Using Unsupervised Paradigm Acquisition for Prefixes

Daniel Zeman
Ústav formální a aplikované lingvistiky
Univerzita Karlova
Malostranské náměstí 25
CZ-11800 Praha, Czechia
zeman@ufal.mff.cuni.cz

Abstract

We describe a simple method of unsupervised morpheme segmentation of words in an unknown language. All what is needed is a raw text corpus (or a list of words) in the given language. The algorithm identifies word parts occurring in many words and interprets them as morpheme candidates (prefixes, stems and suffixes). New treatment of prefixes is the main innovation over Zeman (2007). After filtering out spurious hypotheses, the list of morphemes is applied to segment input words. Official Morpho Challenge 2008 evaluation is given along with some additional experiments evaluated unofficially. We also analyze and discuss errors with respect to the evaluation method.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: I.2.7 Natural Language Processing; I.2.6 Learning; H.3 [**Information Storage and Retrieval**]: H.3.3 Information Search and Retrieval

General Terms

Algorithms, Experimentation, Languages

Keywords

Morphology, Morphological analysis, Unsupervised methods

1 Introduction

Morphological analysis (MA) is an important step in natural language processing, needed by subsequent processes such as parsing or translation. Unsupervised approaches to MA are important in that they help process less studied (and corpus-poor) languages, where we have small or no machine-readable dictionaries and tools. Ideally, an unsupervised morphological analyzer (UMA) would learn how to analyze a language just by looking at a large text in that language, without any additional resources, not even mentioning an expert or speaker of the language.

Supervised approaches to MA can provide us with three types of information: 1. segmentation of the word into *morphemes,* i.e. smallest units bearing lexical or grammatical meaning; 2. functional explanation of the grammatical morphemes, often expressed in a *morphological tag* (e.g. the PDT[1] tag `AAMS2----3N----` would mean that all the grammatical morphemes together set the features gender = masculine, number = singular, case = genitive (2), degree = superlative (3) and negation = negative); 3. lexical anchoring of the lexical morpheme(s) using the *lemma* (part-of-speech information, although lexical, is usually encoded in the tag). Unsupervised approaches cannot use dictionaries nor do they know about arbitrary labels such as *singular* or *genitive* that people have attached to morphemes. Thus, neither lexical nor grammatical explanation of the morphemes is possible. What unsupervised methods can attempt, however, is the proper morpheme segmentation. Algorithms may not know that a particular morpheme denotes plural, they may even disagree in where exactly the morpheme boundary lies but they can figure out that a particular morpheme A occurs at the end of a whole range of words (a linguist would say that all those

[1] Prague Dependency Treebank (Böhmová et al., 2003)

words are in plural), and that it is optional, i.e. the words occur without the morpheme as well (in singular, the linguist would comment).

In many languages, the morphemes are classified as *stems* and *affixes*, the latter being further subclassified as *prefixes* (preceding stems) and *suffixes* (following stems). A frequent word pattern consists of one stem, bearing the lexical meaning, with zero, one or more prefixes (bearing lexical or grammatical meaning) and zero, one or more suffixes (bearing often grammatical meaning). In languages such as German, *compound words* containing more than one stem are quite frequent. While a stem can appear without any affixes, affixes hardly appear on their own, without stems. For the purposes of this paper, a morphological *paradigm* is a collection of affixes that can be attached to the same group of stems, plus the set of affected stems.

Although the segmentation of the word does not provide any linguistically justified explanation of the components of the word, the output can still be useful for further processing of the text. Having got a paradigm, we can generate all unseen morpheme combinations satisfying that paradigm. We can recognize stems of new words. Thus, we are able to group all words with the same stem. The hope is that one stem means one lexical meaning. All words in a group will share the lexical meaning and differ in the grammatical one. By dropping just part of the meaning (hopefully the less important) we reduce the data sparseness of more complex models like syntactic parsing, machine translation, search and information retrieval.

Zeman (2007) contains a comprehensive overview of related work. In addition, there are 7 new papers on this topic by Bernhard (2007), Bordag (2007), Chan (2007), McNamee (2007), Monson et al. (2007), Pitler and Keshava (2007), and Tepper (2007). The approach described in this paper is a direct extension of Zeman (2007) and we will frequently refer to him.

The rest of the paper is organized as follows: in Section 2, we review the method of Zeman (2007) for stem-suffix learning. In Section 3, the proper morpheme segmentation (identifying the learned morphemes in new words) is explained. The main innovation, learning and identifying prefixes using two different methods, is described in Section 4. Section 6 presents results of experiments, Section 7 brings some examples from the data to see what is wrong and why. Section 8 discusses some other problems that are left for future work.

2 Learning stems and suffixes

We begin with reviewing the paradigm acquisition first described in Zeman (2007). The algorithm searches for positions where words can be cut in two parts. In accordance with the motivation, the first part is called the *stem* and the second part is called the *suffix*. An important feature of each split is that the stem occurs in training data with multiple suffixes (or without any suffix) and the suffix has been observed with multiple stems. Otherwise, the algorithm would be just collecting words with coincidentally identical parts.

In the first step, all possible segmentations of every word are generated.

Example: The word *bank* can be segmented as *bank, ban+k, ba+nk, b+ank*.

We remember all stems and suffixes. For each stem, we remember all co-occurring suffixes, and for each suffix we remember all co-occurring stems. A group of suffixes with a common set of stems is called a *paradigm*.

Various techniques are applied to filter out spurious paradigm candidates (see Zeman (2007) for more details and examples):

1. If there are more suffixes than stems in a paradigm, the paradigm is removed.
2. If all suffixes in a paradigm begin with the same letter, there is another paradigm where the letter is part of the stem. The rule is to prefer longer stems and shorter suffixes. It means that the paradigm where the border letter is part of stems will be preserved, and the one where the letter is in suffixes will be removed.
3. If the suffixes of paradigm B form a subset of suffixes of paradigm A (A $\supset$ B) and there is no C, different from A, such that B is also subset of C ($\forall C \neq A : \left(B \not\subset C\right)$), we add the stems of B to the stems of A, and remove B. A subset paradigm is merged with its superset, as long as there is only one superset candidate. As mentioned in Zeman (2007), the process of identifying subsets is computationally quite expensive. We replaced the algorithm used by Zeman (2007) by a new one based on dynamic programming. Starting at the longest suffix sets, we gradually identify their possible subsets by dropping one element at a time. The resulting graph of superset-subset relations contains suffix sets that do not occur in any paradigm. However, building it is linear in original number of paradigms and their mean size. Traversing the graph is trivial and relatively few steps are required to find the closest real superset paradigms. In the case of ~ 69,000 English paradigms, where the old approach required billions of hash queries, we now observed just about 600,000 steps together for constructing and query-

ing the graph. The new algorithm makes the method capable of processing more data in less time, allowing for morphologically more complex languages and/or more benevolent filtering in the preceding steps.

4. Paradigms with only one suffix are removed.

Zeman (2007) showed examples of largest paradigms for all four Morpho Challenge 2007 languages plus Czech. We illustrate the paradigms on a language that is new in this year's Morpho Challenge: **Arabic.** The first line of each example contains suffixes, the second line shows stems.

هم ,ها ,ه ,نـا ,ك ,ا ,0
ديـون إنفـــاق أوراق أهداف أمور أموال أطفـال

هما ,هم ,ها ,ه ,نـا ,ا ,0
نفـــوذ مخاوف قبـــول جنـود تأييـــد ارتيـــاح أسـعار

تهمـا ,تهـم ,تهـا ,تـه ,تنـــا ,ت ,0
نشـاطا منتجـــا مســـتحقا عمليـــا طـائرا صـادرا خلافـــا تصـــريحا اســـتثمارا

The final set of paradigms yields the lists of known stems and suffixes. The information what stem can occur with what suffix is also available. Note however, that due to subset merging, the expression "can occur" is no longer equivalent to "has occurred in training data". Also, some words are covered by no known stem and/or suffix because all their segmentations were removed during the paradigm filtering.

The three lists (known stems, known suffixes, and known stem-suffix pairs) are the output of the learning phase. They are now used to identify morphemes in new words.[2]

3 Morphematic segmentation

Given the lists obtained during training, we want to find the stem-suffix boundary in a word of the same language. We can also find out that the word is a stem without any suffix.

Again, we consider all possible segmentations of each analyzed word. For each stem-suffix pair, we look up the table of learned stems and suffixes. The following cases are possible:

1. Both stem and suffix are known and allowed to occur together.
2. Both parts are known but they are not known to occur together.
3. Only the stem is known.
4. Only the suffix is known.
5. Neither the stem nor the suffix is known.

Zeman (2007) did not take the case 1 into account. If both parts were known (case 2), he marked the segmentation as *certain*. If only one part was known (cases 3 and 4), he marked the segmentation as *possible*. After trying all segmentations, if one or more were certain, they were returned as competing analyses of the word. Only if there were no certain segmentation, the possible segmentations were returned. If there were no possible segmentation either, the whole word was returned as a single morpheme.

4 Learning and segmenting prefixes

The main weakness of the 2007 approach is that it assumes one or two morphemes per word. There is no means of correctly segmenting words that contain both prefixes and suffixes,[3] and compound words. In the present work, we explored two ways of identifying prefixes in addition to the stem-suffix splitting. Both methods work separately from the stem-suffix learning, so after learning a separate list of prefixes, modified segmentation algorithm has to be employed.

[2] In fact, the lists can be used to segment any word, whether new or old, whether or not from the training data. As the method is unsupervised, and the "training data" is merely a word list, there is no reason why training data should be less suitable for testing than any other list of words from the given language.

[3] Note that it is theoretically possible for a word that contains only one prefix and one stem to be segmented using the old approach: the prefix would be taken for stem and the stem for suffix. However, the filtering rules are set up not in favor of such solutions.

4.1 Reversed word method

The least expensive prefix-learning approach seems to be to take the whole apparatus and apply it on reversed words (right-to-left). The strings that the system marks as suffixes are reversed again and marked as prefixes. Problem is, we now have two sets of stems: one from the suffix learning, one from learning of prefixes. For instance, the English word *un+beat+able* yields the stems *unbeat* and *beatable*, respectively. The following algorithm uses the four lists (prefixes, prefix-stems, suffix-stems and suffixes) to find one or more segmentations of a word:

1. For all split points, check whether the left part is a known prefix and the right part is a known prefixed stem. (This corresponds to *certain* segmentations of Zeman (2007).) If so, remember the prefix as applicable.
2. For all split points, check whether the left part is a known suffixed stem and the right part is a known suffix. (This corresponds to *certain* segmentations of Zeman (2007).) If so, remember the suffix as applicable.
3. Remember also empty prefix and empty suffix as applicable.
4. Loop over all combinations of applicable prefixes and suffixes. Make sure that they do not overlap in the word and that at least one character of the word remains to play the role of the stem. Save segmentations found this way.
5. If at least one segmentation into two or more parts has been found, remove the "dummy" segmentation (whole word is a stem).

4.2 Rule-based method

Since it became clear that there are many problems bound to the Reversed word method, we started experimenting with a more conservative approach. We defined a prefix as follows:

1. A prefix is formed by 1 to K word-initial characters.
2. Minimal length of the stem (the remainder of the word after removing the prefix) is L.
3. The prefix occurs at least with N stems for which the following condition holds.
4. The stems with which the prefix occurs also occur without the prefix or with another prefix. The number of different prefixes (including the empty one) seen with the common stem must be at least M.

We set K = 5, L = 2, M = 5 and N = 100.

The algorithm to find prefixes is simple. First split each word in up to K positions, observing conditions 1 and 2, and generate the initial set of prefix candidates. Then loop over them and discard those not complying with conditions 3 and 4. This process typically needs to be repeated iteratively because discarding a prefix decreases number of prefixes at other stems, which in turn could invalidate another prefix, although it already passed first check up. Note that the prefixes counted in condition 4 have to conform to the definition themselves. We observed that the process converged rather quickly. For instance, the English data generated 119,000 first-round candidates. Only 678 candidates passed to the second round, and the set converged to 665 prefixes after four rounds.

We would prefer to get even smaller set of prefixes but were unable to find better setup. Either the system discarded all candidates, or at least the real prefixes did not survive (while some garbage did). For more details, see Error analysis.

A few other comments on the method: We further revised the morphematic segmentation based on prefixes. We ignored the stems found with prefixes. We took the stem-suffix segmentation found by Zeman (2007) and just looked for a known prefix in the beginning of the stem. If we found it, the prefix was made a separate morpheme (regardless whether the stem was actually seen with this prefix).

Also note that a group of several prefixes can occur in a word. This fact is ignored during training and it can happen that two consecutive prefixes are learned as one prefix. We could also look for the known prefixes repetitively; however, due to the noise and very short suspicious prefixes in the set, we believe that this would cause more harm than good.

5 Other modifications

All hyphens are replaced by spaces at the end of the segmentation phase. This adds morpheme boundaries unless the hyphen already is at a morpheme boundary. Hyphens almost always occur in compound words or after some generic English prefixes.

6 Results

Results are compared to a gold standard segmentation created by a supervised morphological analyzer. The evaluation method must reflect the limitations of unsupervised approaches, thus the only information being compared with the gold standard is the fact that two words share a morpheme with the same label.[4] List of pairs of words and the morphemes they share is created first for both the gold standard and the output of the unsupervised analyzer. The next step is computing **precision** (what portion of the morphemes shared in system output were shared correctly, i.e. were also shared in the gold standard?) and **recall** (what portion of the morphemes shared in gold standard were also shared in system output?) The **F** score is the harmonic mean of precision and recall:

$$P = \frac{P + R}{2PR}$$

The results were evaluated by the official Morpho Challenge 2008 evaluation scripts, `sample_word_pairs.pl` and `eval_morphemes.pl`.[5] Some experiments were evaluated using the whole gold standard data set (available only to the challenge organizers; this is called the *official* evaluation) while other were evaluated using only a subset of 500 gold standard words (available to everyone; this is called the *unofficial* evaluation). Morpho Challenge 2008 involved variously-sized data of 5 languages: English (384,903 words), German (1,266,159), Finnish (2,206,719), Turkish (617,298) and Arabic (143,966). The complete results are available at http://www.cis.hut.fi/morphochallenge2008/results.shtml.

We submitted two sets of results. The method of Zeman (2007) was submitted as method 1 to this year's competition. The data submitted as method 3 (from "three morphemes") resulted from the reversed word method described in Section 4.1. Here it is marked "of. rev. prf.", while the unofficial results of the rule-based method from Section 4.2 are marked "un. rul. prf." This experiment also contains the hyphen segmentation described in Section 5.

	English		
	P	R	F
official	52.98	42.07	46.90
unofficial	53.39	44.53	48.56
of. rev. prf.	76.92	8.47	15.27
un. rul. prf.	54.88	46.72	50.48

	German		
	P	R	F
official	53.12	28.37	36.98
unofficial	30.28	25.47	27.67
of. rev. prf.	72.27	7.15	13.01
un. rul. prf.	46.02	37.92	41.58

	Finnish		
	P	R	F
official	58.51	20.47	30.33
unofficial	47.44	22.99	30.97
of. rev. prf.	72.41	3.42	6.54
un. rul. prf.	46.94	35.18	40.22

	Turkish		
	P	R	F
official	65.81	18.79	29.23
unofficial	59.46	22.53	32.68
of. rev. prf.	73.30	3.01	5.79
un. rul. prf.	60.57	37.21	46.10

	Arabic		
	P	R	F
official	77.24	12.73	21.86
unofficial	79.86	8.76	15.78
of. rev. prf.	89.62	5.18	9.79
un. rul. prf.	80.90	13.41	23.00

The main result of the experiment is the comparison of the two methods for finding prefixes. The reversed word method generally brings very high precision and very low recall and F-score. The rule-based method, on the other hand, is an improvement over both the official and the unofficial baseline results. It generally decreases precision as a price for improving both recall and F-score.

[4] Even the order of the morphemes is not significant, although Zeman (2007) stated the contrary.
[5] The detailed description of the evaluation method and the scripts for download are at http://www.cis.hut.fi/morphochallenge2008/evaluation.shtml.

The processing of one language took from about 5 minutes (Arabic) to almost 1 hour (Finnish) on a 64bit AMD Opteron.

7 Error analysis

The training data is noisy and contains many typos. Our system does not use up the word frequency statistics that could help to filter out noise. The damaging impact that noise can cause can be illustrated on the group of English words *abrupt – abruptly – abruptness – *abrupty*. The first three form a well understood pattern that occurs with a few hundred other stems *(absent-minded, aimless, anxious, artless, assertive… etc.)* The fourth word, *abrupty,* is in fact a misspelled version of *abruptly*. The typo in the suffix prevented this group from being included in the main paradigm. Typos are infrequent (compared to correct material) and only one other stem exhibited this particular one: *explicit*. As a result, the group formed a separate paradigm and got filtered out on the rule 1 (more suffixes than stems).

Since the rule was introduced to exclude spurious paradigms with one one-letter stem and thousands of suffixes, we tried to weaken it and allow paradigms that have N times more suffixes than stems for a small N (≤5). However, the change decreased recall as well as the overall F-score, so we did not include it in the final experiments.

About two thirds of English words do not survive the paradigm filtering. There are 385 Kwords on input and 122 thousand segmentations on output. (In fact, more must have been dropped, as some new segmentations were introduced during the subset merging.)

Two-way checking of morphemes in the reversed word method caused the high precision and extremely low recall for all languages. The learned paradigms look reasonable, although they are not too large. Some examples are: (English) *three-, two-, four-, 0 + decade-old, foot-thick, fifths, hour-plus; 0, re, re-, over + capitalise, stimulate, tighten, commit;* (German) *südo, nordo, o, südwe, nordwe, we + stprovinz, sthorizont, stchinesischen, stpolnischen, stafrikanischen, stzipfel, stdeutsche, stengland, stchina; 0, ab, ein, aus, zu + gewanderter, gewanderten, wanderungsdruck, gewanderte, wanderungswelle.* The first German example well illustrates that it would make more sense to put the border characters to the prefix instead of the stem. The reversed word method found 3,279 English prefixes, 12,585 German, 20,537 Finnish, 5,127 Turkish and 261 Arabic.

The rule-based prefix method is generally more restrictive in learning prefixes, although the segmentation approach we combined with it is more benevolent. The rule-based algorithm learned 665 English, 1,890 German, 4,628 Finnish, 2,178 Turkish and 331 Arabic. There are three kinds of prefixes: 1. very short strings that are probably not true prefixes but are too frequent to be filtered *(a, a-, aa, abf, abg, ac, ag, ah, ai, ak...)*; 2. real prefixes (English *anti, anti-, auto, by, co, co-, dis, ex, mis, re, un, …;* German *ab, an, anti, anti-, anzu, auf, aufge, aufzu, aus, ausge, be, dar, ...)*; 3. first parts of compounds (English *ash, back, bank, bell, down, five-, half, …;* German *abend, acht, aids, aids-, akten, alarm, alpen, ...)*.

8 Ways to improve

Some options have already been mentioned: using word frequencies to eliminate typos, more or less strict stem matching in the segmentation phase. There are alternatives to the most strict matching (stem and suffix must have occurred together). For instance, instead of requiring that the stem and the suffix occur together, we can ask whether the stem occurred with N other suffixes that co-occur with the tried suffix in at least one paradigm. Another option is to try to figure out whether the word can belong to a paradigm that allows for such suffix. For example, the strictest method did not split *a-com's* to *a-com* and *'s*, although the word *a-com* was in training data and *'s* is part of many paradigms. However, this particular segmentation got discarded in the filtering phase.

A better approach to compound words is needed. The hyphen rule did a good job for English, and the prefix processing is able to separate first parts of compounds in many instances. However, there are many other compounds that are not solved satisfactorily. Either their first part is longer than the prefix threshold, or they have more than two parts.

The naming of morphemes matters! Even when the way how the evaluation works is designed not to depend on the labels, we are still responsible for giving same labels to same things. If we fail to recognize that "-d" and "-ed" is essentially the same English morpheme (which can easily happen if they came from different paradigms), we will be penalized in F-score. A related problem is learning phonological changes that occur in some languages on the stem-affix border.

The border letters that occur in all words of a paradigm (and can thus be assigned to both stems or suffixes) can trouble subset merging mechanisms. For instance, the largest German paradigm has suffixes *0, m, n, r, re, rem, ren, rer, res, s*. All stems of this paradigm end in *e*. There is another paradigm with suffixes *0, e, em, en, er, ere, es*. Here the *e* must remain in suffixes because of the only exception, the empty suffix *0*. The two paradigms cannot be merged. However, if the *e* in the former paradigm was shifted to the suffixes, merging would be trivial and immediate. The question is, how do we know that in this particular case the bordering letters should be treated differently?

Compound suffixes are currently treated as one suffix. The German suffixes *er, ere, eres, erem, eren* should in fact be treated as pairs of suffixes *er+0, er+e, er+es, er+em, er+en*. We could search for a subset of the suffix set that is contained twice, once with a prefix and once without it. Testing of this condition would have to be fuzzy.

9 Conclusion

We have presented a paradigm acquisition method that can be used for unsupervised segmentation of words into morphemes. The main improvement over Zeman (2007) is the unsupervised learning of prefixes. Compounds and typos are the most important yet-to-be-addressed issues.

10 Acknowledgements

This research has been supported by the following projects: Czech Academy of Sciences, the "Information Society" program, project No. 1ET101470416, and Ministry of Education of the Czech Republic, project MSM0021620838.

11 References

Delphine Bernhard. 2007. *Simple Morpheme Labelling in Unsupervised Morpheme Analysis*. In: Working Notes for the Cross Language Evaluation Forum (CLEF) 2007 Workshop, Budapest, Hungary.

Alena Böhmová, Jan Hajič, Eva Hajičová, Barbora Hladká. 2003. *The Prague Dependency Treebank: A Three-Level Annotation Scenario*. In: Anne Abeillé (ed.): Treebanks: Building and Using Syntactically Annotated Corpora. Kluwer Academic Publishers, Dordrecht, The Netherlands.

Stefan Bordag. 2007. *Unsupervised and Knowledge-free Morpheme Segmentation and Analysis.* In: Working Notes for the Cross Language Evaluation Forum (CLEF) 2007 Workshop, Budapest, Hungary.

Erwin Chan. 2007. *Towards Unsupervised Induction of Morphonemic Rules.* In: Working Notes for the Cross Language Evaluation Forum (CLEF) 2007 Workshop, Budapest, Hungary.

Paul McNamee, James Mayfield. 2007. *N-Gram Morphemes for Retrieval.* In: Working Notes for the Cross Language Evaluation Forum (CLEF) 2007 Workshop, Budapest, Hungary.

Christian Monson, Jaime Carbonell, Alon Lavie, Lori Levin. 2007. *ParaMor: Finding Paradigms across Morphology.* In: Working Notes for the Cross Language Evaluation Forum (CLEF) 2007 Workshop, Budapest, Hungary.

Emily Pitler, Samarth Keshava. 2007. *A Segmentation Approach to Morpheme Analysis.* In: Working Notes for the Cross Language Evaluation Forum (CLEF) 2007 Workshop, Budapest, Hungary.

Michael A. Tepper. 2007. *Using Hand-Written Rewrite Rules to Induce Underlying Morphology.* In: Working Notes for the Cross Language Evaluation Forum (CLEF) 2007 Workshop, Budapest, Hungary.

Daniel Zeman. 2007. *Unsupervised Acquiring of Morphological Paradigms from Tokenized Text.* In: Working Notes for the Cross Language Evaluation Forum (CLEF) 2007 Workshop, Budapest, Hungary. ISSN 1818-8044. Revised version to appear in C. Peters et al. (eds.): CLEF 2007, LNCS 5152, pp. 892–899, Springer-Verlag, Berlin / Heidelberg, Germany, 2008.