Swordfish: Using Ngrams in an Unsupervised Approach to Morphological Analysis

Chris Jordan[*]
Dalhousie University
6050 University Ave.
Hfx., NS, Canada B3H 1W5
cjordan@cs.dal.ca

John Healy
Dalhousie University
6050 University Ave.
Hfx., NS, Canada B3H 1W5
healy@cs.dal.ca

Vlado Keselj[†]
Dalhousie University
6050 University Ave.
Hfx., NS, Canada B3H 1W5
vlado@cs.dal.ca

Abstract

Morphological analysis refers to the art of separating a word into its base units of meaning, or morphemes. Many popular approaches to this, including Porter's algorithm, have been rule-based. These rule-based algorithms however, generally only perform stemming, the identification of root morphemes, which is only a part of morphological analysis. Such algorithms can only reasonably be applied to languages with a limited number of possible affixes for a given term. Rule based algorithms require a great deal more complexity in order to handle languages with many affixes reliably. We propose Swordfish, an ngram-based unsupervised approach to morphological analysis, as an alternative. An ngram is simply a substring of length n which occurs within a corpus. We take those ngrams with the highest probabilities of occurring within our corpus to be our candidate morphemes. We apply a recursive algorithm, which repeatedly splits a term using a probabilistic-based criterion. The evaluation on the PASCAL dataset shows somewhat better performance on English and worse on Finnish and Turkish word lists than the state-of-the-art system Morfessor, with a significantly lower cost in running time.

1 Introduction

Words in a language are typically a combination of smaller base units of meaning, referred to as morphemes. The act of separating a word into its morphemes is called morphological analysis. For example, the code block below illustrates the morphemes in the word "pretested":

```
pre + test + ed
```

Morphological analysis is an important task, as it allows for the identification of the base units of meaning in a word. Morphemes can be used to identify terms which are semantically similar. This is a common process in document retrieval and has been shown to improve performance of these systems (Kantrowitz et al., 2000). Morphemes are also useful in speech recognition (Siivola et al., 2003) since in many languages the spelling and pronunciation of a word are directly related.

Finding morphemes, unfortunately, is not always simple to do particularly in compounding languages such as Finnish, German, Swedish or Greek; and in highly inflective languages such as Finnish and Hungarian (Hirsimäki et al., 2005). Rule-based stemmers such as Porter's Algorithm (Baeza-Yates and Ribeiro-Neto, 1999) have had some success in identifying and removing affixes in the English language. However, English is not a particularly complex language, at least in terms of affixes. There are typically only one or two prefixes or suffixes possible for most words. A rule-based approach for compounding or highly inflective languages is not particularly effective, due to the sheer number of possible word forms. Similarly, to develop training data for a supervised approach would require an inordinate effort to gain an appropriate level of coverage.

An unsupervised approach to morphological analysis is attractive for highly inflective languages and languages with extensive use of com-

[*] http://www.chrisjordan.ca
[†] http://www.cs.dal.ca/∼ vlado/

pounding. Additionally, an unsupervised approach by definition requires no or minimal linguistic knowledge, which makes it convenient for less common languages and languages with sparse linguistic resources. An ideal unsupervised approach, by definition, should require no training data and very little user input in order to learn the morphemes for a given language. Furthermore, an ideal unsupervised approach should be language-independent and, as such, should be able to extract morphemes from any language, given enough exposure to it. Swordfish is such an approach. The Swordfish algorithm processes a corpus that lists terms and their respective frequencies. A language model is built, using ngram frequencies, and candidate morphemes are identified within words.

2 Previous Work

Morphological analysis and stemming are two similar text processing tasks. Morphological analysis identifies all the morphemes in a word, while stemming finds the root morpheme or stem for a term. Since stems are morphemes themselves, stemming is a subset of morphological analysis. Porter's algorithm (Baeza-Yates and Ribeiro-Neto, 1999) is a popular rule-based approach to stemming words in the English language. There are other rule-based approaches (Kantrowitz et al., 2000; Frakes and Fox, 2003) each differing in the degree to which they will stem a term. The stronger the stemmer, the more a word is altered and the smaller the document index. Retrieval recall also tends to increase with the strength of the stemmer, while precision decreases. While rule-based approaches do encounter some success in the English language, they are less successful when applied to compounding or highly inflective languages (Hirsimäki et al., 2005).

There has been a limited amount of work done in the area of unsupervised morphological analysis. One proposed approach is Morfessor (Hirsimäki et al., 2005; Creutz and Lagus, 2005), which recursively builds a morpheme lexicon. It begins by initializing the set of candidate morphemes, the morpheme lexicon, as the entire vocabulary. Using the frequencies which these morphemes occur in the corpus, Morfessor passes through the entire vocabulary, splitting words into the most likely morphemes. Each word is recursively split into the two most likely substrings until such a split is less probable than the substring being split. The morpheme lexicon is updated with the resulting splits. Passes through the vocabulary are made until the lexicon can no longer be improved. Essentially, the Morfessor algorithm looks for the most likely morphemes by splitting words in the most likely manner. The candidate morpheme lexicon which is produced tends to have good precision though it suffers from low recall. Recall and precision in morphological analysis are standard measures expressed in terms of correctly and incorrectly identified "breakpoints" within a word. These measures will be explained in more details in later sections.

A significant component to the Swordfish algorithm is a suffix array which is used to extract the ngrams. The Yamamoto–Church algorithm (Yamamoto and Church, 2001), modified slightly to handle a list of term frequencies instead of regular text, was employed here due to its ease of implementation and $O(N \log(N))$ run time. In the proposed Swordfish algorithm presented here, ngrams used during the morphological analysis must also be a longest common prefix (LCP) with a length greater than or equal to 1. Hence only those ngrams that occur in multiple terms will be considered as candidate morphemes.

3 Swordfish

The Swordfish algorithm consists of two main phases. The first phase computes the ngram frequencies for our corpus via a modified Yamamoto-Church algorithm (Yamamoto and Church, 2001) that deals with word lists instead of regular text. We include all n-grams, i.e., word substrings, of lengths ranging from 1 to the maximal word length. This phase of our algorithm takes up the majority of run time and memory usage. The resulting set of ngrams are treated as a lexicon of possible morphemes. With this lexicon, it is possible to calculate a probability model for the ngrams using maximum likelihood estimates (MLE) as shown in Equation 1.

$$P(ngram_i) = \frac{freq(ngram_i)}{\sum freq(ngram_n)} \qquad (1)$$

where $P(ngram_i)$ is the probability of $ngram_i$ in the corpus probability model. $freq(ngram_i)$ is the frequency which $ngram_i$ occurs in the corpus and $\sum freq(ngram_n)$ is the total number of ngram occurrences in the corpus.

The second phase uses our ngram frequencies to determine probable splits for dividing words into

their base morphemes. There are several plausible methods for dividing a word into its base morphemes. We make the assumption that, since words are built up from morphemes, we expect to see the ngrams representing these morphemes more frequently than we would a random string of characters of equal length.

The steps Swordfish takes to split terms are as follows:

Step 1: Calculate the probability of the current term occurring in the ngram lexicon. This probability is calculated using the MLE. If the term does not occur in the lexicon then it has a probability of 0 or only appears as a substring of a unique larger ngram.

Step 2: Find the two ngrams with the highest probability of forming the term. The probability of two ngrams forming a particular term is considered to be the product of their probabilities in the lexicon. In other words, given a term t we identify the two subterms x' and y' as:

$$(x', y') = \arg\max_{xy=t} P(x)P(y)$$

Step 3: If the probability of the current term is less than the product of the two ngrams in Step 2 then the term is split into its two constituent ngrams. These ngrams are then considered to be terms themselves and Steps 1–3 are repeated.

Step 4: The final set of ngrams that result from Step 3 are considered to be the morphemes for the original term.

The Swordfish algorithm essentially considers all ngrams to be possible morphemes. For a given term, it recursively splits it into two substrings based on the most likely combination of ngrams. The set of ngrams resulting from this splitting process are the suggested morphemes for the original term. The Swordfish algorithm has two strong advantages. First, it is parameter-free and thus requires no tuning on the part of a user; second, it is a purely unsupervised approach, requiring no training data to accomplish its task.

4 Evaluation

The development of Swordfish was prompted by PASCAL's 2005 challenge to facilitate the unsupervised segmentation of words in to morphemes (Morpho Challenge, 2005). For this reason, we use both their corpus and their methodologies to evaluate our algorithm. They use three corpora: one in English; one in Finish; and one in Turkish. The corpora were presented in term frequency lists derived from real world corpora. Currently, Swordfish requires this format in order to run, so any text document would have to be preprocessed into a term frequency list beforehand.

The algorithms for this challenge are evaluated by sampling a gold standard data set that contains a subset of the terms split into their appropriate morphemes. These splits are referred to as surface-level segmentations, or segmentations that contain exactly the same characters as the initial term. Thus where a more traditional morphological analysis might separate 'unsupervised' into 'un+supervise+ed' our ideal separation will be 'un+supervis+ed'.

This evaluation is not weighted by frequency of terms. For example, an error on the term 'the' would be equivalent to an error on the term 'agglutinative'. The metrics used to evaluate performance are precision, recall, and F-measure. In order to compute these values one runs an algorithm across a given term frequency list and divides the terms into morphemes. A random subset of these terms have been tagged and provided to participants as an evaluation set. Of these, a random sample has been selected to evaluate the performance of the various algorithms.

5 Results

Table 1 compares the results for precision, recall and F-measure scores from a baseline run, which shows the results of placing a split between every character, Morfessor, and the Swordfish algorithm. From initial inspection based on the F-measure, we can see that Swordfish outperforms Morfessor on English while Morfessor produces more accurate results on Finnish. The baseline outperforms both approaches on Turkish.

At its heart, this problem can be thought of as a classification problem, with the division between each character in every term being classified as either a morpheme boundary or not a morpheme boundary. This reduces our precision/recall problem to the traditional problem of false positives vs false negatives. A false positive refers to a morpheme boundary being predicted at a location where no morpheme boundary exists while a false negative refers to a morpheme boundary unidentified as such. From this point of view, it is evident that Morfessor has accepted a large number

Language	Algorithm	Precision	Recall	F-measure
English	Baseline	14.56	100.00	25.42
	Morfessor	74.19	26.64	39.20
	Swordfish	55.11	37.45	44.60
Finnish	Baseline	19.71	100.00	32.92
	Morfessor	83.66	29.51	43.63
	Swordfish	70.39	23.58	35.33
Turkish	Baseline	25.87	100.00	41.11
	Morfessor	76.33	24.17	36.71
	Swordfish	58.90	16.79	26.13

Table 1: Comparison of algorithms: precision, recall, F-measure

of false negatives, thus its low recall, in order to minimize its false positives, and thus keep a high level of precision.

Swordfish has similar performance to Morfessor. Both algorithms have a precision that is much higher than its recall, over all languages. Comparing Swordfish with Morfessor in the analysis of English, Swordfish has higher recall and lower precision. In the analysis of Finnish and Turkish, Morfessor has a higher precision and recall. From these preliminary results, it appears that morphemes in English can be more easily extracted using ngrams, while the recursive approach to building a morpheme lexicon used in Morfessor is more effective for Finnish. Neither approach appear to be effective for Turkish though the high precision scores do indicate promise for further research on them.

Two factors not taken into consideration in this evaluation are running times and memory usage of the algorithms. In our initial experiments Morfessor seems to be an order of magnitude slower than Swordfish. Data sets that took a few hours with Swordfish took closer to a day to run with Morfessor.

On the other hand, Swordfish seems to use approximately an order of magnitude more memory than Morfessor. In our experiments we've seen Morfessor use 300MB of memory to process a 20MB file while Swordfish took 3GB to process the same file. Swordfish is currently implemented in Perl and uses some very inefficient hashes to store its suffix arrays and LCP tables.

It should also be noted that these numbers are just observations and more formal benchmarking will be required before any concrete comparisons of memory usage and running time can be made between the two algorithms.

6 Conclusions

In this work, we present Swordfish, a recursive approach to morphological analysis using ngrams. It is purely unsupervised and requires no parameter tuning or supervised training. It constructs an ngram lexicon from which all candidate morphemes are drawn. Morphemes are extracted from terms based on the most probable combination of ngrams.

A major component of the Swordfish algorithm is a Perl module implementing the Yamamoto–Church algorithm used to calculate our ngram frequencies. Currently this algorithm is particularly memory intensive due to heavy use of Perl hashes. There is current development to make this module more memory efficient, which should result in a dramatic decrease in the memory necessary to run the Swordfish algorithm.

Regardless, the Swordfish algorithm is a time-efficient algorithm and results in moderately high precision. Unfortunately, it suffers from low recall. We believe that using a ngram lexicon as the foundation for performing morphological analysis shows a lot potential. Further research should go into how the probabilities of ngrams are compared to the probabilities of terms. The method used here, where the probability of the term is compared to the product of the ngram probabilities is rather simplistic. A better method for comparison may lead to improved recall.

As well, further investigation should be conducted into how the ngram lexicon is constructed. Obviously, not all ngrams are morphemes for a language. In our current approach though, all ngrams are considered possible morphemes. The lexicon might be improved by filtering out ngrams that are determined to not be morphemes. This process can be considered to be "removing noise"

from the lexicon and may lead to greater precision.

Both algorithms here were evaluated against randomly selected terms. Unfortunately, there was no evidence to suggest conclusively that our results were not simply an aberration. In order to properly evaluate our algorithm we would like to repeatedly bootstrap our gold standard in order to generate test sets. These test sets could be used to create confidence intervals for our precision, recall and F-measure.

In the results reported here, Swordfish outperforms Morfessor on English but Morfessor is better on Finnish and Turkish. The use of an ngram lexicon, as implemented in Swordfish, has potential as an unsupervised approach to morphological analysis. There is still a great deal of work that could be done to improve the algorithm, especially with regards to the Turkish language. It remains to be seen whether a language-independent approach to morpheme extraction will succeed, or whether the problem will require differing approaches for different families of languages.

7 Acknowledgments

This research was funded by the Natural Sciences and Engineering Research Council of Canada, NSERC and Sun Microsystems.

References

Mark Kantrowitz, Behrang Mohit, and Vibhu Mittal. 2000. Stemming and its effects on TFIDF ranking (poster session). *SIGIR '00: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, 357–359.

Vesa Siivola, Teemu Hirsimäki, Mathias Creutz, and Mikko Kurimo. 2003. Unlimited Vocabulary Speech Recognition Based on Morphs Discovered in an Unsupervised Manner. *Proc. Eurospeech'03*, 2293-2296.

Teemu Hirsimäki, Mathias Creutz, Vesa Siivola, Mikko Kurimo, Sami Virpioja, and Janne Pylkkönen. 2005. Unlimited vocabulary speech recognition with morph language models applied to Finnish. *Computer, Speech and Language*.

R. Baeza-Yates and B. Ribeiro-Neto. 1999. *Modern Information Retrieval*. Addison-Wesley.

William B. Frakes and Christopher J. Fox. 2003. Strength and similarity of affix removal stemming algorithms. *SIGIR Forum*, 37(1): 26–30.

Mathias Creutz and Krista Lagus. 2005. Unsupervised Morpheme Segmentation and Morphology Induction from Text Corpora Using Morfessor 1.0. *Technical Report: A81*. Helsinki University of Technology.

Mikio Yamamoto and Kenneth W. Church. 2001. Using suffix arrays to compute term frequency and document frequency for all substrings in a corpus. *Comput. Linguist.*, 27(1): 1–30.

PASCAL Unsupervised Segmentation of Words into Morphemes Challenge 2005. 2006. WWW: http://www.cis.hut.fi/morphochallenge2005/.