A Simpler, Intuitive Approach to Morpheme Induction

Samarth Keshava
Yale University
New Haven, CT 06520
samarth.keshava@yale.edu

Emily Pitler
Yale University
New Haven, CT 06520
emily.pitler@yale.edu

Abstract

We present a simple, psychologically plausible algorithm to perform unsupervised learning of morphemes. The algorithm is most suited to Indo-European languages with a concatenative morphology, and in particular English. We will describe the two approaches that work together to detect morphemes: 1) finding words that appear as substrings of other words, and 2) detecting changes in transitional probabilities. This algorithm yields particularly good results given its simplicity and conciseness: evaluated on a set of 532 human-segmented English words, the 252-line program achieved an F-score of 80.92% (Precision: 82.84% Recall: 79.10%).

1 Introduction

This paper addresses the problem of segmenting natural language words into morphemes, the smallest units of language that still contain meaning. While one cannot extract meanings from lists of words and their frequencies, we can nevertheless use statistical information to make useful predictions about likely morphemes.

There is a large body of literature on morpheme induction, and while it is impossible to give a complete survey, see (Goldsmith, 2001) for a good summary of previous approaches. Goldsmith divides these past attempts into four categories: identification of morpheme boundaries using transitional probabilities; identification of morpheme-internal bigrams or trigrams; discovery of relationships between pairs of words; and an information-theoretic approach to minimize the number of letters in the morphemes of the language. Our work combines ideas from several of these approaches and does not fit neatly into any one of the categories.

The key idea in this paper is to use words that appear as substrings of other words and transitional probabilities together to detect morpheme boundaries. The first approach derives from the observation that the stem left over after removing prefixes and suffixes is often a legitimate word. Though, due to spelling changes, this is not always the case and therefore this method should not be used to actually segment a word. Given a large enough corpus, however, the most common morphemes can be found in this way. The other idea, using transitional probabilities, was initially presented by (Harris, 1955). Given an utterance, Harris proposed finding how many other utterances in the corpus shared each starting fragment of that utterance. He hypothesized that peaks in these counts correspond to morpheme boundaries.

(Hafer and Weiss, 1974) further developed the ideas presented in Harris's paper. Using Harris's transitional probability technique as a starting point, Hafer and Weiss created 15 different algorithms that achieved various levels of precision and recall. One issue with their approach is its heavy reliance on empirically determined parameters. For example, their best algorithm (with a precision of 91.0% and a recall of 61.0%) posited a morpheme boundary if the suffix is a word and the predecessor count is at least 5, or if the predecessor count is at least 17 and the successor count is at least 2.

Our goal was to design a simple algorithm based on our intuition that simpler algorithms are more likely to approximate human processes. We consciously limited both the number of language-specific assumptions that our program makes and "magic numbers"—parameters arbitrarily tuned to make the program work. We did not limit the length of morphemes, the number of morphemes per word, or the total number of morphemes.

2 Methodology

Our algorithm has four basic steps. We

1. build trees with probabilities based on the corpus,
2. score word fragments using these trees to obtain a large list of morphemes,
3. prune this list of morphemes, and
4. segment the test words using the morpheme list and the lexicographic trees.

Each of these steps is described in further detail below.

2.1 Building the Lexicographic Trees

At the beginning of the algorithm, we create two trees of letters and their associated counts: the "forward tree" and "backward tree". We explain here the construction of the "forward tree" (the other construction is symmetric). Suppose the alphabet of the language has b letters, and the longest word in the corpus consists of d letters. Then conceptually, we construct a complete b-way tree with depth d. At each node, each of the b branches represents one of the letters in the language. Thus, any path from the root to some node spells out the starting fragment of some word(s), and the node itself contains the frequency of that string. (Note that in practice, actually creating such a tree would be prohibitive as well as wasteful since most letter combinations never occur; thus we actually only store nodes with non-zero counts.)

The forward and backward trees allow us to calculate conditional probabilities in O(1) time given a starting or ending substring of a word. For example, we would use the forward tree to calculate $Pr_f(\text{s}|\text{report})$ (by dividing the frequency of words starting with "reports" by the frequency of words starting with "report"). In the opposite direction, we would use the backward tree to calculate $Pr_b(\text{e}|\text{ports})$ (by dividing the frequency of words ending in "eports" by the frequency of words ending in "ports").

2.2 Scoring Potential Morphemes

Once we have finished constructing the trees as described above, we begin finding morphemes. We maintain two lists of morphemes: a prefix list and a suffix list.[1] To populate the suffix list, for each word, we scan from the end of the word and consider every possible suffix in order of increasing length. Suppose we are considering the suffix $B\beta$ in the word $\alpha AB\beta$. We hypothesize the proposed suffix is correct if

1. αA is also a word in the corpus,
2. $Pr_f(A|\alpha) \approx 1$, and
3. $Pr_f(B|\alpha A) < 1$.

Similarly, the criteria for determining if αA is a prefix in the word $\alpha AB\beta$ is as follows

1. $B\beta$ is also a word in the corpus,
2. $Pr_b(B|\beta) \approx 1$, and
3. $Pr_b(A|B\beta) < 1$.

The first criterion corresponds to the observation that prefixes and suffixes are often added on to root words. For example, after removing the suffix "ed" from "corresponded", the resulting fragment "correspond" is still a word. The second and third criteria are checked using the forward and backward trees. They check that the stem has multiple children (thus implying other prefixes or suffixes can be joined to the stem) and that the stem's parent has only one child (thus identifying it as a true stem). Using the same example as before, the algorithm would check that $Pr_f(\text{d}|\text{correspon}) \approx 1$, and that $Pr_f(\text{e}|\text{correspond}) < 1$. If a given morpheme passes all three tests, we increase its score by 19 points; otherwise, we decrease its score by 1. After we have iterated through the entire corpus, we consider all strings with positive scores morphemes.

The rule of rewarding word fragments by 19 and punishing by 1 may seem arbitrary, but the constants were chosen so that a string has a positive final score only if it passes our tests at least five percent ($= \frac{1}{1+19}$) of the times it appears. Moreover, the numbers 19 and 1 are not special; any positive numbers x and y such that $\frac{y}{x+y} = .05$ would

[1] We use the terms prefix and suffix loosely, to denote any morpheme generally found at the beginning or end of words. For example, "man" is not technically a suffix, but it is a morpheme that often appears at the end of a word.

produce identical results.[2] The rewarding and punishing scheme is more effective than checking the percentage of tests passed because given two morphemes with the same percentage, the more common morpheme will have a higher score. Thus, the punishing/rewarding scheme takes into account both the reliability and the frequency of the string appearing as a morpheme. Single letters such as 't', which sometimes deceivingly appear to be prefixes, are punished far more often than they are rewarded. Strings such as 'psycho', which do not appear often but are almost always true morphemes when they do appear, are rewarded more often than they are punished. Suffixes like 's' are punished occasionally but rewarded very frequently, and are ranked at the top of the list.

2.3 Pruning

Clearly, this method is not perfect. In particular, one problem that often arises is that the final list of morphemes includes strings that are the concatenation of two other morphemes. For example, the list might include all of 'er', 's', and 'ers'. This is undesirable since the final step of segmenting words may process the word "throwers" as throw+ers instead of as throw+er+s. Fortunately, though, this problem has a relatively simple solution which we refer to as "pruning". We scan each list of morphemes, and if any morpheme is composed of two others with better scores, then it is thrown out.

2.4 Segmenting Words

Finally, we come to the actual segmenting of words. Given the list of morphemes, one possible approach is to simply peel morphemes off the ends of words as they are found. But words such as "politeness" pose a problem: should it be segmented as politenes+s or as polite+ness? Neither the scores nor the lengths of morphemes can be reliably used to answer this question. In this case, 's' would have a higher score, while 'ness' is a longer morpheme. They key observation is that the same probability criteria that was used earlier to detect morphemes can be applied here to measure the appropriateness of segmenting at a particular position. In this example, we expect $Pr_f(\text{n}|\text{polite})$ to be lower than $Pr_f(\text{s}|\text{politenes})$ which leads to the correct segmentation.

[2]Suppose that we rewarded and punished by $x > 0$ and $y > 0$ respectively, satisfying $y/(x + y) = 0.05$. Then $y = 0.05\,(x + y) \Rightarrow 0.05x = 0.95y \Rightarrow x = 19y$. Thus, if a string is rewarded r times and punished p times, it would have a score of $xr - yp = 19yr - yp = y(19r - p)$, which is exactly y times our score. In particular, a string has a positive score if and only if it had a positive score in our algorithm.

Table 1: Evaluation results of *RePortS*

Language	Precision	Recall	F-Score
English	82.84 %	79.10 %	**80.92 %**

Thus, our method for segmenting is as follows. First, we scan each word from the end, and find all morphemes $B\beta$ from the suffix list such that our word can be written as $\alpha B\beta$ (for some α). The morpheme with the lowest value of $Pr_f(B|\alpha)$ that is also smaller than 1 is chosen. If such a morpheme is found, it is removed and the processed is repeated until no more morphemes can be removed. We then repeat the same process, attempting to peel off morphemes in the prefix list from the beginning of the word (using Pr_b instead of Pr_f).

3 Results

The algorithm described above was implemented as a Perl program called *RePortS*[3]. The English frequency-word list provided by the Neural Networks Research Centre at the Helsinki University of Technology was combined with a year's worth of articles from the *Wall Street Journal* and a Linux dictionary file to obtain a corpus containing 185,696 words for training. To determine the performance of the algorithm, we ran our program on a "gold standard" of 532 words (again, provided by the Neural Networks Research Centre) and evaluated our proposed segmentation against the human-determined standard (see Table 1).

Our program identified a total of 1795 morphemes (808 in the prefix list and 987 in the suffix list). Table 2 contains the ten highest-scoring morphemes from each list.

The program was tested on a dual 2.8 GHz processor with 2 GB of memory. We monitored the total running time, i.e. training and segmentation time, and the maximum memory usage of *RePortS*. They are reported in Table 3 for test data of different sizes (note that the same training corpus was used in both cases).

While our algorithm was designed with English and other Indo-European languages in mind, we

[3]The earliest versions of the algorithm determined that the most common prefix, stem and suffix are 're', 'port' and 's', respectively; hence, the name *RePortS*.

Table 2: Top English morphemes

Morpheme	Score
un	15858
re	5312
dis	3783
non	2998
over	2717
mis	1812
in	1689
sub	1632
pre	1418
inter	1189

Morpheme	Score
s	24351
ly	18847
ness	10430
ing	8740
ed	5669
al	2655
ism	2169
less	1940
ist	1669
able	1613

Table 3: Resource usage for different test data

# Words	Time	Space
532	0m 27sec	$\sim$ 139 MB
167,377	34m 37sec	$\sim$ 139 MB

decided to test our program with other languages as well. We used test data for Turkish and Finnish provided by the Neural Networks Research Centre. Our results for these two languages are given in Table 4.

4 Discussion

As mentioned earlier, our algorithm performs well given its conciseness and simplicity: the Perl implementation was a total of 252 lines including comments and the algorithm itself can be fully described in four basic steps. Examples of words that our program segments correctly include “repayments” and “passionflowers”. The former contains several affixes and *RePortS* correctly suggests re+pay+ment+s as the segmentation. The latter, on the other hand, is an uncommon compound word segmented as passion+flower+s.

However, the algorithm is obviously not flawless. Consider a word such as “widen” (with a correct segmentation of wid+en). The letters ‘en’ often appear at the end of a word but not as a suffix (e.g. even, ten, hen, etc.) Thus, the potential morpheme ‘en’ is punished far more frequently than it is rewarded, and does not appear in the final list of morphemes. This omission causes us to incorrectly segment words such as “widen” (which our program leaves untouched). However, on the whole, our program performs better with the exclusion of ‘en’.

Table 4: Evaluation results of *RePortS*

Language	Precision	Recall	F-Score
Turkish	72.68 %	43.01 %	**54.04 %**
Finnish	83.76 %	32.30 %	**46.62 %**

Furthermore, there is some evidence that our algorithm is psychologically plausible. As shown in (Saffran et al., 1996a) and (Saffran et al., 1996b), adults as well as infants are able to identify words from continuous speech where the only available cues are transitional probabilities between phonemes. These results show that it is possible for humans to keep track of transitional probabilities and use it in segmentation tasks. However, as (Yang, 2004) points out, transitional probabilities by themselves are not sufficient for larger corpora, and indeed, our algorithm depends on other information as well.

5 Future Work

One notable feature of *RePortS* is that it uses only a list of words and their frequencies. Clearly, contextual information is lost when English text is collapsed into such a list. We feel that the performance could only be improved by extending our algorithm to take advantage of such information. Along the same lines, the inclusion of phonological information may also improve the performane of our algorithm. (Saffran et al., 1996b) showed that humans learned better when presented with both transitional probabilities and prosidic cues than with transitional probabilities alone. Thus, this too is an avenue for improvement.

On a slightly different note, another possibility for further research would be to modify our program to even more closely mirror human learning. More specifically, humans generally do not perform “batch learning”. Therefore, instead of feeding hundreds of thousands of words to the program at once, we could supply the words in smaller chunks, and in the order in which infants would likely hear them. It would be interesting to compare our current results to those from this process.

6 Conclusion

We described an efficient algorithm that uses statistical relationships within and between words to predict morpheme boundaries. Humans are also sensitive to such patterns in natural language.

Moreover, our heuristics make intuitive sense. While we do not claim that humans use our algorithm to segment words, we believe that further research along this line has potential to reveal insight into human language processing.

The program *RePortS* performs quite well against a human-segmented gold standard for English; its precision and recall were both approximately 80%, with an F-Score of 80.92%. Moreover, even though the algorithm was designed for Indo-European languages with a concatenative morphology, it achieved surprisingly decent results for Finnish and Turkish. We experimented with other variants that achieved higher F-Scores, but the algorithm presented here achieved the best balance between performance and elegance.

7 Acknowledgements

We thank Charles Yang for his support throughout the project. It was only through his encouragement and support that we expanded our class project into this work. We would also like to thank Bob Berwick for pointing out the relevance of (Hafer and Weiss, 1974) to our work.

References

John Goldsmith. 2001. Unsupervised Learning of the Morphology of a Natural Language. *Computational Linguistics*, 27(2):153–198.

Margaret A. Hafer and Stephen F. Weiss. 1974. Word Segmentation by Letter Successor Varities. *Information Storage and Retrieval*, 10:371–385.

Z. Harris. 1955. From Phoneme to Morpheme. *Language*, 31(2):190–222.

Jenny R. Saffran, Richard N. Aslin, and Elissa L. Newport. 1996a. Stastical Learning by 8-Month-Old Infants. *Science*, 274(5294):1926–1928.

Jenny R. Saffran, Elissa L. Newport, and Richard N. Aslin. 1996b. Word Segmentation: The Role of Distributional Cues. *Journal of Memory and Language*, 35(4):606–621.

Charles D. Yang. 2004. Universal Grammar, statistics or both? *TRENDS in Cognitive Sciences*, 8(10):451–456.