High-Performance, Language-Independent Morphological Segmentation

Sajib Dasgupta and **Vincent Ng**
Human Language Technology Research Institute
University of Texas at Dallas
Richardson, TX 75083-0688
{sajib,vince}@hlt.utdallas.edu

Abstract

This paper introduces an unsupervised morphological segmentation algorithm that shows robust performance for four languages with different levels of morphological complexity. In particular, our algorithm outperforms Goldsmith's Linguistica and Creutz and Lagus's Morphessor for English and Bengali, and achieves performance that is comparable to the best results for all three PASCAL evaluation datasets. Improvements arise from (1) the use of relative corpus frequency and suffix level similarity for detecting incorrect morpheme attachments and (2) the induction of orthographic rules and allomorphs for segmenting words where roots exhibit spelling changes during morpheme attachments.

1 Introduction

Morphological analysis is the task of segmenting a word into *morphemes*, the smallest meaning-bearing elements of natural languages. Though very successful, *knowledge-based* morphological analyzers operate by relying on manually designed segmentation heuristics (e.g. Koskenniemi (1983)), which require a lot of linguistic expertise and are time-consuming to construct. As a result, research in morphological analysis has exhibited a shift to *unsupervised* approaches, in which a word is typically segmented based on morphemes that are automatically induced from an unannotated corpus. Unsupervised approaches have achieved considerable success for English and many European languages (e.g. Goldsmith (2001), Schone and Jurafsky (2001), Freitag (2005)). The recent PASCAL Challenge on *Unsupervised Segmentation of Words into Morphemes*[1] has further intensified interest in this problem, selecting as target languages English as well as two highly agglutinative languages, Turkish and Finnish. However, the evaluation of the Challenge reveals that (1) the success of existing unsupervised morphological parsers does not carry over to the two agglutinative languages, and (2) no segmentation algorithm achieves good performance for all three languages.

Motivated by these state-of-the-art results, our goal in this paper is to develop an *unsupervised* morphological segmentation algorithm that can *work well across different languages*. With this goal in mind, we evaluate our algorithm on four languages with different levels of morphological complexity, namely English, Turkish, Finnish and Bengali. It is worth noting that Bengali is an under-investigated Indo-Aryan language that is highly inflectional and lies between English and Turkish/Finnish in terms of morphological complexity. Experimental results demonstrate the robustness of our algorithm across languages: it not only outperforms Goldsmith's (2001) Linguistica and Creutz and Lagus's (2005) Morphessor for English and Bengali, but also compares favorably to the best-performing PASCAL morphological parsers when evaluated on all three datasets in the Challenge.

The performance improvements of our segmentation algorithm over existing morphological analyzers can be attributed to our extending Keshava and Pitler's (2006) segmentation method, the best performer for English in the aforementioned

[1] http://www.cis.hut.fi/morphochallenge2005/

PASCAL Challenge, with the capability of handling two under-investigated problems:

Detecting incorrect attachments. Many existing morphological parsers incorrectly segment "candidate" as "candid"+"ate", since they fail to identify that the morpheme "ate" should *not* attach to the word "candid". Schone and Jurafsky's (2001) work represents one of the few attempts to address this inappropriate morpheme attachment problem, introducing a method that exploits the semantic relatedness between word pairs. In contrast, we propose two arguably simpler, yet effective techniques that rely on *relative corpus frequency* and *suffix level similarity* to solve the problem.

Inducing orthographic rules and allomorphs. One problem with Keshava and Pitler's algorithm is that it fails to segment words where the roots exhibit spelling changes during attachment to morphemes (e.g. "denial" = "deny"+"al"). To address this problem, we automatically acquire allomorphs and orthographic change rules from an unannotated corpus. These rules also allow us to output the actual segmentation of the words that exhibit spelling changes during morpheme attachment, thus avoiding the segmentation of "denial" as "deni"+"al", as is typically done in existing morphological parsers.

In addition to addressing the aforementioned problems, our segmentation algorithm has two appealing features. First, it can segment words with any number of morphemes, whereas many analyzers can only be applied to words with one root and one suffix (e.g. DéJean (1998), Snover and Brent (2001)). Second, it exhibits robust performance even when inducing morphemes from a very large vocabulary, whereas Goldsmith's (2001) and Freitag's (2005) morphological analyzers perform well only when a small vocabulary is employed, showing deteriorating performance as the vocabulary size increases.

The rest of this paper is organized as follows. Section 2 presents related work on unsupervised morphological analysis. In Section 3, we describe our basic morpheme induction algorithm. We then show how to exploit the induced morphemes to (1) detect incorrect attachments by using relative corpus frequency (Section 4) and suffix level similarity (Section 5) and (2) induce orthographic rules and allomorphs (Section 6). Section 7 describes our algorithm for segmenting a word using the induced morphemes. We present evaluation results in Section 8 and conclude in Section 9.

2 Related Work

As mentioned in the introduction, the problem of unsupervised morphological learning has been extensively studied for English and many other European languages. In this section, we will give an overview of the related work on this problem.

Harris (1955) develops a strategy for identifying morpheme boundaries that checks whether the number of different letters following a sequence of letters exceeds some given threshold. DéJean (1998) improves Harris's segmentation algorithm by first inducing a list of 100 most frequent morphemes and then using those morphemes for word segmentation. The aforementioned PASCAL Challenge on Unsupervised Word Segmentation has undoubtedly intensified interest in this problem. Among the participating groups, Keshava and Pitler's (2006) segmentation algorithm combines the ideas of DéJean and Harris and achieves the best result for the English dataset, but it only offers mediocre performance for Finnish and Turkish.

There is another class of unsupervised morphological learning algorithms whose design is driven by the Minimum Description Length (MDL) principle. Specifically, EM is used to iteratively segment a list of words using some predefined heuristics until the length of the morphological grammar converges to a minimum. Brent et al. (1995) are the first to introduce an information-theoretic notion of compression to represent the MDL framework. Goldsmith (2001) also adopts the MDL approach, providing a new compression system that incorporates signatures when measuring the length of the morphological grammar. Creutz (2003) proposes a probabilistic maximum *a posteriori* formulation that uses prior distributions of morpheme length and frequency to measure the goodness of an induced morpheme, achieving better results for Finnish but worse results for English in comparison to Goldsmith's Linguistica.

3 The Basic Morpheme Induction Algorithm

Our unsupervised segmentation algorithm is composed of two steps: (1) inducing *prefixes*, *suffixes* and *roots* from a vocabulary that consists of words taken from a large corpus, and (2) segmenting a word using these induced morphemes. This section describes our *basic* morpheme induction method.

3.1 Extracting a List of Candidate Affixes

The first step of our morpheme induction method involves extracting a list of candidate prefixes and suffixes. We rely on a fairly simple idea originally proposed by Keshava and Pitler (2006) for extracting candidate affixes. Assume that α and β are two character sequences and $\alpha\beta$ is the concatenation of α and β. If $\alpha\beta$ and α are both found in the vocabulary, then we extract β as a candidate suffix. Similarly, if $\alpha\beta$ and β are both found in the vocabulary, then we extract α as a candidate prefix.

The above affix induction method is arguably overly simplistic and therefore can generate many spurious affixes. To filter spurious affixes, we (1) score each affix by multiplying its *frequency* (i.e. the number of distinct words to which each affix attaches) and its *length*[2], and then (2) retain only the *K* top-scoring affixes, where *K* is set differently for prefixes and suffixes. The value of *K* is somewhat dependent on the vocabulary size, as the affixes in a larger vocabulary system are generated from a larger number of words. For example, we set the thresholds to 70 for prefixes and 50 for suffixes for English; on the other hand, since the Finnish vocabulary is almost six times larger than that of English, we set the corresponding thresholds to be approximately six times larger (400 and 300 for prefixes and suffixes respectively).[3]

3.2 Detecting Composite Suffixes

Next, we detect and remove *composite suffixes* (i.e. suffixes that are formed by combining multiple suffixes [e.g. "ers" = "er"+"s"]) from our induced suffix list, because their presence can lead to under-segmentation of words (e.g. "walkers", whose correct segmentation is "walk"+"er"+"s", will be erroneously segmented as "walk"+"ers"). Composite suffix detection is a particularly important problem for languages like Bengali in which composite suffixes are abundant (see Dasgupta and Ng (2007)). Note, however, that simple concatenation of multiple suffixes does not always produce a composite suffix. For example, "ent", "en" and "t" all are valid suffixes in English, but "ent" is not a composite suffix. Hence, we need a more sophisticated method for composite suffix detection.

Our detection method is motivated by the following observation: if *xy* is a composite suffix and a word *w* combines with *xy*, then it is highly likely that *w* will also combine with its first component suffix *x*. Note that this property does not hold for non-composite suffixes. For instance, words that combine with the non-composite suffix "ent" (e.g. "absorb") do not combine with its first component suffix "en". Consequently, given two suffixes *x* and *y*, our method posits *xy* as a composite suffix if *xy* and *x* are similar in terms of the words to which they attach. Specifically, we consider *xy* and *x* to be similar if their similarity value as computed by the formula below is greater than 0.6:

$$Similarity\,(xy, x) = P(x \mid xy) = \frac{|W'|}{|W|},$$

where $|W'|$ is the number of distinct words that combine with both *xy* and *x*, and $|W|$ is the number of distinct words that combine with *xy*.

3.3 Extracting a List of Candidate Roots

Finally, we extract a list of candidate roots using the induced list of affixes as follows. For each word, *w*, in the vocabulary, we check whether *w* is *divisible*, i.e. whether *w* can be segmented as *r*+*x* or *p*+*r*, where *p* is an induced prefix, *x* is an induced suffix, and *r* is a word in the vocabulary. We then add *w* to the root list if it is not divisible. Note, however, that the resulting root list may contain *compound* words (i.e. words with multiple roots). Hence, we make another pass over our root list to remove any word that is a concatenation of multiple words in the vocabulary.

4 Detecting Incorrect Attachments Using Relative Frequency

Our induced root list is not perfect: many correct roots are missing due to over-segmentation. For example, since "candidate" and "candid" are in the vocabulary and "ate" is an induced suffix, our root induction method will incorrectly segment "candidate" as "candid"+"ate"; as a result, it does not consider "candidate" as a root. So, to improve the root induction method, we need to determine that the attachment of the morpheme "ate" to the root word "candid" is incorrect. In this section, we propose a simple yet novel idea of using relative cor-

[2] The dependence on frequency and length is motivated by the observation that less-frequent and shorter affixes (especially those of length 1) are more likely to be erroneous (see Goldsmith (2001)).

[3] Since this method for setting our vocabulary-dependent thresholds is fairly simple, the use of these thresholds should not be viewed as rendering our segmentation algorithm language-dependent.

pus frequency to determine whether the attachment of a morpheme to a root word is plausible or not.

Consider again the two words “candidate” and “candid”. While “candidate” occurs 6380 times in our corpus, “candid” occurs only 119 times. This frequency disparity can be an important clue to determining that there is no morphological relation between “candidate” and “candid”. Similar observation is also made by Yarowsky and Wicentowski (2000), who successfully employ relative frequency similarity or disparity to rank candidate VBD/VB pairs (e.g. “sang”/“sing”) that are irregular in nature. Unlike Yarowsky and Wicentowski, however, our goal is to detect incorrect affix attachments and improve morphological analysis.

Our incorrect attachment detection algorithm, which exploits frequency disparity, is based on the following hypothesis: if a word *w* is formed by attaching an affix *m* to a root word *r*, then the corpus frequency of *w* is likely to be less than that of *r* (i.e. the frequency ratio of *w* to *r* is less than one). In other words, we hypothesize that the inflectional or derivational forms of a root word occur less frequently in a corpus than the root itself.

To illustrate this hypothesis, Table 1 shows some randomly chosen English words together with their *word-root frequency ratios* (WRFRs). The <word, root> pairs in the left side of the table are examples of correct attachments, whereas those in the right side are not. Note that only those words that represent correct attachments have a WRFR less than 1.

The question, then, is: to what extent does our hypothesis hold? To investigate this question, we generated examples of correct attachments by randomly selecting 400 words from our English vocabulary and then removing those that are root words, proper nouns, or compound words. We then manually segmented each of the remaining 378 words as Prefix+Root or Root+Suffix with the aid of the CELEX lexical database (Baayean et al., 1996). Somewhat surprisingly, we found that the WRFR is less than 1 in only 71.7% of these attachments. When the same experiment was repeated on 287 hand-segmented Bengali words, the hypothesis achieves a higher accuracy of 83.6%.

To better understand why our hypothesis does not work well for English, we measured its accuracy separately for the Root+Suffix words and the Prefix+Root words, and found that the hypothesis fails mostly on the suffixal attachments (see Table 2). Though surprising at first glance, the relatively poor accuracy on suffixal attachments can be attributed to the fact that many words in English (e.g. “amusement”, “winner”) appear more frequently in our corpus than their corresponding root forms (e.g. “amuse”, “win”). For Bengali, our hypothesis fails mainly on verbs, whose inflected forms occur more often in our corpus than their roots. This violation of the hypothesis can be attributed to the grammatical rule that the main verb of a Bengali sentence has to be inflected according to the subject in order to maintain sentence order.

To improve the accuracy of our hypothesis on detecting correct attachments, we relax our initial hypothesis as follows: if an attachment is correct, then the corresponding WRFR is less than some predefined threshold *t*, where $t > 1$. However, we do not want *t* to be too large, since our algorithm may then determine many incorrect attachments as correct. In addition, since our hypothesis has a high accuracy for prefixal attachments than suffixal attachments, the threshold we employ for prefixes can be smaller than that for suffixes. Taking into account these considerations, we use a threshold of 10 for suffixes and 2 for prefixes for all the languages we consider in this paper.

Correct Parses			**Incorrect Parses**		
Word	**Root**	**WRFR**	**Word**	**Root**	**WRFR**
bear-able	bear	0.01	candid-ate	candid	53.6
attend-ance	attend	0.24	medic-al	medic	483.9
arrest-ing	arrest	0.06	prim-ary	prim	327.4
sub-group	group	0.0002	ac-cord	cord	24.0
re-cycle	cycle	0.028	ad-diction	diction	52.7
un-settle	settle	0.018	de-crease	crease	20.7

Table 1: Word-root frequency ratios

	Root+Suffix	**Prefix+Root**	**Overall**
# of words	344	34	378
WRFR < 1	70.1%	88.2%	71.7%

Table 2: Hypothesis validation for English

Now we can employ our hypothesis to detect incorrect attachments and improve root induction as follows. For each word, *w*, in the vocabulary, we check whether (1) *w* can be segmented as $r+x$ or $p+r$, where *p* and *x* are valid prefixes and suffixes respectively and *r* is another word in the vocabulary, and (2) the WRFR for *w* and *r* is less than our predefined thresholds (10 for suffixes and 2 for prefixes). If both conditions are satisfied, it means that *w* is divisible. Hence, we add *w* into the list of roots if at least one of the conditions is violated.

5 Suffix Level Similarity

Many of the incorrect suffixal attachments have a WRFR between 1 and 10, but the detection algorithm described in the previous section will determine all of them as correct attachments. Hence, in this section, we propose another technique, which we call *suffix level similarity*, to identify some of these incorrect attachments.

Suffix level similarity is motivated by the following observation: if a word *w* combines with a suffix *x,* then *w* should also combine with the suffixes that are "morphologically similar" to *x*. To exemplify, consider the suffix "ate" and the root word "candid". The words that combine with the suffix "ate" (e.g. "alien", "fabric", "origin") also combine with suffixes like "ated", "ation" and "s". Given this observation, the question of whether "candid" combines with the suffix "ate" then lies in whether or not "candid" combines with "ated", "s" and "ation". The fact that "candid" does not combine with many of the above suffixes provides suggestive evidence that "candidate" cannot be derived from "candid".

More specifically, to check whether a word *w* combines with a suffix *x* using suffix level simialrity, we (1) find the set of words W_x that can combine with *x*; (2) find the set of suffixes S_x that attach to all of the words in W_x under the constraint that S_x does not contain *x*; and (3) find the 10 suffixes in S_x that are most "similar" to *x*. The question, then, is how to define similarity. Intuitively, a good similarity metric should reflect, for instance, the fact that "ated" is a better suffix to consider in the attachment decision for "ate" than "s" (i.e. "ated" is more similar to "ate" than "s"), since "s" attaches to most nouns and verbs in English and hence should not be a distinguishing feature for incorrect attachment detection.

We employ a probabilistic measure (PM) that computes the similarity between suffixes *x* and *y* as the product of (1) the probability of a word combining with *y* given that it combines with *x* and (2) the probability of a word combining with *x* given that it combines with *y*. More specifically,

$$PM(x, y) = P(y \mid x) * P(x \mid y) = \frac{n}{n_1} * \frac{n}{n_2},$$

where n_1 is the number of distinct words that combine with *x*, n_2 is the number of distinct words that combine with *y*, and *n* is the number of distinct words that combine with both *x* and *y*.[4]

After getting the 10 suffixes that are most similar to *x*, we employ them as features and use the associated similarity values (we scale them linearly between 1 and 10) as the weights of these 10 features. The decision of whether a suffix *x* can attach to a word *w* depends on whether the following inequality is satisfied:

$$\sum_{1}^{10} f_i w_i > t,$$

where f_i is a boolean variable that has the value 1 if *w* combines with x_i, where x_i is one of the 10 suffixes that are most similar to *x*; w_i is the scaled similarity between *x* and x_i; and *t* is a predefined threshold that is greater than 0.

One potential problem with suffix level similarity is that it is an overly strict condition for those words that combine with only one or two suffixes in the vocabulary. For example, if the word "character" has just one variant in the vocabulary (e.g. "characters"), suffix level similarity will determine the attachment of "s" to "character" as incorrect, since the weighted sum in the above inequality will be 0. To address this sparseness problem, we rely on both relative corpus frequency and suffix level similarity to identify incorrect attachments. Specifically, if the WRFR of a <word, root> pair is between 1 and 10, we determine that an attachment to the root is incorrect if

$$-WRFR + \gamma * (\textit{suffix level similarity}) < 0,$$

where γ is set to 0.15.

Finally, since long words have a higher chance of getting segmented, we do not apply suffix level similarity to words whose length is greater than 10.

6 Inducing Orthographic Rules and Allomorphs

The biggest drawback of the system, described thus far, is its failure to segment words where the roots exhibit spelling changes during attachment to morphemes (e.g. "denial" = "deny"+"al"). The reasons are (1) the system does not have any knowledge of language-specific orthographic rules (e.g. in English, the character 'y' at the morpheme boundary is changed to 'i' when the root combines

[4] Note that this metric has the desirable property of returning a low similarity value for "s": while *n* is likely to be large, it will be offset by a large n_2.

with the suffix "al"), and (2) the vocabulary we employ for morpheme induction does not normally contain the *allomorphic variations* of the roots (e.g. "deni" is allomorphic variation of "deny"). To segment these words correctly, we need to generate the allomorphs and orthographic rules automatically given a set of induced roots and affixes.

Before giving the details of the generation method, we note that the induction of orthographic rules is a challenging problem, since different languages exhibit orthographic changes in different ways. For some languages (e.g. English) rules are mostly predictable, whereas for others (e.g. Finnish) rules are highly irregular. It is hard to obtain a generalized mapping function that aligns every <root, allomorph> pair, considering the fact that our system is unsupervised. An additional challenge is to ensure that the incorporation of these orthographic rules will not adversely affect system performance (i.e. they will not be applied to regular words and thus segment them incorrectly). Yarowsky and Wicentowski (2000) propose an interesting algorithm that employs four similarity measures to successfully identify the most probable root of a highly irregular word. Unlike them, however, our goal is to (1) check whether the learned rules can actually improve an unsupervised morphological system, not just to align <root, allomorph> pair, and (2) examine whether our system is extendable to different languages.

Taking into consideration the aforementioned challenges, our induction algorithm will (1) handle orthographic character changes that occur only at morpheme boundaries; (2) generate allomorphs for suffixal attachments only[5], assuming that roots exhibit the character changes during attachment, not suffixes; and (3) learn rules that aligns <root, allomorph> pairs of edit distance 1 (which may involve 1-character replacement, deletion or insertion). Despite these limitations, we will see that the incorporation of the induced rules improves segmentation accuracy significantly.

Let us first discuss how we learn a *replacement* rule, which identifies <allomorph, root> pairs where the last character of the root is replaced by another character. The steps are as follows:

(1) Inducing candidate allomorphs

If $\alpha A\beta$ is a word in the vocabulary (e.g. "denial", where α="den", A="i", and β="al"), β is an induced suffix, αB is an induced root (e.g. "deny", where B="y"), and the attachment of β to αB is correct according to relative corpus frequency (see Section 4), then we hypothesize that αA is an allomorph of αB. For each induced suffix, we use this hypothesis to generate the allomorphs and identify those that are generated from at least two suffixes as *candidate allomorphs*. We denote the list of <candidate allomorph, root, suffix> tuples by L.

(2) Learning orthographic rules

Every <candidate allomorph, root, suffix> tuple as learned above is associated with an orthographic rule. For example, from the words "denial", "deny" and suffix "al", we learn the rule "y:i / _ + al"[6]; from "social", "sock" and "al", we learn the rule "k:i / _ + al", which, however, is erroneous. So, we check whether each of the learned rules occurs frequently enough for all the <allomorph, root> pairs associated with a suffix, with the goal of filtering the low-frequency orthographic rules. Specifically, for each suffix β, we repeat the following steps:

(a) Counting the frequency of rules. Let L_β be the list of <candidate allomorph, root> pairs in L that are associated with the suffix β. For each pair p in L_β, we first check whether its candidate allomorph appears in any other <candidate allomorph, root> pairs in L_β. If not, we increment the frequency of the orthographic rule associated with p by 1. For example, the pair <"deni", "deny"> increases the frequency of the rule "y:i" by 1 on condition that "deni" does not appear in any other pairs.

(b) Filtering the rules. We first remove the infrequent rules, specifically those that are induced by less than 15% of the tuples in L_β. Then we check whether there exists two rules of the form A:B and A:C in the induced rule list. If so, then we have a morphologically undesirable situation where the character A changes to B and C under the same environment (i.e. β). To address this problem, we first calculate the strength of a rule as follows:

$$strength(A:B) = \frac{frequency(A:B)}{\sum_{@} frequency(A:@)}$$

We then retain only those rules whose frequency*strength is greater than some predefined threshold. We denote the list of rules that satisfy the above constraints by R_β.

(c) Identifying valid allomorphs. For each rule in R_β, we identify the associated <candidate allo-

[5] We only learn rules for suffixes of length greater than 1, since most suffixes of length 1 do not participate in orthographic changes.

[6] This is the Chomsky and Halle notation for representing orthographic rules. *a*:*b* / *c* _ *d* means *a* changes to *b* when the left context is *c* and the right context is *d*.

morph, root> pairs in L_β. We refer to the candidate allomorphs in each of those pairs as *valid allomorphs* and add them to the list of roots. We also remove from the original root list the words that can be segmented by the induced allomorphs and the associated rules (e.g. "denial").

(d) Identifying composite suffixes. For each rule in R_β, we also check whether it can identify composite suffixes where the first component suffix's last character is replaced during attachment to the second component suffix (e.g. "liness" = "ly"+"ness"). Specifically, if (1) $A{:}B$ / _ β is a rule in R_β, (2) $\alpha A\beta$ (say "liness"), β (say "ness") and αB (say "ly") are induced suffixes, and (3) $\alpha A\beta$ satisfies the requirements of a composite suffix (see Section 3.2), then we determine that $\alpha A\beta$ is a composite suffix composed of αB and β.

We employ the same procedure for learning *insertion* and *deletion* rules, except that strength is always set to 1 for these two types of rules. The threshold we set at step (b) is somewhat dependent on the vocabulary size, since the frequency count of each rule will naturally be larger when a larger vocabulary is used. Following our method for setting vocabulary-dependent thresholds (see Section 3.1), we set the threshold to 4 for English and 25 for Finnish, for instance.

Finally, we adapt our candidate allomorph detection method described above to induce allomorphs that are generated through orthographic changes of edit distance greater than 1. Specifically, if $\alpha\beta$ is a word in the induced root list (e.g. "stability"[7], where α="stabil" and β="ity"), β is an induced suffix, and the attachment of β to α is correct according to suffix level similarity, then we hypothesize that α ("stabil") is a candidate allomorph. For each induced suffix, we use this hypothesis to generate candidate allomorphs and consider as valid allomorphs only those that are generated from at least three different suffixes.[8]

7 Word Segmentation

After inducing the morphemes, we can use them to segment a word *w* in the test set. Specifically, we (1) *generate* all possible segmentations of *w* using only the induced affixes and roots, and (2) apply a sequence of tests to *remove* candidate segmentations until we are left with only one candidate, which we take to be the final segmentation of *w*.

Our first test involves removing any candidate segmentation $m_1m_2 \ldots m_n$ that violates any of the linguistic constraints below:

- At least one of $m_1, m_2, \ldots, m_n$ is a root.
- For $1 \le i < n$, if m_i is a prefix, then m_{i+1} must be a root or a prefix.
- For $1 < i \le n$, if m_i is a suffix, then m_{i-1} must be a root or a suffix.
- m_1 can't be a suffix and m_n can't be a prefix.

Next, we apply our second test, in which we retain only those candidate segmentations that have the smallest number of morphemes. For example, if "friendly" has two candidate segmentations "friend"+"ly" and "fri"+"end"+"ly", we will select the first one to be the segmentation of *w*.

If more than one candidate segmentation still exists, we score each remaining candidate using the heuristic below, selecting the highest-scoring candidate to be the final segmentation of *w*. Basically, we score each candidate segmentation by adding the *strength* of each morpheme in the segmentation, where (1) the strength of an affix is the number of distinct words in the vocabulary to which the affix attaches, multiplied by the length of the affix, and (2) the strength of a root is the number of distinct morphemes with which the root combines, again multiplied by the length of the root.

8 Evaluation

In this section, we will first evaluate our segmentation algorithm for English and Bengali, and then examine its performance on the PASCAL datasets.

8.1 Experimental Setup

Vocabulary creation. We extracted our English vocabulary from the Wall Street Journal corpus of the Penn Treebank and the BLLIP corpus, preprocessing the documents by first tokenizing them and then removing capitalized words, punctuations and numbers. In addition, we removed words of frequency 1 from BLLIP, because many of them are proper nouns and misspelled words. The final English vocabulary consists of approximately 60K distinct words. We applied the same pre-processing

[7] The correct segmentation of "stability" is "stable"+"ity". The "stabil"-"stable" allomorph-root pair is an example of an orthographic change of edit distance 2.

[8] This technique can also be used to induce out-of-vocabulary (OOV) roots. For example, the presence of "perplexity", "perplexed" and "perplexing" in a vocabulary allows us to induce the root "perplex". OOV root induction is particularly important for languages like Bengali, where verb roots mostly take the imperative form and hence are absent in a vocabulary created from a newspaper corpus, which normally comprises only the first and third person verb forms.

steps to five years of articles taken from the Bengali newspaper *Prothom Alo* to generate our Bengali vocabulary, which consists of 140K words.

Test set preparation. To create our English test set, we randomly chose 5000 words from our vocabulary that are at least 4-character long[9] and also appear in CELEX. Although 95% of the time we adopted the segmentation proposed by CELEX, in some cases the CELEX segmentations are erroneous (e.g. "rolling" and "lodging" remain unsegmented in CELEX). As a result, we cross-check with the online version of Merriam-Webster to make the necessary changes. To create the Bengali test set, we randomly chose 5000 words from our vocabulary and manually removed proper nouns and misspelled words from the set before giving it to two of our linguists for hand-segmentation. The final test set contains 4191 words.

Evaluation metrics. We use two standard metrics -- *exact accuracy* and *F-score* -- to evaluate the performance of our segmentation algorithm on the test sets. Exact accuracy is the percentage of the words whose proposed segmentation is identical to the correct segmentation. F-score is the harmonic mean of recall and precision, which are computed based on the placement of morpheme boundaries.[10]

8.2 Results for English and Bengali

The baseline systems. We use two publicly available and widely used unsupervised morphological learning systems -- Goldsmith's (2001) Linguistica[11] and Creutz and Lagus's (2005) Morphessor 1.0[12] -- as our baseline systems. The first two rows of Table 3 show the results of these systems for our test sets (with all the training parameters set to their default values). As we can see, Linguistica performs substantially better for English in terms of both exact accuracy and F-score, whereas Morphessor outperforms Linguistica for Bengali.

Our segmentation algorithm. Results of our segmentation algorithm are shown in rows 3-6 of Table 3. Specifically, row 3 shows the results of our basic segmentation system as described in Section 3. Rows 4-6 show the results where our three techniques (i.e. relative frequency, suffix level similarity and allomorph detection) are incorporated into the basic system one after the other. It is worth mentioning that (1) our basic algorithm already outperforms the baseline systems in terms of both exact accuracy and F-score; and (2) while each of our additions to the basic algorithm boosts system performance, relative corpus frequency and allomorph detection contribute to performance improvements particularly significantly. As we can see, the best segmentation performance is achieved when all of our three additions are applied to the basic algorithm.

	English				Bengali			
	A	P	R	F	A	P	R	F
Linguistica	68.9	84.8	75.7	80.0	36.3	58.2	63.3	60.6
Morphessor	64.9	69.6	85.3	76.6	56.5	89.7	67.4	76.9
Basic induction	68.1	79.4	82.8	81.1	57.7	79.6	81.2	80.4
Relative frequency	74.0	86.4	82.5	84.4	63.2	85.6	79.9	82.7
Suffix level similarity	74.9	88.6	82.3	85.3	66.1	89.7	78.8	83.9
Allomorph detection	**78.3**	88.3	86.4	**87.4**	**68.3**	89.3	81.3	**85.1**

Table 3: Results (reported in terms of exact accuracy (A), precision (P), recall (R) and F-score (F))

8.3 PASCAL Challenge Results

To get an idea of how our algorithm performs in comparison to the PASCAL participants, we conducted evaluations on the PASCAL datasets for English, Finnish and Turkish. Table 4 shows the F-scores of four segmentation algorithms for these three datasets: the best-performing PASCAL system (Winner), Morphessor, our system that uses the basic morpheme induction algorithm (Basic), and our system with all three extensions incorporated (Complete). Below we discuss these results.

English. There are 533 test cases in this dataset. Using the vocabulary created as described in Section 8.1, our Complete algorithm achieves an F-score of 79.4%, which outperforms the winner (Keshava and Pitler, 2006) by 2.6%. Although our basic morpheme induction algorithm is similar to that of Keshava and Pitler, a closer examination of the results reveals that F-score increases significantly with the incorporation of relative frequency and allomorph detection.

Finnish and Turkish. The real challenge in the PASCAL Challenge is the evaluation on Finnish

[9] Words of length less than 4 do not have any morphological segmentation in English. Hence, by imposing this length restriction on the words in our test set, we effectively make the evaluation more challenging. This is also the reason for our using words that are at least 3-character long in the Bengali test set.
[10] See http://www.cis.hut.fi/morphochallenge2005/evaluation.shtml for details.
[11] http://humanities.uchicago.edu/faculty/goldsmith/Linguistica2000/
[12] http://www.cis.hut.fi/projects/morpho/

and Turkish due to their morphological richness. We use the 400K and 300K most frequent words from the Finnish and Turkish datasets provided by the organizers as our vocabulary. When tested on the gold standard of 661 Finnish and 775 Turkish words, our Complete system achieves F-scores of 65.2% and 66.2%, which are better than the winner's scores (Bernhard (2006)). In addition, Complete outperforms Basic by 3-6% in F-score; these results suggest that the new techniques proposed in this paper (especially allomorph detection) are also very effective for Finnish and Turkish.

	English	**Finnish**	**Turkish**
Winner	76.8	64.7	65.3
Morphessor	66.2	66.4	70.1
Basic	75.8	59.2	63.4
Complete	79.4	65.2	66.2

Table 4: F-scores for the PASCAL gold standards

As mentioned in the introduction, none of the participating PASCAL systems offers robust performance across different languages. For instance, Keshava and Pitler's algorithm, the winner for English, has F-scores of only 47% and 54% for Finnish and Turkish respectively, whereas Bernhard's algorithm, the winner for Finnish and Turkish, achieves an F-score of only 66% for English. On the other hand, our algorithm outperforms the winners for *all* the languages in the competition, demonstrating its robustness across languages.

Finally, although Morphessor achieves better results for Turkish and Finnish than our Complete system, it performs poorly for English, having an F-score of only 66.2%. On the other hand, our results for Finnish and Turkish are not significantly poorer than those of Morphessor.

9 Conclusions

We have presented an unsupervised word segmentation algorithm that offers robust performance across languages with different levels of morphological complexity. Our algorithm not only outperforms Linguistica and Morphessor for English and Bengali, but also compares favorably to the best-performing PASCAL morphological parsers when evaluated against all three target languages -- English, Turkish, and Finnish -- in the Challenge. Experimental results indicate that the use of relative corpus frequency for incorrect attachment detection and the induction of orthographic rules and allomorphs have contributed to the performance of our algorithm particularly significantly.

References

R. H. Baayen, R. Piepenbrock and L. Gulikers. 1996. The CELEX2 lexical database (CD-ROM), LDC, Univ of Pennsylvania, Philadephia, PA.

D. Bernhard. 2006. Unsupervised morphological segmentation based on segment predictability and word segment alignment. In *PASCAL Challenge Workshop on Unsupervised Segmentation of Words into Morphemes*.

M. R. Brent, S. K. Murthy and A. Lundberg. 1995. Discovering morphemic suffixes: A case study in minimum description length induction. In *Proceedings of the Fifth International Workshop on AI and Statistics*.

M. Creutz. 2003. Unsupervised segmentation of words using prior distributions of morph length and frequency. In *Proceedings of the ACL,* pages 280-287.

M. Creutz and K. Lagus. 2005. Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0. In *Computer and Information Science, Report A81*, Helsinki University of Technology.

S. Dasgupta and V. Ng. 2007. Unsupervised word segmentation for Bangla. In *Proceedings of ICON*, pages 15-24.

H. DéJean. 1998. Morphemes as necessary concepts for structures discovery from untagged corpora. In *Workshop on Paradigms and Grounding in Natural Language Learning*, pages 295-299.

D. Freitag. 2005. Morphology induction from term clusters. In *Proceedings of CoNLL*, pages 128-135.

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

Z. Harris. 1955. From phoneme to morpheme. In *Language*, 31(2): 190-222.

S. Keshava and E. Pitler. 2006. A simpler, intuitive approach to morpheme induction. In *PASCAL Challenge Workshop on Unsupervised Segmentation of Words into Morphemes*.

K. Koskenniemi. 1983. Two-level morphology: a general computational model for word-form recognition and production. *Publication No. 11. Helsinki: University of Helsinki Department of General Linguistics.*

P. Schone and D. Jurafsky. 2001. Knowledge-free induction of inflectional morphologies. In *Proceedings of the NAACL*, pages 183-191.

M. G. Snover and M. R. Brent. 2001. A Bayesian model for morpheme and paradigm identification. In *Proceedings of the ACL*, pages 482-490.

D. Yarowsky and R. Wicentowski. 2000. Minimally supervised morphological analysis by multimodal alignment. In *Proceedings of the ACL*, pages 207-216.