Induction of a Simple Morphology for Highly-Inflecting Languages

Mathias Creutz and **Krista Lagus**
Neural Networks Research Centre
Helsinki University of Technology
P.O.Box 5400, FIN-02015 HUT, Finland
{Mathias.Creutz, Krista.Lagus}@hut.fi

Abstract

This paper presents an algorithm for the unsupervised learning of a simple morphology of a natural language from raw text. A generative probabilistic model is applied to segment word forms into morphs. The morphs are assumed to be generated by one of three categories, namely prefix, suffix, or stem, and we make use of some observed asymmetries between these categories. The model learns a word structure, where words are allowed to consist of lengthy sequences of alternating stems and affixes, which makes the model suitable for highly-inflecting languages. The ability of the algorithm to find real morpheme boundaries is evaluated against a gold standard for both Finnish and English. In comparison with a state-of-the-art algorithm the new algorithm performs best on the Finnish data, and on roughly equal level on the English data.

1 Introduction

We are intrigued by the endeavor of devising artificial systems that are capable of learning natural language in an unsupervised manner. As untagged text data is available in large quantities for a large number of languages, unsupervised methods may be applied much more widely, or with much lower cost, than supervised ones.

Some languages, such as Finnish, Turkish, and Swahili, are highly-inflecting. We wish to use this term in a wide sense including many kinds of processes for word forming, e.g., compounding and derivation. Their essential challenge for natural language applications arises from the very large number of possible word forms, which causes problems of data sparsity. For instance, creating extensive word lists is not a feasible strategy for obtaining good coverage on the vocabulary necessary for a general dictation task in automatic speech recognition. Instead, a model of the language should incorporate regularities of how words are formed; cf. e.g., (Siivola et al., 2003; Hacioglu et al., 2003).

We will now focus on methods that try to induce the morphology of a natural language from raw text, that is, on algorithms that learn in an unsupervised manner how words are formed. If a human were to learn a language in an analogous way, this would correspond to being exposed to a stream of large amounts of language without observing or interacting with the world where this language is produced. This is clearly not a realistic assumption about language learning in humans. However, Saffran et al. (1996) show that adults are capable of discovering word units rapidly in a stream of a nonsense language, where there is no connection to a meaning of the discovered word-like units. This suggests that humans do use distributional cues, such as transition probabilities between sounds, in language learning. And these kinds of statistical patterns in language data can be successfully exploited by appropriately designed algorithms.

Existing morphology learning algorithms are commonly based on the Item and Arrangement model, i.e., words are formed by a concatenation of morphemes, which are the smallest meaning-bearing units in language. The methods segment words, and the resulting segments are supposed to be close to linguistic morphemes. In addition to producing a segmentation of words the aim is often to discover structure, such as knowledge of which word forms belong to the same inflectional paradigm.

Typically, generative models are used, either formulated in a Bayesian framework, e.g., (Brent, 1999; Creutz, 2003); or applying the Minimum Description Length (MDL) principle, e.g., (de Marcken, 1996; Goldsmith, 2001; Creutz and Lagus, 2002). There is another approach, inspired by the works of Zellig Harris, where a morpheme boundary is suggested at locations where the predictability of the next letter in a letter sequence is low, cf. e.g., (Déjean, 1998).

As it is necessary to learn both which segments are plausible morphemes and what sequences of morphemes are possible, the learning task is al-

huumori	n	taju	ttom	uute	nne
humor	of	sense	-less	-ness	your

Figure 1: Morpheme segmentation of the Finnish word 'huumorintajuttomuutenne' ("your lack of sense of humor").

leviated by making simplifying assumptions about word structure. Often words are assumed to consist of one stem followed by one, possibly empty, suffix as in, e.g., (Déjean, 1998; Snover and Brent, 2001). In (Goldsmith, 2001) a recursive structure is proposed, such that stems can consist of a sub-stem and a suffix. Also prefixes are possible. Other algorithms (Creutz and Lagus, 2002; Creutz, 2003) have been developed for highly-inflecting languages, such as Finnish, where words can consist of lengthy sequences of alternating stems and affixes (see Fig. 1 for an example). These resemble algorithms that segment text without blanks (or transcribed speech) into words, e.g., (de Marcken, 1996; Brent, 1999), in that they do not distinguish between stems and affixes, but split words into so called *morphs*, which carry no explicit category information.

Some algorithms do not rely on the Item and Arrangement (IA) model, but learn relationships between words by comparing the orthographic similarity of pairs of words. In (Neuvel and Fulop, 2002), a morphological learner based on the theory of Whole Word Morphology is outlined. Full words are related to other full words, and complex word forms are analyzed into a variable and non-variable component. Conceivably, in this framework non-concatenative morphological processes, such as umlaut in German, should not be as problematic as in the IA model.

Other algorithms combine information of both orthographic and semantic similarity of words (Schone and Jurafsky, 2000; Baroni et al., 2002). Semantic similarity is measured in terms of similar word contexts. If two orthographically similar words occur in the context of roughly the same set of other words they probably share the same base form, e.g. German 'Vertrag' vs. 'Verträgen' (treaty).

Further cues for morphological learning are presented in (Schone and Jurafsky, 2001) and (Yarowsky and Wicentowsky, 2000). The latter utilizes frequency distributions over different inflectional forms (e.g., how often an English verb occurs in its past tense form in comparison to its base form). The algorithm is not entirely unsupervised.

However, none of these non-IA models suits highly-inflecting languages as they assume only two or three constituents per word, analogous to stem and suffix. In order to cope with a broader range of languages we would need the following: On the one hand, words should be allowed to consist of any number of alternating stems and affixes, making the model more flexible than, e.g., the model in (Goldsmith, 2001). On the other hand, in contrast with (Creutz and Lagus, 2002; Creutz, 2003), sequential dependencies between morphs, i.e., morphotactics, should be taken into account in order to reduce the error rate.

We present a model that incorporates both of these aspects. Experiments show that the new algorithm is able to obtain considerable improvements over the segmentation produced by the algorithm described in (Creutz, 2003). Moreover, it performs better than a state-of-the-art morphology-learning algorithm, Linguistica (Goldsmith, 2001), when evaluated on Finnish data. In the evaluation, the ability of the algorithms to detect morpheme boundaries are measured against a gold standard for both Finnish and English, languages with rather different types of word structure.

2 A probabilistic category-learning algorithm

2.1 Linguistic assumptions

We use a Hidden Markov Model (HMM) to model morph sequences. Previously, HMM's have been used extensively for, e.g., segmentation or tagging purposes. The challenge in this task lies in knowing neither the segments (morphs), nor their tags (categories) in advance. To make the task easier, we utilize the following linguistic assumptions, formulated probabilistically:

(a) Categorial assumption. We assume that with respect to sequential behavior, morphs fall into two main categories, stems and affixes. Affixes are further divided into prefixes and suffixes.

(b) Impossible category sequences. We want to be able to cope with languages with extensive compounding and many consecutive affixes, but not with just any sequence. In particular, we do not wish to allow that a suffix may start a word, or that a prefix end it. Moreover, a prefix should not be followed directly by a suffix. These restrictions are captured by the following regular expression:

```
word = ( prefix* stem suffix* )+
```
(1)

Note that no assumptions are made regarding whether the language is more likely to employ prefixation or suffixation.

(c) Likely properties of morphs in each category. Grammatical affixes mainly carry syntactic information. We therefore assume that affixes are likely to be common "general-purpose" morphs that can be used in connection with a large number of other morphs. By contrast, the set of stems is much larger and there are a considerable number of rare stems that mainly carry semantic information. In order for all stems to be distinguishable from each other they are not likely to be very short morphs.

2.2 Probabilistic generative model for word formation

We use an HMM to assign probabilities to each possible segmentation of a word form. The word is segmented into morphs, each of which belongs to one category: prefix, suffix, or stem. We assume a first-order Markov chain, i.e., a bigram model, for the morph categories. For each category, there is a separate probability distribution over the set of possible morphs. The probability of a particular segmentation of the word w into the morph sequence $\mu_1\mu_2 \ldots \mu_k$ is thus:

$$p(\mu_1\mu_2 \ldots \mu_k \mid w) = \Big[\prod_{i=1}^{k} p(C_i \mid C_{i-1}) \cdot p(\mu_i \mid C_i)\Big] \cdot p(C_{k+1} \mid C_k). \quad (2)$$

The bigram probability of a transition from one morph category to another is expressed by $p(C_i \mid C_{i-1})$. For instance, the probability of observing a stem after a prefix would be written as $p(\text{STM} \mid \text{PRE})$. The probability of observing the morph μ_i when the category C_i is given is expressed by $p(\mu_i \mid C_i)$. The categories C_0 and C_{k+1} represent word boundaries. That is, we take into account the transition from a word boundary to the first morph in the word, as well as the transition from the last morph to a word boundary.

Note also that a morph can be generated from several categories, e.g., a particular morph can function as a stem or a suffix depending on the context.

2.3 The algorithm step by step

The algorithm involves the following steps: (i) production of a baseline segmentation, (ii) initialization of $p(\mu_i \mid C_i)$ and $p(C_i \mid C_{i-1})$, (iii) removal of redundant morphs, and (iv) removal of noise morphs. All steps involve a modification of the morph segmentations, except step (ii), where the probability distributions are initialized.

Steps (ii)–(iv) are all concluded with a re-estimation of the probabilities by means of Expectation-Maximization (EM): The words segmented into morphs are re-tagged using the Viterbi algorithm by maximizing Equation 2. The probabilities $p(\mu_i \mid C_i)$ and $p(C_i \mid C_{i-1})$ are then re-estimated from the tagged data. This is repeated until convergence of the probabilities.

Step (iv) is further followed by a final pass of the Viterbi algorithm, which re-splits and tags the words. Viterbi re-splitting improves the segmentation somewhat, but it is much slower than mere tagging. Therefore Viterbi re-splitting is only performed at the final stage.

2.3.1 Baseline segmentation

A good initial morph segmentation is obtained by the baseline segmentation method (Creutz and Lagus, 2002). The choice of baseline segmentation was motivated by the fact that we wanted the best possible segmentation that suits highly-inflecting languages. The baseline algorithm is based on a probabilistic model that learns a set of morphs, or a morph lexicon, that contains the most likely "building blocks" of the word forms observed in the corpus used as data. The learning process is guided by two prior probability distributions, a prior distribution on morph lengths and a prior distribution on morph frequencies, i.e., the balance between frequent and rare morphs.

2.3.2 Initialization of the probability distributions

Given an initial baseline morph segmentation and initial category membership probabilities $p(C_i \mid \mu_k)$ for each segment (morph), random sampling of this distribution can be utilized to obtain specific tags for the morphs. From the tagged segmentation we can estimate the desired values $p(C_j \mid C_i)$ and $p(\mu_k \mid C_i)$.

Below we describe how the initial category membership probabilities $p(C_i \mid \mu_k)$ emerge. These are probabilities that a particular morph is a prefix, suffix, or stem. In addition, during the process a temporary noise category is utilized, to hold segments that cannot be considered as prefix, suffix, or stem.

Identifying plausible affixes and stems. To identify plausible affixes in our corpus we take the baseline splitting and collect information on the contexts that every discovered morph occurs in. More specifically, we assume that a morph is likely to be a prefix if it is difficult to predict what the following morph is going to be. That is, there are many possible right contexts of the morph. Correspondingly, a morph is likely to be a suffix if it is difficult to predict what the preceding morph can be.

We use *perplexity* to measure the predictability of the preceding or following morph in relation to a specific target morph. The following formula can be used for calculating the left perplexity of a target morph μ:

$$\textit{Left-ppl}(\mu) = \Big[\prod_{\nu_i \,\in\, \text{Left-of}(\mu)} p(\nu_i \mid \mu) \Big]^{-\frac{1}{N_\mu}}. \quad (3)$$

There are N_μ occurrences of the target morph μ in the corpus. The morph tokens ν_i occur to the left of, immediately preceding, the occurrences of μ. The probability distribution $p(\nu_i \mid \mu)$ is calculated over all such ν_i. Right perplexity can be computed analogously.

Next, we implement a graded threshold of suffix-likeness by applying a sigmoid function to the left perplexity of the morphs.

$$\textit{Suffix-like}(\mu) \;=\; [1 + e^{-a\cdot(\textit{Left-ppl}(\mu)-b)}]^{-1}. \quad (4)$$

The parameter b is the perplexity threshold, which indicates the point where a morph μ is as likely to be a suffix as a non-suffix. The parameter a governs the steepness of the sigmoid. The equations for prefix are identical except that right perplexity is applied instead of left perplexity.

As for stems, we assume that the stem-likeness of a morph correlates positively with the *length* in letters of the morph. We employ a sigmoid function as above, which yields:

$$\textit{Stem-like}(\mu) \;=\; [1 + e^{-c\cdot(\textit{Length}(\mu)-d)}]^{-1}, \quad (5)$$

where d is the length threshold and c governs the steepness of the curve.

Initial probability of a morph belonging to a category. Prefix-, suffix- and stem-likeness assume values between zero and one, but they are no probabilities, since they usually do not sum up to one.

In order to create a probability distribution, we first introduce a fourth category besides prefixes, suffixes and stems. This category is the *noise* category and corresponds to cases where *none* of the proper morph classes is likely. Typically noise morphs arise as a consequence of over-segmentation of rare word forms in the baseline word splitting.

We set the probability of a morph being noise (NOI) to:

$$\begin{aligned} p(\text{NOI} \mid \mu) &= [1 - \textit{Prefix-like}(\mu)] \\ &\cdot [1 - \textit{Suffix-like}(\mu)] \cdot [1 - \textit{Stem-like}(\mu)]. \quad (6) \end{aligned}$$

We then distribute the remaining probability mass proportionally between prefix (PRE), suffix (SUF), and stem (STM), e.g.:

$$p(\text{SUF} \mid \mu) = \frac{\textit{Suffix-like}(\mu) \cdot [1 - p(\text{NOI} \mid \mu)]}{\textit{Prefix-like}(\mu) + \textit{Suffix-like}(\mu) + \textit{Stem-like}(\mu)}. \quad (7)$$

2.3.3 Removal of redundant morphs

As a result of applying the baseline segmentation algorithm, there are possibly many redundant morphs, that is, morphs that consist of other discovered morphs. Each morph is studied. If it is possible to split it into two other known morphs, the most probable split is selected and the redundant morph is removed. The most probable split is determined as:

$$\arg \max_{\mu_1, C_1, \mu_2, C_2} p(\mu_1 \mid C_1) \cdot p(C_2 \mid C_1) \cdot p(\mu_2 \mid C_2), \quad (8)$$

where C_1 and C_2 are morph categories, and μ_1 and μ_2 are substrings of the redundant morph μ, such that the concatenation of μ_1 and μ_2 yields μ.

However, some restrictions apply: Splitting into “noise morphs” is not allowed. Furthermore, forbidden category transitions are not allowed to emerge, such as a direct transition from a prefix to a suffix without going through a stem. Nor is splitting into sub-morphs with very low probability accepted.

2.3.4 Removal of noise morphs

As noise morphs are mainly very short morphs and a product of over-segmentation in the baseline splitting algorithm, they are removed by joining with either of the adjacent morphs. The new morph is then labeled as noise. This is repeated until the resulting morph can qualify as a stem, which is determined by the Equation 5. The following heuristics are applied: Joining with shorter morphs is preferred, and joining noise with noise or a stem is always preferred to joining with a prefix or a suffix. These priorities are motivated by the observation that noise morphs tend to be fragments of what should be a stem.

3 Evaluation

We have produced gold standard segmentations with marked morpheme boundaries for 1.4 million Finnish and 36 000 English word forms. We evaluate the segmentations produced by our splitting algorithm against the gold standard, and compute precision and recall on discovered morpheme boundaries. Precision is the proportion of correct boundaries among all morph boundaries suggested by the algorithm. Recall is the proportion of correct boundaries discovered by the algorithm in relation to all morpheme boundaries in the gold standard.

The gold standard was created semi-automatically, by first running all words through a morphological analyzer based on the two-level morphology of Koskenniemi (1983).[1] For each

[1]The software was licensed from Lingsoft, Inc. <http:

word form, the analyzer outputs the base form of the word together with grammatical tags indicating, e.g., the part-of-speech, case, or derivational type of the word form. In addition, the boundaries between the constituents of compound words are often marked. We thoroughly investigated the correspondence between the grammatical tags and the corresponding morphemes and created a rule-set for segmenting the original word forms with the help of the output of the analyzer.

As there can sometimes be many plausibly correct segmentation of a word we supplied several alternatives when needed, e.g., English 'evening' (time of day) vs. 'even+ing' (verb). We also introduced so called "fuzzy" boundaries between stems and endings, allowing some letter to belong to either the stem or ending, when both alternatives are reasonable, e.g., English 'invite+s' vs. 'invit+es' (cf. 'invit+ing'), or Finnish 'tähde+n' vs. 'tähd+en' ("of the star"; the base form is 'tähti').[2]

4 Experiments

We report experiments on Finnish and English corpora. The new category-learning algorithm is compared to two other algorithms, namely the baseline segmentation algorithm presented in (Creutz, 2003), which was also utilized for initializing the segmentation in the category-learning algorithm, and the Linguistica algorithm (Goldsmith, 2001).[3]

4.1 Data sets

The Finnish corpus consists of mainly news texts from the CSC (The Finnish IT Center for Science)[4] and the Finnish News Agency. The corpus consists of 32 million words and it was divided into a development set and a test set, each containing 16 million words. For experiments on English we have used the Brown corpus[5]. It contains one million words, divided into a development set of 250 000 words and a test set of 750 000 words.

The development sets were utilized for optimizing the algorithms and for selecting parameter values, whereas the test sets were used solely in the final evaluation.

	Finnish	English
Word tokens	Word types	Word types
10 000	5 500	2 400
50 000	20 000	7 400
250 000	65 000	20 000
16 000 000	1 100 000	–

Table 1: Sizes of the Finnish and English test sets.

The algorithms were evaluated on different subsets of the test set to produce the precision-recall curves in Figure 2. The sizes of the subsets are shown in Table 1. As can be seen, the Finnish and English data sets contain the same number of word tokens (words of running text), but the number of word types (distinct word forms) is higher in the Finnish data. The word type figures are important, since what was referred to as a 'corpus' in the previous sections is actually a word list. That is, one occurrence of each distinct word form in the data is picked for the morphology learning task.

The word forms in the test sets for which there are no gold standard segmentations are simply left out of the evaluation. The proportions of such word forms are 5%, 6%, 8%, and 15% in the Finnish sets of size 10 000, 50 000, 250 000 and 16 million words, respectively. For English the proportions are 5%, 9%, and 14% for the data sets (in growing order).

4.2 Parameters

The development sets were used for setting the values of the parameters of the algorithms. As a criterion for selecting the optimal values, we used the (equally weighted) F-measure, which is the harmonic mean of the precision and recall of detected morpheme boundaries. For each data size and language separately, we selected the configuration yielding the best F-measure on the development set. These values were then fixed and utilized when evaluating the performance of the algorithms on the test set of corresponding size.

In the Baseline algorithm, we optimized the prior morph length distribution. The prior morph frequency distribution was left at its default value.

The Category algorithm has four parameters: a, b, c, and d; cf. Equations 4, and 5. The constant values $c = 2$, $d = 3.5$ work well for every data set size and language, as does the relation $a = 10/b$. The perplexity threshold, b, assumes values between 5 and 100 depending on the data set. Conveniently, the algorithm is robust with respect to the value of b and the result is always better than that of the Baseline algorithm, except for values of b that are orders

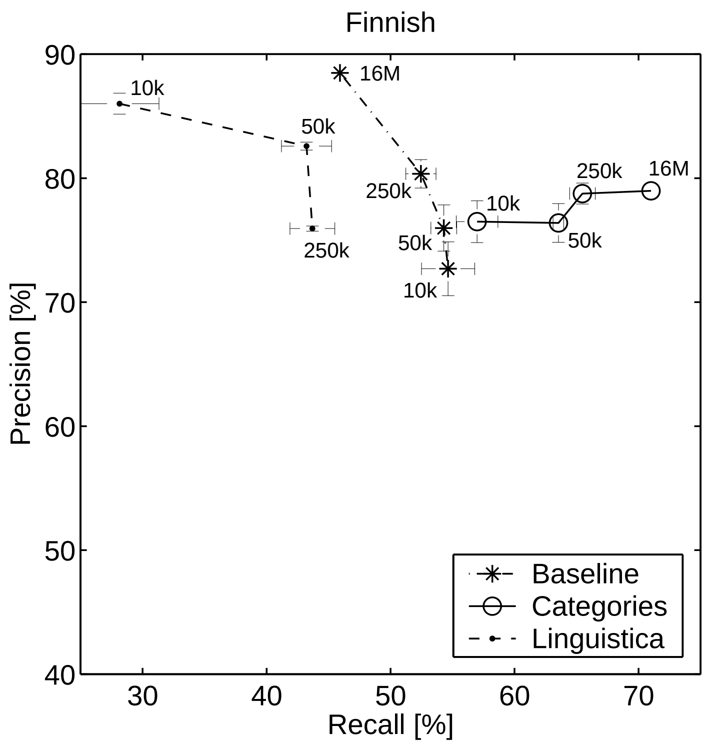
//www.lingsoft.fi>.

[2]Our gold standard segmentations for Finnish and English words are not public, but we are currently investigating the possibility of making them public.

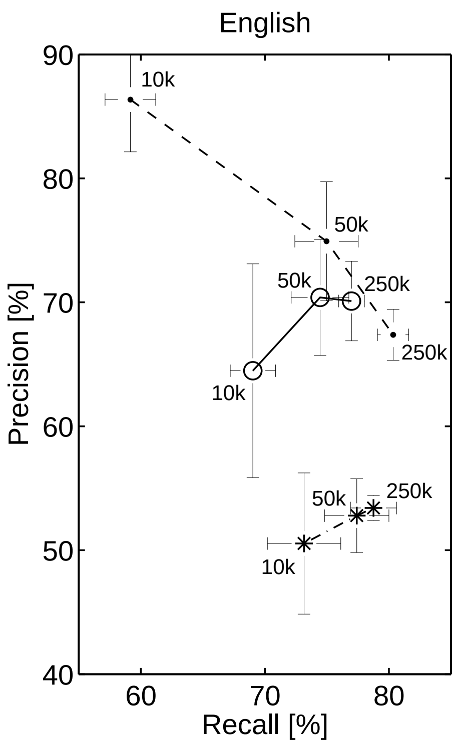
[3]We used the December 2003 version of the software, available at <http://humanities.uchicago.edu/faculty/goldsmith/Linguistica2000/>.

[4]<http://www.csc.fi/kielipankki/>

[5]Available at the Linguistic Data Consortium: <http://www.ldc.upenn.edu>



(a)



(b)

Figure 2: Precision and recall of the three algorithms on test sets of increasing sizes on both Finnish (a) and English (b) data. Each data point is an average of 4 runs on separate test sets, with the exception of the 16M (16 million) words for Finnish (with 1 test set), and the 250k (250 000) words for English (3 test sets). In these cases the lack of test data constrained the number of runs. The standard deviations of the averages are shown as intervals around the data points. There is no 16M data point for Linguistica on Finnish, because the algorithm is very memory-consuming and we could not run it on larger data sizes than 250 000 words on our PC. In most curves, when the data size is increased, recall also rises. An exception is the Baseline curve for Finnish, where precision rises, while recall drops.

of magnitude too large.

In the Linguistica algorithm, we used the commands ‘Find suffix system’ and ‘Find prefixes of suffixal stems’.

4.3 Results

Figure 2 depicts the precision and recall of the algorithms on test sets of different sizes.

When studying the curves for Finnish (Fig. 2a), we observe that the Baseline and Category algorithms perform on a similar level on the smallest data set (10k). However, from there the performances diverge: the Category algorithm improves on both precision and recall, whereas the Baseline algorithm displays a strong increase in precision while recall actually decreases. This means that words are split less often but the proposed splits are more often correct. This is due to measuring the cost of both the lexicon and the data in the optimization function: with a much larger corpus (more data) the optimal solution contains a much larger morph lexicon. Hence, less splitting ensues. The effect is not seen on the English data (Fig. 2b), but this might be due to the smaller corpus sizes.

For Linguistica, an increase in the amount of data is reflected in higher recall, but lower precision. Linguistica only suggests a morpheme boundary between a stem and an affix, if the same stem has been observed in combination with at least one other affix. This leads to a “conservative word-splitting behavior”, with a rather low recall for small data sets, but with high precision. As the amount of data increases, the sparsity of the data decreases, and more morpheme boundaries are suggested. This results in higher recall, but unfortunately lower precision. As Linguistica was not designed for discovering the boundaries within compound words, it misses a large number of them.

For Finnish, the Category algorithm is better than the other two algorithms when compared on data sets of the same size. We interpret a result to be better, even though precision might be somewhat lower, if recall is significantly higher (or vice versa). As an example, for the 16 million word set, the category algorithm achieves 79.0% precision and 71.0% recall. The Baseline achieves 88.5% precision but only 45.9% recall. T-tests show significant differences at the level of 0.01 between all algorithms on Finnish, except for Categories vs. Baseline at 10 000 words.

For English, the Baseline algorithm generally performs worst, but it is difficult to say which of the other two algorithm performs best. According to T-tests there are no significant differences at the level of 0.05 between the following: Categories vs. Linguistica (50k & 250k), and Categories vs. Baseline (10k). However, if one were to extrapolate from the current trends to a larger data set, it would seem likely that the Category algorithm would outperform Linguistica.

4.4 Computational requirements

The Baseline and Category algorithms are implemented as Perl scripts. On the Finnish 250 000 word set, the Baseline algorithm runs in 45 minutes, and the Category algorithm additionally takes 20 minutes on a 900 MHz AMD Duron processor with a maximum memory usage of 20 MB. The Linguistica algorithm is a compiled Windows program, which uses 500 MB of memory and runs in 90 minutes, of which 80 minutes(!) are taken up by the saving of the results.

5 Discussion

It is worth remembering that the gold standard splitting used in these evaluations is based on a traditional morphology. If the segmentations were evaluated using a real-world application, perhaps somewhat different segmentations would be most useful. For example, the tendency to keep common words together, seen in the Baseline model and generally in Bayesian or MDL-based models, might not be at all troublesome, e.g., in speech recognition or machine translation applications. In contrast, excessive splitting might be a problem in both applications.

When compared to the gold standard segmentation used here, the Baseline algorithm produces three types of errors that are prominent: (i) excessive segmentation especially when trained on small amounts of data, (ii) too little segmentation especially with large amounts of data, and (iii) erroneous segments suggested in the beginning of words due to the fact that the same segments frequently occur at the end of words (e.g. ‘s+wing’). The Category algorithm is able to clearly reduce these types of errors due to its following properties: (i) the joining of noise morphs with adjacent morphs, (ii) the removal of redundant morphs by splitting them into sub-morphs, and (iii) the simple morphotactics involving three categories (stem, prefix, and suffix) implemented as an HMM. Furthermore, (iii) is necessary for being able to carry out (i) and (ii).

The Category algorithm does a good job in finding morpheme boundaries and assigning categories to the morphs, as can be seen in the examples in Figure 3, e.g., ‘photograph+er+s’, ‘un+expect+ed+ly’, ‘aarre+kammio+i+ssa’ (“in treasure chambers”; ‘i’ is a plural marker and ‘ssa’ marks the inessive case), ‘bahama+saar+et’ (“[the] Bahama islands”; ‘saari’ means “island” and ‘saaret’ is the plural form). The reader interested in the analyses of other words can try our on-line demo at `http://www.cis.hut.fi/projects/morpho/`.

It is nice to see that the same morph can be tagged differently in different contexts, e.g. ‘pää’ is a prefix in ‘pää+aihe+e+sta’ (“about [the] *main* topic”), whereas ‘pää’ is a stem in ‘pää+hän’ (“in [the] *head*”). In this case the morph categories also resolve the semantic ambiguity of the morph ‘pää’. Occasionally, the segmentation is correct, but the category tagging differs from the linguistic convention, e.g., ‘taka+penkki+läis+et’ (“[the] ones in [the] back seat”), where ‘läis’ is tagged as a stem instead of a suffix.

The segmentation of ‘pääaiheesta’ is not entirely correct: ‘pää+aihe+e+sta’ contains an superfluous morph (‘e’), which should be part of the stem, i.e., ‘pää+aihee+sta’. This mistake is explained by a comparison with the plural form ‘pää+aihe+i+sta’, which is correct. As the singular and plural only differ in one letter, ‘e’ vs. ‘i’, the algorithm has found a solution, where the alternating letter is treated as an independent “number marker”: ‘e’ for singular, ‘i’ for plural.

In the Linguistica algorithm, stems and suffixes are grouped into so called signatures, which can be thought of as inflectional paradigms: a certain set of stems goes together with a certain set of suffixes. Words will be left unsplit unless the potential stem and suffix fit into a signature. As a consequence, if there is only the plural of some particular English noun in the data, but not the singular, Linguistica will not split the noun into a stem and the plural ‘s’, since this does not fit into any signature. In this respect, our category-based algorithm is better at coping with data sparsity. For highly-inflecting languages, such as Finnish, this is especially important.

In contrast with Linguistica, our algorithms can incorrectly “overgeneralize” and suggest a suffix,

aarre + kammio + *i* + *ssa*	jäädy + *ttä* + *ä*	abandon	long + *est*
aarre + kammio + *i* + *sta*	jäädy + *ttä* + *ä* + *kseen*	abandon + *ed*	long + fellow + *'s*
aarre + kammio + *ita*	jäädy + *ttä* + *isi*	abandon + *ing*	longish
aarre + kammio + *nsa*	maclare + *n*	abandon + *ment*	long + itude
aarre + kammio + *on*	nais + auto + *ili* + *ja*	beauti + *ful*	master + piece + *s*
aarre + kammio + *t*	nais + auto + *ili* + *ja* + *a*	beauti + fully	micro + organ + *ism* + *s*
aarre + kammio + *ta*	nais + auto + *ili* + joista	beauty + *'s*	near + *ly*
bahama + saar + *et*	prot + *e* + *iin* + *eja*	calculat + *ed*	necess + *ary*
bahama + saari + *en*	prot + *e* + *iin* + *i*	calculat + *ion* + *s*	necess + ities
bahama + saari + *lla*	prot + *e* + *iin* + *ia*	con + figur + *ation*	necess + *ity*
bahama + saari + *lle*	pää + aihe + *e* + *sta*	con + firm + *ed*	photograph
bahama + saar + *ten*	pää + aihe + *i* + *sta*	express + *ion* + *ist*	photograph + *er* + *s*
edes + autta + *isi* + *vat*	pää + *hän*	express + *ive* + *ness*	photograph + *y*
edes + autta + *ko* + *on*	pää + *kin*	fanatic + *ism*	phrase + *d*
edes + autta + *maan*	pää + *ksi*	invit + *ation* + *s*	phrase + ology
edes + autta + *ma* + *ssa*	taka + penkki + *lä* + *in*+ *en*	invit + *e*	phrase + *s*
haap + *a* + koske + *a*	taka + penkki + läis + *et*	invit + *ed*	sun + rise
haap + *a* + koske + *en*	voida + *kaan*	invit + *e* + *es*	thanks + giving
haap + *a* + koske + *lla*	voi + *mme* + *ko*	invit + *es*	un + avail + *able*
haap + *a* + koski	voisi + *mme*	invit + *ing*	un + expect + *ed* + *ly*

Figure 3: Examples of segmentations learned from the large Finnish data set and a small English data set. Discovered stems are underlined, suffixes are *slanted*, and prefixes are rendered in the standard font.

where there is none, e.g., 'maclare+n' ("MacLaren"). Furthermore, nonsensical sequences of suffixes (which in other contexts are true suffixes) can be suggested, e.g., 'prot+e+iin+i', which should be 'proteiini' ("protein"). A model with more fine-grained categories might reduce such shortcomings in that it could model morphotactics more accurately.

Another aspect requiring attention in the future is allomorphy. Currently each discovered segment (morph) is assigned a role (prefix, stem, or suffix), but no further "meaning" or relation to other morphs. In Figure 3 there are some examples of allomorphs, morphs representing the same morpheme, i.e., morphs having the same meaning but used in complementary distributions. The current algorithm has no means for discovering that 'on' and 'en' mark the same case, namely illative, in 'aarre+kammio+on' ("into [the] treasure chamber") and 'haap+a+koske+en' ("to Haapakoski"). [6] To enable such discovery in principle, one would probably need to look at contexts of nearby words, not just the word-internal context. Additionally, one should allow the learning of a model with richer category structure. Moreover, 'on' and 'en' do not always mark the illative case. In 'bahama+saari+en' the genitive is marked as 'en', and in 'edes+autta+ko+on' ("may he/she help") 'on' marks the third person singular. Similar examples can be found for English, e.g., 'ed' and 'd' are allomorphs in 'invit+ed' vs. 'phrase+d', and so are 'es' and 's' in 'invit+es' vs. 'phrase+s'. However, the meaning of 's' is often ambiguous. It can mark either the plural of a noun or the third person singular of a verb in the present tense. But this kind of ambiguity is in principle solvable in the current model; the Category algorithm resolves similar, also semantic, ambiguities occurring between the three current categories: prefix, stem, and suffix.

[6]Furthermore the algorithm cannot deduce that the illative is actually realized as a vowel lengthening + 'n': 'kammioon' vs. 'koskeen'.

6 Conclusions

We described an algorithm that differs from earlier morpheme segmentation algorithms in that it models dependencies between morph categories in sequences of arbitrary length. Even a simple model with few categories, namely prefix, suffix, and stem is able to capture relevant dependencies that considerably improve the obtained segmentation on both Finnish and English, languages with rather different types of word structure. An interesting future direction is whether the application of more complex model structures may lead to improvements in the morphology induction task.

Acknowledgments

We are grateful to Krister Lindén, as well as the anonymous reviewers for their valuable and thorough comments on the manuscript.

References

M. Baroni, J. Matiasek, and H. Trost. 2002. Unsupervised learning of morphologically related words based on orthographic and semantic similarity. In *Proc. Workshop on Morphological & Phonological Learning of ACL'02*, pages 48–57.

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

M. Creutz and K. Lagus. 2002. Unsupervised discovery of morphemes. In *Proc. Workshop on Morphological and Phonological Learning of ACL'02*, pages 21–30, Philadelphia, Pennsylvania, USA.

M. Creutz. 2003. Unsupervised segmentation of words using prior distributions of morph length and frequency. In *Proc. ACL'03*, pages 280–287, Sapporo, Japan.

C. G. de Marcken. 1996. *Unsupervised Language Acquisition*. Ph.D. thesis, MIT.

H. Déjean. 1998. Morphemes as necessary concept for structures discovery from untagged corpora. In *Workshop on Paradigms and Grounding in Natural Language Learning*, pages 295–299, Adelaide.

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

K. Hacioglu, B. Pellom, T. Ciloglu, O. Ozturk, M. Kurimo, and M. Creutz. 2003. On lexicon creation for Turkish LVCSR. In *Proc. Eurospeech'03*, pages 1165–1168, Geneva, Switzerland.

K. Koskenniemi. 1983. *Two-level morphology: A general computational model for word-form recognition and production*. Ph.D. thesis, University of Helsinki.

S. Neuvel and S. A. Fulop. 2002. Unsupervised learning of morphology without morphemes. In *Proc. Workshop on Morphological & Phonological Learning of ACL'02*, pages 31–40.

J. R. Saffran, E. L. Newport, and R. N. Aslin. 1996. Word segmentation: The role of distributional cues. *Journal of Memory and Language*, 35:606–621.

P. Schone and D. Jurafsky. 2000. Knowledge-free induction of morphology using Latent Semantic Analysis. In *Proc. CoNLL-2000 & LLL-2000*, pages 67–72.

P. Schone and D. Jurafsky. 2001. Knowledge-free induction of inflectional morphologies. In *Proc. NAACL-2001*.

V. Siivola, T. Hirsimäki, M. Creutz, and M. Kurimo. 2003. Unlimited vocabulary speech recognition based on morphs discovered in an unsupervised manner. In *Proc. Eurospeech'03*, pages 2293–2296, Geneva, Switzerland.

M. G. Snover and M. R. Brent. 2001. A Bayesian model for morpheme and paradigm identification. In *Proc. 39th Annual Meeting of the ACL*, pages 482–490.

D. Yarowsky and R. Wicentowsky. 2000. Minimally supervised morphological analysis by multimodal alignment. In *Proc. ACL-2000*, pages 207–216.