estimated over the segmented training corpus, and as in the case of statistical morphs, word boundaries are modeled explicitly as separate units.

A slightly newer version of the grammatical morph segmentation is called *Hutmegs* (Helsinki University of Technology Morphological Evaluation Gold Standard). Hutmegs is publicly available[10] for research purposes (Creutz and Lindén, 2004). For full functionality, an inexpensive license must additionally be purchased from Lingsoft, Inc.

3.3. Words

Vocabularies containing entire word forms have also been tested. The number of possible Finnish word forms is very high. Since no vocabulary can hold an unlimited number of words, we have selected the most common words in the training corpus to make up the vocabulary. Instead of discarding the remaining OOV words, they are split into phonemes. The *n*-gram probabilities are estimated as usual over the training corpus, where the rare word forms have been split into phonemes.

Word breaks are modeled so that we have two variants of each phoneme, one for occurrences in the beginning or middle of a word and one for occurrences at the end of a word. Each unsplit word is assumed implicitly to end in a word break.

Even if we choose a huge vocabulary of the 410,000 most common words in the training corpus, 5.0% of all words are OOV and need to be split. The OOV rate of the test sets BOOK and NEWS are 7.3% and 5.0%, respectively. We have also chosen to experiment with a smaller vocabulary containing 69,000 words in order to have a vocabulary of approximately the same size for statistical

[9] Licensed from Lingsoft, Inc.: http://www.lingsoft.fi/.
[10] URL: http://www.cis.hut.fi/projects/morpho/.

morphs (66,000), grammatical morphs (79,000), and words. For the smaller word vocabulary, the OOV rates are as follows: 15.1% (training), 19.9% (test BOOK), 13.1% (test NEWS).

The splitting of OOV words into phonemes allows for a direct comparison with other language models that have an OOV-rate of 0%. It is thus convenient in two ways: Firstly, we achieve a theoretical 100% coverage of any possible word (and non-word) of the language. Secondly, we can make fair comparisons with other language models that work in the same way, without having to weight two measures against each other in the comparison (i.e., cross-entropy or perplexity against the OOV-rate). This, of course, presupposes that the method for achieving a 0% OOV is not *worse* than the standard method involving OOV words. Therefore, we compared the effect of splitting OOV words into phonemes with the effect of leaving them out (or actually, replacing them with a special OOV symbol). This comparison of speech recognition accuracy was performed only on the 410,000 word model, and both approaches performed on roughly equal level (see Section 5.3).

3.4. Example

Table 1 shows the splittings of the same Finnish example sentence (*''Tuore-mehuasema aloitti maanantaina omenamehun puristamisen Pyynikillä.''*) using the six different lexicon configurations. The statistical morph segmentations differ from each other in the larger (66k) and the smaller (26k) lexicon. In the larger lexicon the two morphemes 'tuore' (fresh) and 'mehu' (juice) occur together as 'tuoremehu'. The place name ''Pyynikki'' is segmented as 'pyynik' (in front of the ending '-illä') in the large lexicon, whereas it is split into the nonsense 'pyy+nik' by the smaller lexicon. In the grammatical segmentation ''Pyynikki'' is unknown to the morphological analyzer and has been split into phonemes. The word for ''juice factory'' ('tuoremehuasema') is rare and therefore

Table 1
A sentence of the training corpus: *''Tuoremehuasema aloitti maanantaina omenamehun puristamisen Pyynikillä.''* segmented using different lexicons

Model	Segmentation
Statist. morphs (26k)	tuore mehu asema # al oitti # maanantai na # omena mehu n # purista misen # pyy nik illä #
Statist. morphs (66k)	tuoremehu asema # aloitti # maanantai na # omena mehu n # purista misen # pyynik illä #
Gramm. morphs (79k)	tuore mehu asema # aloitt i # maanantai na # omena mehu n # purista mise n # p yy n i k i ll ä #
Words (69k)	t u o r e m e h u a s e m a# aloitti# maanantaina# o m e n a m e h u n# p u r i s t a m i s e n# pyynikillä#
Words (410k)	t u o r e m e h u a s e m a# aloitti# maanantaina# omenamehun# puristamisen# pyynikillä#
Words-OOV (410k)	OOV# aloitti# maanantaina# omenamehun# puristamisen# pyynikillä#
Literal translation	fresh juice station # start -ed # Monday on # apple juice of # press -ing # Pyynikki in #

(An English translation reads: *''On Monday a juice factory started to press apple juice in Pyynikki.''*) The word fragments are separated by space. Word breaks are indicated by a number sign (#). In case of the word models, the word breaks are part of other fragments, otherwise they are units of their own.

it is an OOV in all word models. In the word models 69k and 410k, it has been split into phonemes. In the words-OOV (410k) model it has been replaced by an OOV symbol.

3.5. Kneser-Ney smoothing

The n-gram probabilities were estimated from the corpus for each of the segmentation approaches. Modified Kneser-Ney smoothing (Chen and Goodman, 1999) was utilized due to its favorable behavior, not only on low-order, but also on high-order n-grams. The estimation of the n-gram probabilities was carried out using the SRI Language Modeling Toolkit (Stolcke, 2002).

3.6. Cross-entropy comparisons

The performance evaluation of a language model is usually based on the probability the language model assigns for an independent test text. Because the probability as such depends strongly on the length of the text, derivative measures normalized over words are often used, the most common being cross-entropy and perplexity.

Given the text data T consisting of W_{T} words and a language model M, the *cross-entropy* $H_{\mathrm{M}}(T)$ of the model on the data is given by

$$H_{\mathrm{M}}(T) = -\frac{1}{W_T}\log_2 P(T|M), \tag{7}$$

which can be interpreted as the number of bits needed to encode each word on average (Chen and Goodman, 1999). The data probability $P(T|M)$ is often decomposed into probabilities of words: $P(T|M) = \prod_{i=1}^{W_T} P(w_i|w_{i-1},\ldots,w_1;M)$, and in n-gram models only a few preceding words are considered instead of the whole history $(w_{i-1},\ldots,w_1)$. Note, however, that Eq. (7) does not assume that the underlying model defines probabilities on *words*, as long as the model gives the probability for the whole test text. Thus, even if we compare cross-entropies of models that have different word fragment sets, the cross-entropy comparisons are fair.

Perplexity is very closely related to cross-entropy. It is defined as

$$\mathrm{Perp}_{\mathrm{M}}(T) = \left(\prod_{i=1}^{W_T} P(w_i|w_{i-1},\ldots,w_1;M)\right)^{-\frac{1}{W_T}} \tag{8}$$

and it is easy to see that the relation to cross-entropy is given by

$$\mathrm{Perp}_{\mathrm{M}}(T) = 2^{H_{\mathrm{M}}(T)}. \tag{9}$$

It has been suggested that cross-entropy predicts the word error rates better than perplexity (Goodman, 2001). Thus, we have decided to report cross-entropies instead of perplexities in the experiments.

We tested n-gram models of order 2–7 and the results are shown in Fig. 3. The reported model size refers to the amount of memory the language model occupies in the memory of the decoder of our speech recognizer. The models are stored in a tree structure similar to the structure presented by Whittaker and Raj (2001), but without quantization. The word model without phonemes is

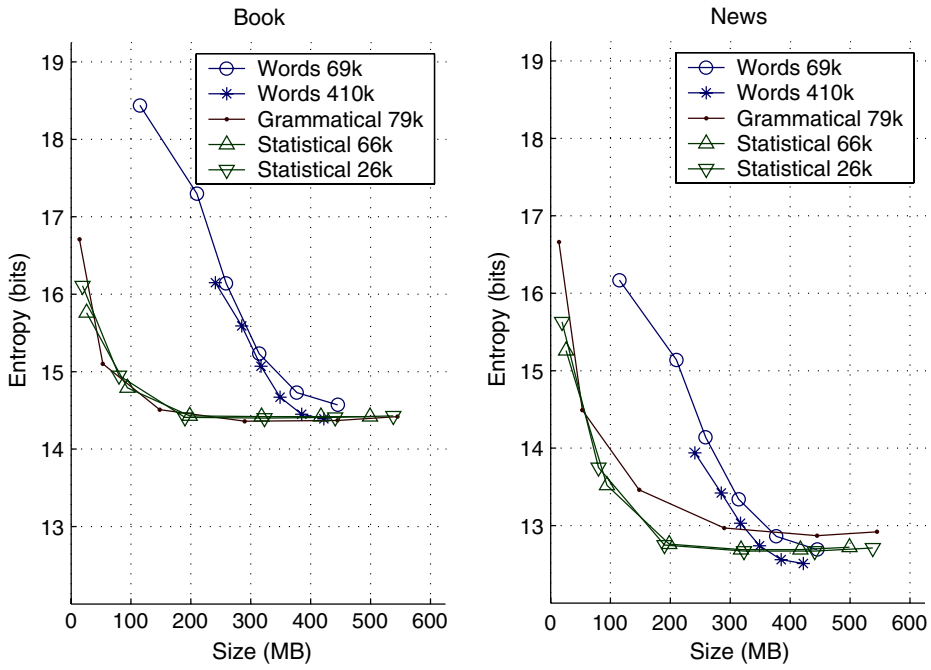

Fig. 3. The cross-entropy of the different models on the test data versus the corresponding model sizes. The six points along each curve are the orders 2–7 of the *n*-gram models.

omitted from the entropy comparison, because the OOV words would make the comparison meaningless.

The main notion from the results in Fig. 3 is that the statistical and grammatical morph models seem to be more effective with respect to model sizes when the order of the *n*-gram model is small. On higher-order models, the differences even out. We discuss the entropy results more in Section 5.3 together with the speech recognition results.

4. Speech recognition system

4.1. Acoustic modeling

In this section, we describe the acoustic models used in the experiments. Since our emphasis is on the language modeling, the acoustic part is not discussed in great detail. The main difference to the corresponding English LVCSR systems, in addition to the Finnish phonemes, is the utilization of monophone models embedded with explicit models for the phone duration.

4.1.1. Features and phoneme models

The acoustic features in our speech recognition system are based on 12 mel-frequency cepstral coefficients and power. In addition, first-order derivatives are computed for each feature yielding a 26-dimensional feature vector. The acoustic phoneme models are traditional hidden

Markov models (HMMs) with three states in a left-to-right topology. The emission probability densities are modeled by Gaussian mixtures (10 kernels per mixture) with diagonal covariance matrices.

The feature space is transformed linearly with a global matrix. The transformation is optimized in a maximum likelihood manner (Gales, 1999) to minimize the correlation between the feature components in each phoneme class. The acoustic models are initialized with Self-Organizing maps and then refined with Viterbi training (Kurimo, 1997).

Because our decoder does not handle acoustic phoneme contexts across units in the lexicon, and we study recognition based on different word fragments, we have used acoustic monophone models instead of triphone models to make the comparisons fair. In total, 44 phoneme models were used. Most of the Finnish phonemes have a short and a long variant, and we used separate models for them.

4.1.2. Phone duration models

Phone durations play a significant part in the comprehension of Finnish speech. There exist many word pairs which are distinguishable mainly by the duration of their phones. It is therefore necessary to use duration information to improve the speech recognition. Unfortunately, the traditional HMM-based acoustic models are rather poor at modeling phone durations, because of the intrinsic geometric state duration distributions (e.g., Pylkkönen and Kurimo, 2004).

Gamma distributions have been suggested as good models for state durations but this breaks the Markov property, which is heavily utilized in HMM algorithms, such as Baum-Welch and Viterbi searches. Modifying the algorithms to take these explicit duration distributions into account seriously degrades their efficiency. Several approaches have been suggested to overcome these problems (e.g., Russell and Cook, 1987; Bonafonte et al., 1993). In our preliminary testings, we have concluded that a simple post-processor approach proposed by Juang et al. (1985) provides a good trade-off between accuracy and efficiency for our purposes.

The post-processor duration model uses the output of the Viterbi alignment and adds a penalty term depending on the state occupancy durations in the alignment. This way the different alignments can be ranked using better state duration models than the standard HMMs offer. The augmentation of the likelihood obtained from the Viterbi alignment can be formulated as

$$\log \hat{f} = \log f + \alpha \sum_{j=1}^{N} \log d_j(\tau_j), \tag{10}$$

where f denotes the original likelihood score. In the augmentation term, α is an empirical scaling factor, N is the number of distinct HMM states through which the best path traversed, d_j are the duration probability distribution functions of those states, and τ_j are the time spent in each state.

Although there is a mismatch between the normal Viterbi alignment and the true optimal path in the explicit state duration models, the post-processor performs very well in practice. Due to the operation of the decoder, a large number of competing paths are ranked using the better state duration models, guiding the decoder to better results. The augmentation has a negligible impact on the recognition efficiency, making the post-processor model an interesting choice for integrating explicit state duration distributions into the HMMs.

4.2. The decoder

In speech recognition systems based on the HMM framework, the final recognition result is decided by the *decoder*. As the acoustic and language models give probabilities for HMM state sequences, decoding is theoretically very simple: one just computes the probability of every possible HMM state sequence given the observed speech, and selects the most probable one. However, because the number of possible state sequences is astronomical even for short speech signals, only a fraction of the sequences can be considered, and several different decoding approaches have been presented in the literature. A good overview of different techniques has been presented by Aubert (2002).

4.2.1. Overview of the approach

The one-pass decoder used in the experiments was originally developed along the principles of Ducoder (Willett et al., 2000). The main idea is that the language model is not built in the recognition network. Instead, the search process consists of two parts. The first part (*local acoustic search*) computes the acoustically best words (or word fragments in our case) starting from a given frame. The second part (*hypothesis search*) controls the local search and selects the acoustically best word fragments for certain frames. It combines the acoustically best fragments to longer word sequences (referred to as *hypotheses*), adds language model probabilities, and stores the most probable hypotheses in frame-wise stacks. Various pruning methods can be used to discard improbable hypotheses. The main advantage of separating the acoustic search and language modeling is that complex language models can be used easily without affecting the acoustic search.

While many decoders use a lexicon containing a collection of the most frequent words, our lexicon mostly contains word fragments. Thus, a word break does not follow every fragment implicitly, and it has to be modeled in the lexicon and language model.

4.2.2. Local acoustic search

The task of the local acoustic search is to compute the acoustic probabilities for the most probable word fragments starting from the given frame. In the HMM framework, a common practice is to make the *Viterbi approximation*, and find only the most probable state sequence for each word fragment (Rabiner, 1989). This can be done efficiently by combining all fragments into a single big tree-shaped HMM, so that common prefixes of the state sequences are merged. Then the *Viterbi search* algorithm can be used to compute the cumulative probability of the best path to each state for each frame. After completing the search, every state corresponding to a last state of a word fragment, gives the best alignment and best probability of the fragment ending at each frame.

After the Viterbi search is completed, the results can be utilized to choose the best ending time for each word fragment. Currently, the frame which has the best average log-probability per frame is chosen. In addition, a few neighboring frames are used in the next stage to search for the best location for fragment boundaries.

4.2.3. Stacks and hypotheses

The local search described above can be used to find the acoustically best word fragments starting from a given frame, but since it considers only single fragments, several fragments must be

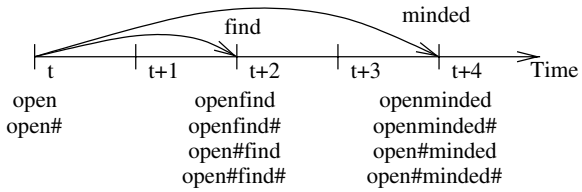

Fig. 4. Local acoustic search, stacks, and hypotheses. The two acoustically most probable fragments *find* and *minded* are used to expand the two hypotheses at frame t. The resulting hypotheses are placed in stacks according to the best ending times. The hypotheses comprise all combinations including and excluding word breaks (#).

combined to form longer hypotheses. The hypotheses are stored in frame-wise stacks, so that all the hypotheses ending at a certain frame are stored in the corresponding stack. Initially, only the first stack contains an empty hypothesis. Then, the decoder performs the local search to find the acoustically best word fragments starting from the first frame. New hypotheses are generated by copying the empty hypothesis and appending each fragment in turn to the hypothesis. The new hypotheses are stored according to the ending times of the fragment. At this point, the language model probabilities can also be taken into account, because when a fragment is appended to a hypothesis, the language model history is fixed, and the probability of the hypothesis can be updated.

Fig. 4 presents a simplified English example. The local acoustic search starts on frame t and the two acoustically most promising fragments *find* and *minded* are found. The two hypotheses ending at frame t are then expanded using the two fragments. This results in eight new hypotheses, as versions with and without word breaks are considered for each hypothesis.

4.2.4. Pruning

The procedure described above generates an exponentially growing number of hypotheses unless the decoder discards all but the most probable hypotheses. Pruning is essential to make the decoding feasible, but it is also a potential source for recognition errors. It is noteworthy that the effect of the pruning on the recognition result is not always very intuitive. When recognition experiments are made in order to improve some part of the recognition system, for example language modeling, one has to consider the impact of the pruning techniques on the different methods tested. One method may benefit from looser pruning, while another one may be effective also when tighter pruning is applied.

In our decoder, the search space of the local acoustic search is pruned using two standard methods. First, all paths falling outside a log-probability beam, B, are discarded, and the number of active paths, S, is limited during the Viterbi search. Additionally, after the best ending times and the corresponding log-probabilities have been computed for each word fragments, only the F best fragments are utilized.

Pruning is also needed in the hypothesis search. Willett et al. (2000) present various pruning methods, but our decoder prunes hypotheses only within one stack, as it is not straightforward to compare hypotheses ending at different frames. In our decoder, the size of each stack is limited to H hypotheses. In addition, hypotheses which have the L last words in common are pruned so that only the best is retained. Theoretically, the value of L should depend on the language model, because this pruning of the hypotheses should be made only when it is sure that the language

model probabilities cannot change the order of the hypotheses in future. However, all pruning parameters affect each other and the value $L = 2$ has turned out to function well in combination with the other pruning parameters $H = 10$, $F = 100$ and $S = 3000$. The beam-width B has varied between 40 and 85.

5. Recognition experiments

5.1. Setup

The recognition system was evaluated in both the BOOK and the NEWS tasks (see Section 2). For each of the lexicon types (statistical morphs, grammatical morphs, words, and words-OOV), and for each n-gram model (orders 3–5), development data were used to tune the weight between acoustic and language models. The weight was tuned to optimize the phoneme error rate in the development data. We did not study n-gram models of order 6 or 7 in the recognition, because of the high memory requirements.

When speech recognition experiments are done in order to evaluate and compare different ways to construct the lexicon, it is important to experiment with different decoder pruning settings to get a clear picture of how the methods behave on different recognition speeds. The pruning settings of the decoder may affect the models using a different type of lexicon differently and the optimal setting may vary. For the recognition of the evaluation data, we used four different beam pruning settings in the local acoustic search to study the performance on varying decoding speeds.

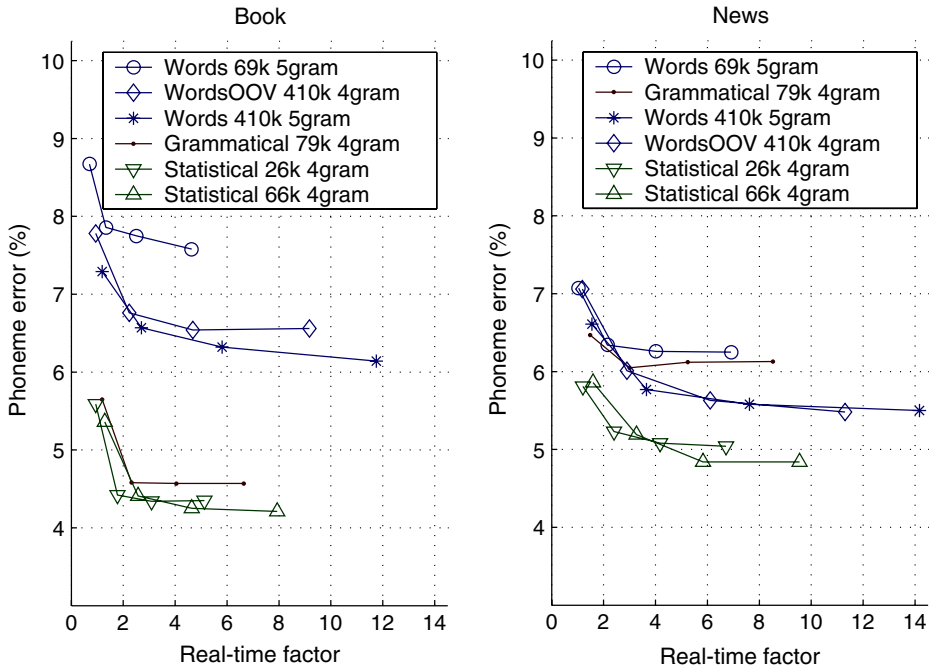

Fig. 5. Phoneme error rates of the speech recognition experiments. The four points along each curve are the four different beam pruning settings.

Table 2
Best phoneme error rates for each model (PHER), the percentage of word fragments consisting of a single phoneme in the test data (Singles), and the average span in phonemes of the *n*-grams used (AvgSpan)

Model	BOOK			NEWS		
	PHER (%)	Singles (%)	AvgSpan (phon.)	PHER (%)	Singles (%)	AvgSpan (phon.)
Statist. 66k, 4-gr.	4.21	1.2	12.8	4.84	1.8	13.5
Statist. 26k, 4-gr.	4.35	2.1	11.8	5.04	2.7	12.5
Gramm. 79k, 4-gr.	4.57	6.6	10.5	6.05	9.3	10.3
Words 410k, 5-gr.	6.14	12.0	22.9	5.50	9.6	25.7
Words 69k, 5-gr.	7.58	30.3	13.8	6.25	21.0	17.5

5.2. Results and significance tests

Fig. 5 shows the best phoneme error versus real-time factor curves for different lexicons. Only the curve of the best *n*-gram model (see Table 2) is plotted for each lexicon, and the four points along the curve are the four pruning settings of the decoder. In Fig. 6, the corresponding word error rates are presented. In Fig. 7, the phoneme error rates are shown again, but here the points on the curve are the order of the *n*-gram model, and the horizontal axis shows the size of the language models as described in Section 3.6. The least amount of pruning was used in the decoding.

Both tasks were divided into several independent segments before recognition in order to measure the statistical significance of the results. The BOOK task had 20 segments and the NEWS task 40 segments. The significance was evaluated with the Wilcoxon signed-rank test (Milton and Arnold, 1995) for paired data (level 0.05).

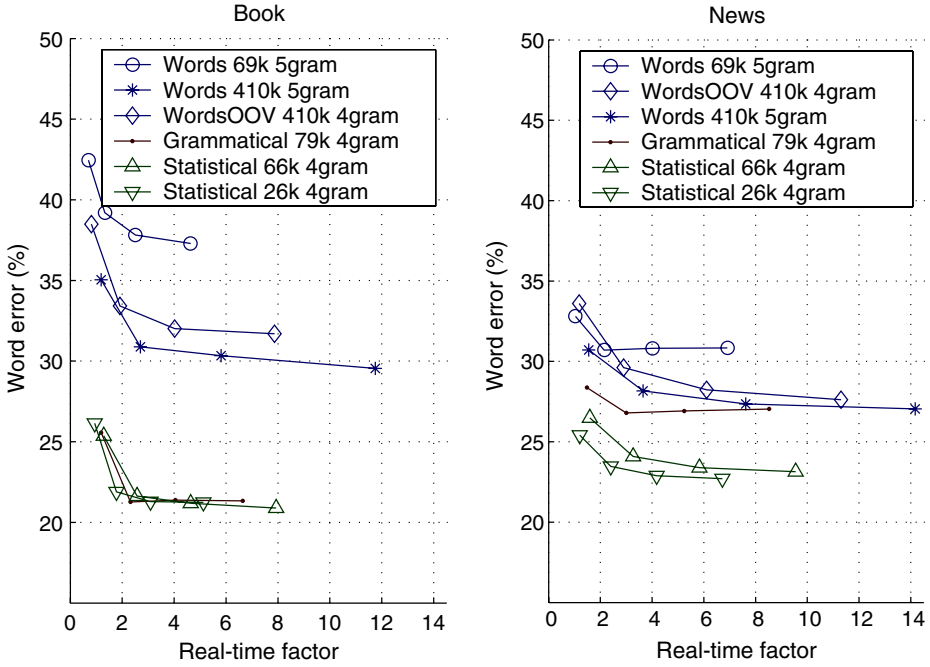

Fig. 6. Word error rates of the speech recognition experiments. The four points along each curve are the four different beam pruning settings.

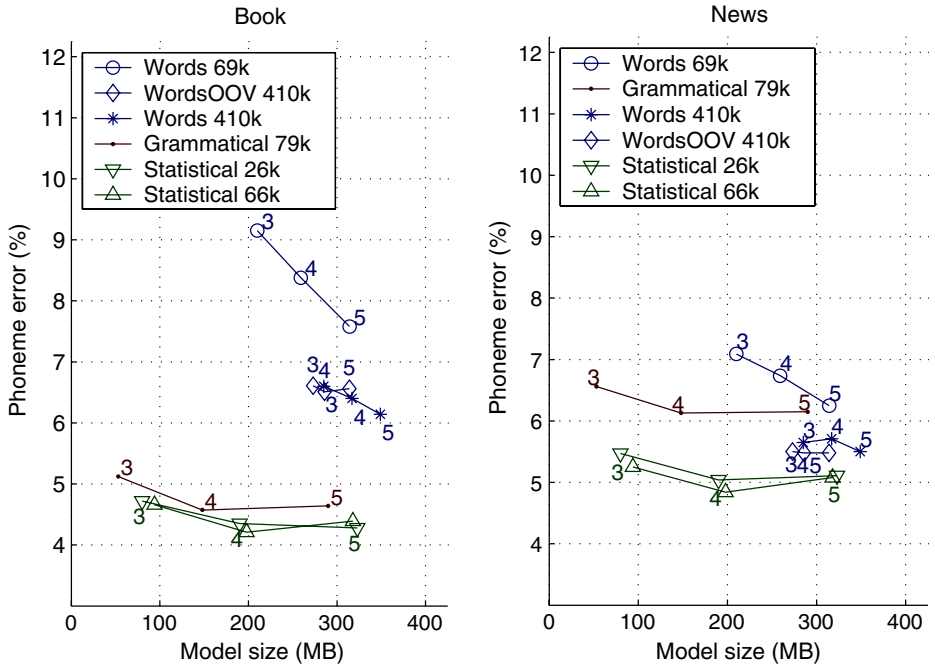

Fig. 7. Phoneme error rates and model sizes for different *n*-gram orders on the decoder settings with the least pruning.

The most relevant significance results are the following: On the left graph in Fig. 5, the difference between the best word model and any of the morph models is significant, while the differences between the three morph models are not significant. In the NEWS task (right side of Fig. 5), the best results of the both statistical morph models are significantly better than the other models. The difference between the two statistical morph models is not significant.

5.3. Discussion of the results

The most interesting result is that the statistical morphs seem to provide a good basis for modeling and recognition of Finnish. The entropy tests in Fig. 3 in Section 3.6 already suggested that morph models are more efficient than word models with respect to memory consumption. Furthermore, the same efficiency can also be seen in Fig. 7. Even the 3-gram statistical morph models seem to outperform the higher-order word models in both tasks. It is also interesting to see that the statistical morph model is not very sensitive to the number of morphs in the lexicon. While the number of words considerably affects the performance of the word model, the two statistical morph models of different sizes are consistently equal.

Results in Figs. 5 and 6 also provide a verification that modeling OOV words as sequences of phonemes (Words 410k) does not degrade recognition performance in comparison to leaving them out altogether (Words-OOV 410k). Both models achieve roughly the same error rates.

5.3.1. Why are the statistical morphs good?

But what is the reason for the good recognition performance of the statistical morphs? Fig. 3 showed that the 3-gram statistical morph model gives higher entropies than the 5-gram word

model, but still it gives lower error rates. To rule out the possibility that the pruning methods used by the decoder would cause the difference, we ran a few additional recognition tests in the NEWS task, loosening various pruning settings further. Even so, the difference between the best statistical morph model and word model remained. Also, as the acoustic phoneme models have been monophones in our tests, the acoustic probabilities of the hypotheses are the same for morphs and words.

The exact reason for the difference is not clear, but one possibility is that statistical morph models offer a good balance between restricting the search space and modeling the rare words. In the word models, very common words might be modeled better, but the rare long words are more problematic. Modeling them phoneme by phoneme offers a huge number of acoustically similar hypotheses, and the weak phoneme language model is poor at guessing the correct recognition result. This hypothesis is partly supported by the number of single phonemes the splitting methods need to use for the test data. There seems to be some correlation with the actual error rates regardless of the applied splitting method, as can be seen in Table 2.

Table 2 further shows that the error rate does not correlate with the average span of the *n*-gram language models used. The span is calculated as the number of phonemes that are on average covered by an *n*-gram window in the test data, including word breaks as ''phonemes''. The best result for the morph models were obtained using 4-grams, which span about 10–13 phonemes. The best results for the word models were obtained using 5-grams spanning 14–26 phonemes. However, there is no correlation between the span, i.e., the size of the prediction window, and error rate. A correlation can only be traced within groups, when comparing morph models with each other, or word models with each other. This observation is not explained by the different orders of *n*-grams used for morphs and words. Thus, the selection of a good fragment inventory seems important and the effects of a bad fragment inventory cannot be fully compensated by longer-range *n*-grams despite the utilization of an efficient smoothing technique (Kneser-Ney).

5.3.2. The effect of the task

While the grammatical morphs seem to give good results in the BOOK task, their performance is much worse in the NEWS task. This is most likely due to the many foreign names in the latter task. The rule-based grammatical morphs cannot split these names properly, and the language model has to resort to the single phonemes to build the names correctly.

The nature of the task also affects the word models and statistical morphs. When moving from the BOOK to the NEWS task, the word models seem to improve, while the morph models seem to degrade, even if the entropy results in Section 3 suggested that both word and morph models would perform better in the NEWS task. This is partly explained by Table 2. The number of single phonemes used by the statistical morph models increases in the NEWS task, while the corresponding number for the word models decreases.

6. Conclusions

In this article, we have described and evaluated language models based on the segmentation of text corpora into suitable word fragments by an unsupervised machine learning algorithm. We have shown how such an automatically derived lexicon can be effectively used in language

modeling and speech recognition for Finnish, which is a good example of agglutinative, highly inflecting and compounding languages. Due to the huge amount of distinct word forms, the traditional methods based on full words are not very effective and it is not straightforward to train efficient language models with good coverage of the language.

The entropy measurements and recognition experiments on two Finnish tasks gave good results. To evaluate the modeling performance of these so-called statistical morphs, we additionally trained *n*-gram models for full words and rule-based grammatical morphs. By using statistical morph models instead of word models, significant relative error rate reductions of 31% and 12% were obtained in two tasks. Also the grammatical morphs performed well in the BOOK task, but much worse in the NEWS task, probably due to the larger number of foreign words not handled by the morphological analyzer. In addition, the statistical morph models seemed to be robust with respect to the number of morphs in the lexicon, and give good results already with lower-order *n*-gram models. This is useful especially if the memory resources of the speech recognizers are limited.

As a final conclusion, it seems important to find a good balance between modeling the frequent and rare words. Statistical morphs seem to provide a good basis for the recognition of Finnish speech with *n*-gram models. How well it can be utilized in other language models remains to be seen in future research.

Acknowledgments

This work was supported by the Academy of Finland in the projects *New information processing principles* and *New adaptive and learning methods in speech recognition*. Funding was also provided by the Finnish National Technology Agency (TEKES) and the Graduate School of Language Technology in Finland. We thank the Finnish Federation of the Visually Impaired, Departments of Speech Science and General Linguistics of the University of Helsinki, and Inger Ekman from the Department of Information Studies of the University of Tampere, for providing the speech data. We are also grateful to the Finnish news agency (STT) and the Finnish IT center for science (CSC) for the text data. This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views. We acknowledge that access rights to data and other materials are restricted due to other commitments. The stimulating discussions with Dr. Krista Lagus were essential for developing the morph segmentation algorithm. The comments by the two anonymous reviewers and Mr. Krister Lindén were very valuable for improving the quality of the manuscript.

References

Argamon, S., Akiva, N., Amir, A., Kapah., O., 2004. Efficient unsupervised recursive word segmentation using minimum description length. In: Proceedings of The 20th International Conference on Computational Linguistics (COLING).

Aubert, X.L., 2002. An overview of decoding techniques for large vocabulary continuous speech recognition. Computer Speech and Language 16 (1), 89–114.

Bilmes, J.A., Kirchhoff, K., 2003. Factored language models and generalized parallel backoff. In: Proceedings of the Human Language Technology Conference (HLT/NAACL), pp. 4–6.

Bonafonte, A., Ros, X., Mariño, J.B., 1993. An efficient algorithm to find the best state sequence in HSMM. In: Proceedings of the 3rd European Conference on Speech Communication and Technology (Eurospeech), pp. 1547–1550.

Brent, M.R., 1999. An efficient, probabilistically sound algorithm for segmentation and word discovery. Machine Learning 34, 71–105.

Byrne, W., Hajič, J., Ircing, P., Jelinek, F., Khudanpur, S., Krbec, P., Psutka, J., 2001. On large vocabulary continuous speech recognition of highly inflectional language – Czech. In: Proceedings of the 7th European Conference on Speech Communication and Technology (Eurospeech), pp. 487–489.

Byrne, W.J., Hajič, J., Krbec, P., Ircing, P., Psutka, J., 2000. Morpheme based language models for speech recognition of Czech. In: Proceedings of the Third, International Workshop on Text, Speech, Dialogue (TSD), pp. 211–216.

Chen, S.F., 1996. Building probabilistic models for natural language. Ph.D. Thesis, Harvard University.

Chen, S.F., Goodman, J., 1999. An empirical study of smoothing techniques for language modeling. Computer Speech and Language 13 (4), 359–393.

Creutz, M., 2003. Unsupervised segmentation of words using prior distributions of morph length and frequency. In: Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL), pp. 280–287.

Creutz, M., Lagus, K., 2002. Unsupervised discovery of morphemes. In: Proceedings of the Workshop on Morphological and Phonological Learning of ACL-02, pp. 21–30.

Creutz, M., Lagus, K., 2005. Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0. Tech. Rep. A81, Publications in Computer and Information Science, Helsinki University of Technology. Available from: <http://www.cis.hut.fi/projects/morpho/>.

Creutz, M., Lindén, K., 2004. Morpheme segmentation gold standards for Finnish and English. Tech. Rep. A77, Publications in Computer and Information Science, Helsinki University of Technology. Available from: <http://www.cis.hut.fi/projects/morpho/>.

Deligne, S., Bimbot, F., 1997. Inference of variable-length linguistic and acoustic units by multigrams. Speech Communication 23 (3), 223–241.

Gales, M.J.F., 1999. Semi-tied covariance matrices for hidden Markov models. IEEE Transactions on Speech and Audio Processing 7 (3), 272–281.

Geutner, P., Finke, M., Scheytt, P., 1998. Adaptive vocabularies for transcribing multilingual broadcast news. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). vol. 2, pp. 925–928.

Goldsmith, J., 2001. Unsupervised learning of the morphology of a natural language. Computational Linguistics 27 (2), 153–198.

Goodman, J.T., 2001. A bit of progress in language modeling, extended version. Tech. Rep. MSR-TR-2001-72, Microsoft Research, Extended version of a paper with the same title published in Computer Speech and Language 15, 403–434.

Hacioglu, K., Pellom, B., Ciloglu, T., Ozturk, O., Kurimo, M., Creutz, M., 2003. On lexicon creation for Turkish LVCSR. In: Proceedings of the 8th European Conference on Speech Communication and Technology (Eurospeech), pp. 1165–1168.

Hakulinen, L., 1979. Suomen kielen rakenne ja kehitys (The structure and development of the Finnish language), 4th Ed., Kustannus-Oy Otava.

Johnson, S., Jourlin, P., Moore, G., Jones, K.S., Woodland, P., 1999. The Cambridge University spoken document retrieval system. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 49–52.

Juang, B.H., Rabiner, L.R., Levinson, S.E., Sondhi, M.M., 1985. Recent developments in the application of hidden Markov models to speaker-independent isolated word recognition. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 9–12.

Kirchhoff, K., Bilmes, J., Das, S., Duta, N., Egan, M., Ji, G., He, F., Henderson, J., Liu, D., Noamany, M., Schone, P., Schwartz, R., Vergyri, D., 2003. Novel approaches to Arabic speech recognition: Report from the 2002 Johns-

Hopkins workshop. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), vol. 1, pp. 344–347.
Kneissler, J., Klakow, D., 2001. Speech recognition for huge vocabularies by using optimized sub-word units. In: Proceedings of the 7th European Conference on Speech Communication and Technology (Eurospeech), pp. 69–72.
Koskenniemi, K., 1983. Two-level morphology: a general computational model for word-form recognition and production. Ph.D. Thesis, University of Helsinki.
Kurimo, M., 1997. Using self-organizing maps and learning vector quantization for mixture density hidden Markov models. Ph.D. Thesis, Helsinki University of Technology.
Kwon, O.-W., Park, J., 2003. Korean large vocabulary continuous speech recognition with morpheme-based recognition units. Speech Communication 39 (3–4), 287–300.
McTait, K., Adda-Decker, M., 2003. The 300k LIMSI German broadcast news transcription system. In: Proceedings of the 8th European Conference on Speech Communication and Technology (Eurospeech), pp. 213–216.
Milton, J.S., Arnold, J.C., 1995. Introduction to Probability and Statistics, third ed. McGraw-Hill, New York (Chapter 10).
Ordelman, R., van Hessen, A., de Jong, F., 2003. Compound decomposition in Dutch large vocabulary speech recognition. In: Proceedings of the 8th European Conference on Speech Communication and Technology (Eurospeech), pp. 225–228.
Pylkkönen, J., Kurimo, M., 2004. Using phone durations in Finnish large vocabulary continuous speech recognition. In: Proceedings of the 6th Nordic Signal Processing Symposium (Norsig), pp. 324–326.
Rabiner, L.R., 1989. A tutorial on hidden Markov models and selected applications in speech recognition. Proceedings of the IEEE 77 (2), 257–286.
Rissanen, J., 1989. Stochastic complexity in statistical inquiry. World Scientific Series in Computer Science 15, 79–93.
Russell, M.J., Cook, A.E., 1987. Experimental evaluation of duration modelling techniques for automatic speech recognition. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), vol. 4, pp. 2376–2379.
Siivola, V., Hirsimäki, T., Creutz, M., Kurimo, M., 2003. Unlimited vocabulary speech recognition based on morphs discovered in an unsupervised manner. In: Proceedings of the 8th European Conference on Speech Communication and Technology (Eurospeech), pp. 2293–2296.
Stolcke, A., 2002. SRILM–an extensible language modeling toolkit. In: Proceedings of the 7th International Conference on Spoken Language Processing (ICSLP), pp. 901-904. Available from: <http://www.speech.sri.com/projects/srilm/>.
Szarvas, M., Furui, S., 2003. Evaluation of the stochastic morphosyntactic language model on a one million word Hungarian task. In: Proceedings of the 8th European Conference on Speech Communication and Technology (Eurospeech), pp. 2297–2300.
Teahan, W.J., Wen, Y., McNab, R., Witten, I.H., 2000. A compression based algorithm for Chinese word segmentation. Computational Linguistics 26 (3), 375–393.
Venkataraman, A., 2001. A statistical model for word discovery in transcribed speech. Computational Linguistics 27 (3), 352–372.
Vergyri, D., Kirchhoff, K., Duh, K., Stolcke, A., 2004. Morphology-based language modeling for Arabic speech recognition. In: Proceedings of the 8th International Conference on Spoken Language Processing (ICSLP), vol. 3, pp. 2245–2248.
Whittaker, E., Raj, B., 2001. Quantization-based language model compression. In: Proceedings of the 7th European Conference on Speech Communication and Technology (Eurospeech), vol. 1, pp. 33–36.
Whittaker, E., Woodland, P., 2000. Particle-based language modelling. In: Proceedings of the 6th International Conference on Spoken Language Processing (ICSLP), pp. 170–173.
Willett, D., Neukirchen, C., Rigoll, G., 2000. Ducoder - the Duisburg University LVCSR stackdecoder. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 1555–1558.