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 that 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 in comparison to [1]. After filtering out spurious hypotheses, the list of morphemes is applied to segment input words. Official Morpho Challenge 2008 evaluation is given together with some additional experiments. Processing of prefixes improved the F-score by 5 to 11 points for German, Finnish and Turkish, while it failed to improve English and Arabic. We also analyze and discuss errors with respect to the evaluation method.

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 understudied (and corpus-poor) languages, for which we have inadequate machine-readable dictionaries and tools. Ideally, an unsupervised morphological analyzer (UMA) would learn to analyze a language by looking at a large text in that language, without any additional resources, not to mention an expert or speaker of the language.

Supervised approaches to MA can provide us with three types of information: 1. segmentation of a word into *morphemes,* i.e. the smallest units bearing lexical or grammatical meaning; 2. functional explanation of grammatical morphemes, often expressed in a *morphological tag* (e.g. the PDT [2] 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 morphemes – a *lemma* (part-of-speech information, although lexical, is usually encoded in the tag). Unsupervised approaches cannot use dictionaries nor do they know about labels such as *singular* or *genitive* that people have attached to morphemes. Thus, neither lexical nor grammatical explanation is possible for any morpheme. What unsupervised methods can attempt is morpheme segmentation. Algorithms for such segmentation may not know if a particular morpheme denotes plural,

C. Peters et al. (Eds.): CLEF 2008, LNCS 5706, pp. 983–990, 2009.

they may even disagree where exactly a morpheme boundary lies. But they can figure out that a particular morpheme occurs at the end of a certain set of words (a linguist would say that all these words are plural), and that it can be optional, i.e. a word may occur without it (identified as singular by the linguist).

In many languages, morphemes are classified as *stems* and *affixes* with latter being further subclassified as *prefixes* (preceding stems) and *suffixes* (following stems). A common word pattern consists of a stem (bearing the lexical meaning) and, optionally, some prefixes (bearing lexical or grammatical meaning) and/or suffixes (often bearing grammatical meaning). In a language such as German, *compound words* containing more than one stem are quite frequent. While a stem can appear without any affix, affixes hardly appear on their own. In 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 a word does not provide any linguistically justified explanation for any of its components, the output can still be useful for further processing. Having got a paradigm, we can generate all unseen morpheme combinations within that paradigm. We can recognize stems of new words and then group all words with the same stem. The hope is that a stem may have a lexical meaning. All words in a group will share this lexical meaning and differ grammatically. By dropping some part of the meaning (hopefully the less important part) we reduce the data sparseness of more complex models for syntactic parsing, machine translation, and information retrieval.

A comprehensive overview of related work is given in [1]. In addition, there are several new papers on UMA describing results of the previous Morpho Challenge. The approaches in [3], [4] and [7] utilize segment predictability and transition probabilities. [5] models character N-grams to find word stems. [8] is a semi-supervised method using a limited set of hand-written rules. [6] seems to be the most similar approach to ours, which is a direct extension of [1].

The rest of the paper is organized as follows: in Sections 2 and 3, we review the method of [1] for stem-suffix learning. The main innovation in learning and identifying prefixes using two different methods is described in Section 4. The rest of the paper presents our experiments and discusses their results.

2 Learning Stems and Suffixes

We begin with reviewing the paradigm acquisition that was first described in [1]. The algorithm searches for positions to cut words each into two parts: the *stem* and the *suffix*. An important feature of such a split is that a stem occurs in training data with an arbitrary number of suffixes and a suffix occurs with multiple stems. Otherwise, the algorithm would just collect words with coincidentally identical parts.

In the first step, all possible segmentations of every word are generated. For instance, the word *bank* can be segmented as *bank, ban+k, ba+nk, b+ank*. We collect all stems and suffixes. For each stem, we store all co-occurring suffixes, and for each suffix we keep all co-occurring stems.

Various techniques are applied to filter out spurious paradigm candidates (see [1] 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 in which the border letter is part of stems will be preserved, and the other 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* ($\forall C \neq A : (B \not\subset C)$), 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 [1], the process of identifying subsets is computationally quite expensive. We replaced the algorithm used in [1] by a new one based on dynamic programming. Starting with 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 approx. 69,000 English paradigms, the old approach required billions of hash queries, whereas now we need only about 600,000 steps together for constructing and querying 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.

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.

3 Morphemic 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 does not have 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.

If both parts were known (case 2), the procedure from [1] marked the segmentation as *certain*. If only one part was known (cases 3 and 4), the segmentation was labeled as *possible*. If there were certain segmentations, they were returned as competing analyses of the word. Otherwise, the possible segmentations were returned. If there was no possible segmentation either, the whole word was returned as a single morpheme.

4 Learning and Detecting Prefixes

The main weakness of the approach in [1] is that it assumes one or two morphemes per word. There is no means of correctly segmenting words that contain both prefixes and suffixes, 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.

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. The problem is that we now have two sets of stems: one from the suffix learning, one from the 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 [1].) 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 [1].) If so, remember the suffix as applicable.
3. Remember also the empty prefix and the empty suffix as applicable.
4. Loop over all combinations of applicable prefixes and suffixes. Make sure that they do not overlap and that at least one character of the word remains to play the role of stem. Save segmentations found this way.
5. If at least one morpheme boundary has been found, remove the "dummy" segmentation (which marks the whole word as a stem).

4.2 Rule-Based Method

The other approach we tested defines a prefix using the following set of parameterized rules. The values of the four parameters are estimates based on a few experiments. We wanted to keep the approach language-independent, so we did the experiments with English data only, and used the same values for all languages. We set $K = 5$, $L = 2$, $M = 5$ and $N = 100$.

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*.

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 the number of prefixes at other stems, which in turn could invalidate another prefix, although it already passed the 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 Section 6.

A few other comments on the method: We further revised the morphemic segmentation based on prefixes. We ignored the stems found with prefixes. We took the stem-suffix segmentation found by [1] 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).

5 Results

Results are compared to 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. Even the order of the morphemes is not significant, although [1] stated the contrary. 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 then computed the usual way, i.e. $F = (P + R) / 2PR$. For more details on the evaluation procedure, see [9].

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 improves results for three languages. It generally decreases precision as a price for improving both recall and F-score. It is not clear what kind of damage the method caused on English and Arabic. This calls for further investigation because previous evaluation on smaller data suggested improvement on all the languages [10].

Table 1. Results. Next to the language name is the best F-score of the other participants. "Stem+suff" is equivalent to [1] (submitted to MC2008 as "method 1"). "Rev strict" adds the reversed prefix method (submitted as "method 3"). "Rule weak" is unofficial post-deadline result (but evaluated on the same data).

	English (56.26)		
	P	R	F
stem+suff	52.98	42.07	**46.90**
rev strict	76.92	8.47	15.27
rule weak	27.72	62.47	38.40

German (54.06)		
P	R	F
53.12	28.37	36.98
72.27	7.15	13.01
41.75	41.97	**41.86**

	Finnish (48.47)		
	P	R	F
stem+suff	58.51	20.47	30.33
rev strict	72.41	3.42	6.54
rule weak	50.12	35.85	**41.80**

Turkish (51.99)		
P	R	F
65.81	18.79	29.23
73.30	3.01	5.79
52.54	33.43	**40.86**

	Arabic (40.87)		
	P	R	F
stem+suff	77.24	12.73	**21.86**
rev strict	89.62	5.18	9.79
rule weak	68.96	11.20	19.27

Vocabulary sizes:	Ar	144 K
	En	385 K
	Tr	617 K
	De	1.3 M
	Fi	2.2 M

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

6 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 words 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 there was only one other stem *(explicit)* that occurred with the same kind of error. 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 a 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 ($\leq$5). However, the change decreased recall as well as the overall F-score, so we did not include it in our final experiments.

Two-way checking of morphemes in the reversed word method is the probable cause for 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 prefixes. There are three kinds of prefixes: 1. very short strings that are probably not true prefixes but are too frequent to be filtered out *(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, …).*

7 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 strictest 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 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 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.

The border letters that occur in all words of a paradigm 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?

8 Conclusion

Our method can be used for unsupervised segmentation of words into morphemes. The main improvement over [1] is the unsupervised learning of prefixes. Compounds and typos are the most important yet-to-be-addressed issues.

Acknowledgements. My thanks go to the anonymous reviewers for their helpful feedback. Funding: Czech Academy of Sciences, project No. 1ET101470416, and Czech Ministry of Education, project MSM0021620838.

References

1. Zeman, D.: Unsupervised Acquiring of Morphological Paradigms from Tokenized Text. In: Peters, C., Jijkoun, V., Mandl, T., Müller, H., Oard, D.W., Peñas, A., Petras, V., Santos, D. (eds.) CLEF 2007. LNCS, vol. 5152, pp. 892–899. Springer, Heidelberg (2008)
2. Böhmová, A., Hajič, J., Hajičová, E., Hladká, B.: The Prague Dep. Treebank: A Three-Level Annotation Scenario. In: Treebanks: Building and Using.... Kluwer, Dordrecht (2003)
3. Bernhard, D.: Simple Morpheme Labeling in Unsupervised Morpheme Analysis. In: Peters, C., Jijkoun, V., Mandl, T., Müller, H., Oard, D.W., Peñas, A., Petras, V., Santos, D. (eds.) CLEF 2007. LNCS, vol. 5152, pp. 873–880. Springer, Heidelberg (2008)
4. Bordag, S.: Unsupervised and Knowledge-free Morpheme Segmentation and Analysis. In: Peters, C., Jijkoun, V., Mandl, T., Müller, H., Oard, D.W., Peñas, A., Petras, V., Santos, D. (eds.) CLEF 2007. LNCS, vol. 5152, pp. 881–891. Springer, Heidelberg (2008)
5. McNamee, P., Mayfield, J.: N-Gram Morphemes for Retrieval. In: Working Notes for the CLEF Worksh., Budapest, Hungary (2007)
6. Monson, C., Carbonell, J., Lavie, A., Levin, L.: ParaMor: Finding Paradigms across Morphology. In: Peters, C., Jijkoun, V., Mandl, T., Müller, H., Oard, D.W., Peñas, A., Petras, V., Santos, D. (eds.) CLEF 2007. LNCS, vol. 5152, pp. 900–907. Springer, Heidelberg (2008)
7. Pitler, E., Keshava, S.: A Segmentation Approach to Morpheme Analysis. In: Working Notes for the CLEF Worksh., Budapest, Hungary (2007)
8. Tepper, M.A.: Using Hand-Written Rewrite Rules to Induce Underlying Morphology. In: Working Notes for the CLEF Worksh., Budapest, Hungary (2007)
9. Kurimo, M., Turunen, V., Varjokallio, M.: Overview of Morpho Challenge 2008. In: Evaluating Systems for Multiling. and Multimodal Inf. Access – 9th CLEF. LNCS. Springer, Heidelberg (2009)
10. Zeman, D.: Using Unsupervised Paradigm Acquisition for Prefixes. In: Working Notes for the CLEF Worksh., Århus, Denmark (2008)