Journal of Interesting Negative Results in NLP and ML, vol. 1 (2008) 1-18 Submitted 10/2007; published 4/2008

Swordfish2: Using Kernel Density Estimation to Smooth N-gram Histograms for Morphological Analysis

Chris Jordan Jane E. Mason John Healy Vlado Keselj
Carolyn Watters

Faculty of Computer Science, Dalhousie University
6050 University Avenue, Halifax, Nova Scotia, Canada, B3H 1W5
{cjordan, jmason, healy, vlado, watters}@cs.dal.ca

Abstract

Swordfish2 is an extension of the original Swordfish algorithm, an unsupervised approach to morphological analysis using character N-gram probabilistic models. In the original algorithm, a generative character N-gram model is created from a word list composed of word frequency pairs. Morphological word splits were discovered by comparing a given term's probability to the probabilities of its corresponding N-grams. Swordfish2 builds multiple generative N-gram models instead of just a single one. Each model represents a different location in a word, thus incorporating the word location of N-grams into the morphological analysis. Histograms for each N-gram are constructed from these generative models and smoothed using kernel density estimation. The original Swordfish algorithm provided evidence that indicated that morphemes are highly probable N-grams. Swordfish2 attempted to extend the original Swordfish approach to include word location, but was unable significantly to improve the morphological analysis on the 2005 Morpho Challenge and CELEX data sets. While these results do provide evidence that support our hypothesis that the most probable N-grams are morphemes, the design of the probabilistic model that proves it remains an open question.

1. Introduction

What is a morpheme?

Morphology is an area of linguistics that involves the study of word structure, both within one language and across languages. The study is based on the idea that words are related to one another by a variety of rules. A morpheme is the smallest unit of meaning in a language. Analyzing words as sequences of morphemes is one approach to describing relationships between words. In the English language, for example, the word `rats` is related to the word `rat` in the same way as the word `cats` is related to the word `cat`. In each case, the former word consists of two morphemes (`rat` + `s` and `cat` + `s`), whereas the latter word consists of only one morpheme. Note that morphemes are not necessarily distinct syllables in a word. For example, the word `unhappiest` contains four syllables, but only three morphemes `un` + `happi` + `est`.

Why is morphological analysis important?

Separating a word into morphemes is a non-trivial operation, but it is extremely useful in areas such as information retrieval (IR), including the classification, clustering or retrieval of documents. In IR, the separation of a word into morphemes can be used to reduce the dimensionality of the vector space by identifying terms that are semantically similar. For example, the words `happier`, `happiest`, `unhappier`, `unhappiest`, `happily`, `unhappily`, `happiness`, and `unhappiness` would all be mapped to the root morpheme `happi`.

What is the difference between stemming and morphological analysis?

Traditionally, this form of morphological reduction of a word into a simpler form, called a *stem*, is known as *stemming*. Stemming is a subset of morphological analysis that has been shown to improve the performance of IR systems (Frakes and Baeza-Yates, 1992; Kantrowitz et al., 2000). Stemming is typically a suffix-based transformation. In English, prefix removal is not used since it would typically make a radical change in the meaning of a word (Porter, 2001). A word *stem* should be distinguished from its *root*. A root is an "inner" word from which the initial word is derived, that is to say, it has an etymological meaning, while stemming is driven by its application in IR.

Different languages have different morphologies

Languages may be divided into three broad categories: *isolating*, *agglutinative*, and *inflective* languages. Isolating languages, such as Chinese, have little or no morphology and thus do not benefit from morphological analysis. Agglutinative languages, also known as agglomerative or compounding languages, are those in which basic roots and words can be combined to make new words. These languages, such as Turkish or Finnish, tend to have many morphemes. Inflectional morphemes are used to modify a word to reflect information such as tense. Inflective languages, also known as fusion languages, such as German, Greek, or Latin, are those in which inflectional morphemes can be joined or *fused* with other morphemes. Note that a language may belong to more than one of the categories discussed. Finnish, for example, is both an agglutinative and an inflective language.

Issues with rule based approaches

Traditional approaches to reducing words to their root morpheme, stemming techniques, have typically involved rule-based algorithms, such as Porter's algorithm for the English language (Porter, 1980; Baeza-Yates and Ribeiro-Neto, 1999). These approaches are language-dependent and require a substantial manual effort of expert linguists. Additionally, even if one succeeds in building wide-coverage rules for a particular language, the issues of generalizability and handling of unknown words may still be present, whereas they are inherently addressed from the start in an unsupervised approach. The difficulty in building rule-based morphological analyzers is evident from the fact that stemmers are available for only a few world languages (Porter, 2001).

Swordfish: an unsupervised approach

The Swordfish algorithm (Jordan et al., 2006a; Jordan et al., 2006b) is an alternative to traditional rule-based stemmers, which uses probabilistic models based on character N-gram frequencies. A deficiency in this algorithm is that these probabilistic models do not consider the location of N-grams within the words being analyzed. Swordfish2 attempts to address this issue by building multiple probabilistic models to represent different word locations. Histograms for each N-gram are built from this set of models to which kernel density smoothing is then applied.

Paper overview

The purpose of this paper is to present Swordfish2, an attempt to expand the basic generative model used in the original Swordfish algorithm. We note that although Swordfish2 is language-independent, like Swordfish, there is no expectation that the algorithm will work equally well with all languages. In Section 5, we present a comparative analysis between Swordfish and Swordfish2 over the 2005 PASCAL Challenge Workshop on unsupervised segmentation of words into morphemes (2005 Morpho Challenge) (MorphoChallenge, 2006) and CELEX (Baayen et al., 1996) data sets. These data sets contain wordlists for English, Finnish, Turkish, and German and corresponding morphological analyses.

2. Previous Work

Traditional approaches to either morphological analysis or stemming involve rule-based algorithms such as Porter's stemming algorithm for English (Porter, 1980; Baeza-Yates and Ribeiro-Neto, 1999). Porter's algorithm removes about 60 different suffixes using a multi-step method, whereas the earlier Lovins stemmer (Lovins, 1968) employs a longest-match algorithm that can remove approximately 260 suffixes. Another English language stemmer, KSTEM (Krovetz, 1993), uses both a dictionary of terms and morphological rules. In this case, suffixes are removed using morphological rules until the remaining term has been found in the dictionary, at which point the suffix removal for this term ends. There also are stemmers for languages other than English, for example Popovic and Willett's Slovene stemmer (Popovic and Willett, 1992), but such rule-based approaches may be less successful for agglutinative or highly inflective languages (Hirsimäki et al., 2006). Various programming languages, such as Snowball (Porter, 2001), are available to assist in the development of stemming algorithms.

One supervised approach to morphological analysis is offered by Wicentowski (Wicentowski, 2004). This model, known as WordFrame, uses training data to learn morphological rules, and can also use lists of pre-specified prefixes, suffixes, and potential roots to enhance performance. One drawback of supervised methods such as WordFrame, however, is the large amount of training data that is needed, particularly for complex languages.

Arabic is an extremely complex inflectional and agglutinative language. These characteristics make it a very challenging language to analyze. A supervised approach to morphological analysis of the Arabic language is presented by Marsi, van den Bosch, and Soudi (Marsi et al., 2005). Marsi *et al.* train their system on data from the Arabic Treebank, using a rule-based analysis which includes producing all segmentations of a term and then using table and dictionary lookups. They note that such an approach is feasible, with the limitation that it is unable to recognize correctly stems of unknown words with ambiguous root forms.

The modern Hebrew language has a complexity similar to Arabic, with a high level of ambiguity and an affixational morphology. Morphemes in a Hebrew word can be combined both agglutinatively and fusionally. Adler and Elhadad (Adler and Elhadad, 2006) discuss an unsupervised morpheme-based hidden Markov model for the disambiguation of this language. Although their focus is on modern Hebrew, their approach is applicable to other languages with an affix morphology. Their model performs unsupervised segmentation and disambiguation in parallel. They note that an advantage of unsupervised methods is that they can adapt to the dynamic nature of languages, such as modern Hebrew, which change over time.

There are other unsupervised approaches to morphological analysis, including the well-known Lingustica and Morfessor models. Linguistica (Goldsmith, 2001) uses an unsupervised signature-based analysis to create general templates for determining morphemes. Morfessor (Creutz and Lagus, 2005), on the other hand, uses statistical analysis to determine the most likely morpheme splits in a word. Initially, all words in the given vocabulary are considered to be morphemes. Using frequency analysis, words are then split into the most likely combination of morphemes. Splitting continues until there has been no improvement in the resulting morpheme lexicon.

At the 2005 Morpho Challenge, Keshava and Pitler (Keshava and Pitler, 2006) presented the best-performing unsupervised morphological analysis algorithm for English. In their approach, they built two probabilistic trees based on the corpus. One tree gave the likelihoods for different characters' occurrence after a given substring. The other tree gave the likelihoods for characters occurring before it. These two trees allowed the Authors to detect quite accurately suffixes and prefixes in the English language. Bernhard (Bernhard, 2006) also presented an approach that first extracts prefixes, suffixes and stems from a lexicon by looking at the probabilities of N-grams which occur next to each other. Word frequencies, however, are not used in the estimation of these N-gram probabilities; each word in the lexicon is only considered once. Word segmentation is performed by comparing words with the same stem and examining these lists of possible affixes.

Swordfish (Jordan et al., 2006a; Jordan et al., 2006b) is another unsupervised approach to morphological analysis that was presented at the 2005 Morpho Challenge. Its performance, however, placed it in the middle of the pack, well back of the top submissions. It is a statistical approach that uses character N-grams to determine the most likely morpheme splits. The underlying hypothesis of the Swordfish algorithm is that because morphemes are N-grams, given the appropriate probabilistic model, the N-grams with the greatest probability mass are also morphemes. The challenge in proving this hypothesis lies in creating such a probabilistic model. The original Swordfish algorithm creates a basic generative model of all possible character N-grams from the vocabulary of a given corpus. Note that in the Swordfish algorithm, N-grams do not span the text, but are limited to a single word; thus, for each word, all possible substrings of the word are generated as N-grams. Once the N-grams have been generated, and the frequency of each one has been determined, a generative model is constructed and used to identify the most likely splits in each word. Swordfish recursively splits each word based on the N-gram probabilities.

The use of character N-grams has been explored previously by Schuegraf *et al.* (Schuegraf and Heaps, 1976) as a means for reducing the size of indices for document retrieval without significantly reducing performance in terms of accuracy. In Schuegraf's approach, N-grams were selected based on the number of documents in which they occurred. The Authors were interested in N-grams that occurred in approximately the same number of documents for reducing the sizes of indices.

Our work counts the number of occurrences of N-grams to build a generative model. A modified version of the Yamamoto-Church algorithm (Yamamoto and Church, 2001) is used to detect N-grams and their frequencies. In their original approach, a suffix array is built from a piece of text. Next, the longest common prefixes (LCP) are calculated for each suffix. The LCP is the longest prefix that a suffix shares with its alphabetic neighbour. Using the suffixes and their LCP, character N-grams are then extracted. The generative N-gram model that is initially built here is created from these raw frequencies using the maximum likelihood estimate (MLE). The MLE approach is employed because it is a straightforward approach to estimating probabilistic models. It creates models purely from raw frequencies and does not have any parameters that need to be tuned.

Probabilistic models that are built using the maximum likelihood estimate often contain zero probabilities and require some smoothing – the redistribution of probability mass from one event to another. Smoothing has been explored extensively for document retrieval (Zhai and Lafferty, 2001). Retrieval models often have an assumption of exchangeability, meaning that these models assume that every permutation of term occurrences is equally likely. This assumption obviously does not hold in the real world, but the models that use it are still effective for retrieval. Common smoothing techniques for retrieval do not directly apply to the lexical models here.

Swordfish2 incorporates N-gram location within words by building multiple generative models to represent those locations. A histogram is constructed for each N-gram using these models. Some of these generative models, however, will have zero probabilities for some N-grams. In order to remove these zero probabilities from the resulting histograms, an alternative technique has to be employed. As the relative locations of morphemes vary from word to word, kernel density estimation (Hastie et al., 2001) was used to smooth our histograms to reflect the uncertainty of these locations. Kernel density estimation is a popular technique in statistics for interpolating a smooth density function for data.

3. Swordfish2 Approach

Swordfish2 is an adaptation of the original Swordfish algorithm. Swordfish is an unsupervised statistical algorithm that can learn and detect morphemes in text with over 50% precision, if given enough exposure to a language. The shortcoming of Swordfish is that it suffers from low recall. In an attempt to address this issue, Swordfish2 creates histograms of N-gram probabilities based on location, which allows position-based conditional probabilities to be derived.

3.1 Probabilistic Model

As with Swordfish, Swordfish2 is based on the hypothesis that because morphemes are N-grams, the most probable N-grams, given the appropriate probabilistic model, are morphemes.

The model used in Swordfish2 is more complex than the basic generative model used in the original Swordfish approach. The first step in Swordfish2 is to generate all possible N-grams from each word in the vocabulary of the corpus and determine the frequency of each of these N-grams. These frequencies are used to construct multiple generative models that represent different word locations. To facilitate this process, the corpus is first broken down into a list of the vocabulary terms and their respective frequencies within the corpus. All the N-grams for each term are then generated using a modified version of the Yamamoto-Church algorithm (Yamamoto and Church, 2001), which uses suffix arrays to generate the N-grams. The original Yamamoto-Church algorithm deals with the full text of a corpus, and therefore is modified in this case to use the constructed list of vocabulary terms and their frequencies. As with the original Swordfish algorithm, these N-grams are substrings of each individual term, and do not span multiple terms.

In the original Swordfish algorithm, a single generative model was created using the maximum likelihood estimate from the N-grams that were extracted from the corpus. This model, however, does not account for the word location in which the N-grams occur. Swordfish2 extends the Swordfish approach to incorporate word location by creating a set of generative models in which each model represents a different word location. For example, Swordfish2 can create a set of three generative models in which the models represent N-grams occurring at the beginning, the middle, and the end of words respectively. The original Swordfish algorithm is in fact an instance of Swordfish2 in which only a single generative model is created, one that represents N-grams occurring over the entire word length.

Using a set of generative models as described previously, histograms can be constructed for each N-gram in which each block represents a different word location. Consequently, these histograms give graphical representations to the probabilities that N-grams have for occurring at different spots in a word. Figure 1, for example, illustrates the histogram for the bigram `ed` in the English language, created from a set of ten generative models, each representing a different word location. These models were constructed using the 2005 Morpho Challenge (MorphoChallenge, 2006) and CELEX(Baayen et al., 1996) English wordlists. It is quite clear from this figure that the bigram `ed` is more likely to occur at the end of words in English than at any other location.

To create a set of generative N-grams models representing different word locations, the word length must first be normalized. This results in N-grams having to occur within a word length interval of $[0, 1]$. The interval is then partitioned into blocks of equal size. Next, the generative models are built by simply tallying the number of N-gram occurrences in each block. Thus, the occurrences for an N-gram are separated into different blocks based on where they start in the vocabulary terms. The number of generative models that are created is equal to the number of partitions in the interval.

The N-gram histograms are created from this set of generative models. Thus, they are sensitive to the number of generative models that the set contains. Varying the number of blocks used to partition the word length interval gives the resulting N-gram histograms curves with different distributions. Because of that, one difficulty in creating these histograms is to determine the number of blocks into which to partition the word length interval. The experiments in Section 5 investigate the effect of partitioning the word length into different numbers of blocks. In this work, we run trials partitioning the word length interval into blocks of three and ten, hence creating sets containing three and ten generative models.

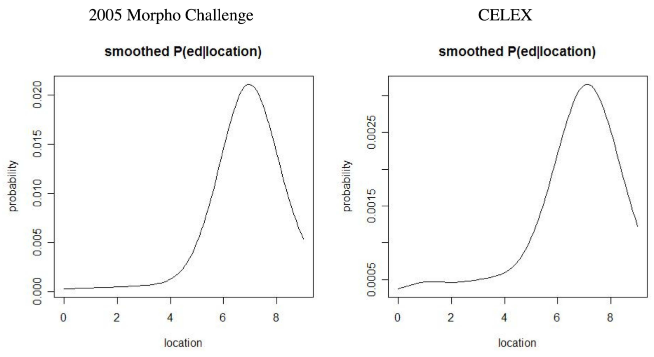

Figure 1: Histograms for the N-gram `ed` in the English language. The 2005 Morpho Challenge and CELEX wordlists were used to derive the histograms on the left and right respectively. In both cases, the histograms show that the N-gram `ed` most commonly occurs at the end of words. These histograms are constructed from ten generative models representing ten word locations equally spaced apart. Kernel density estimation was employed using a bandwidth of 2.5. This result should not be surprising, because `ed` is a common suffix in English.

A difficulty with these histograms is that they are not smooth. As a histogram for a particular N-gram is constructed using its corresponding maximum likelihood estimates from the set of generative models, it is possible and even likely that there will be blocks that contain zero probabilities. Fortunately, these histograms can be smoothed using kernel density estimation. The idea behind kernel density estimation is to relax the assumptions about the data in order to produce a smooth estimate of its density. The kernel density estimation smoothes the contribution of each data point in the histogram over the neighbourhood of the data point. The contribution of any data point x_i to the estimate at point x depends on the distance between x_i and x, as well as on the kernel density function, K, and the bandwidth, h of the function. Thus, the contribution of a data point x_i to any point x can be expressed as

$$f(x) = \frac{1}{n} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right),$$

where $f(x)$ is the smoothed curve for the histogram for a particular N-gram and n is the number of data points in the histogram. Each data point, x_i, is the maximum likelihood estimate of the N-gram from the i^th generative model. Here K is taken to be a Gaussian kernel. The setting of the bandwidth determines the degree of smoothing of the data. By varying the bandwidth used by the kernel density smoothing, the curve can be made flatter or steeper. The larger the bandwidth, the flatter the curve, and vice versa. Changing the curve has the effect of redistributing probability mass. Experiments which vary the bandwidth are discussed in Section 5.

These smoothed N-gram histograms are used to assign probabilities to candidate N-grams in order to split a term into two morphemes. The candidate N-grams are two N-grams, x and y, that together form the term, *term*, that is set to be the word currently being analyzed. The probability for each N-gram, $P(x)$ and $P(y)$, is extracted from their corresponding histograms using the location in which they occur in the word under analysis. This location is computed simply by taking the starting index of the N-gram and dividing it by the length of the word. The probabilities for the two N-grams are then multiplied together and compared with the probability for the term, $P(term)$. The probability for a term, given its location in the word, is derived by treating it like an N-gram and extracting its probability from its corresponding histogram. If the product of the N-gram probabilities is greater than the term probability, then a word split occurs. This splitting algorithm is shown formally in the equation below.

$$\begin{array}{ll} P(term) < \max_{x,y} P(x)P(y), & \text{split term into x and y} \\ P(term) > \max_{x,y} P(x)P(y), & \text{do not split term} \end{array}$$

The procedure is repeated recursively, by considering each of the N-grams x and y as terms, until no more likely splits can be made. The final N-grams of the recursive process are considered to be morphemes. This approach is motivated by the assumption that N-grams that are morphemes should appear independently with a greater probability than N-grams that are not morphemes.

4. Evaluation

The Swordfish2 algorithm is evaluated on two sets of trials, one that uses the wordlists and corresponding morphological analyses made available by the 2005 Morpho Challenge (MorphoChallenge, 2006) and one that uses the data and morphological analyses provided in CELEX (Baayen et al., 1996). The morphological analyses are lists of words that have been correctly split into morphemes and are used to assess the performance of morphological analyzers. The evaluation is based on the placement of the morpheme boundaries. A correctly placed boundary is known as a hit, an incorrect boundary is referred to as an insertion, and a missed boundary is called a deletion. The evaluation measures used are precision, recall and F-measure. Precision is defined as the ratio of hits to the sum of the number of hits and the number of insertions, whereas recall is the ratio of the number of hits to the sum of the number of hits and the number of deletions. F-measure is the harmonic mean of precision and recall. These metrics can be expressed as follows:

$$\begin{aligned} \mathit{Precision} &= \frac{\mathit{hits}}{\mathit{hits} + \mathit{insertions}} \\ \mathit{Recall} &= \frac{\mathit{hits}}{\mathit{hits} + \mathit{deletions}} \\ \mathit{F\text{-}measure} &= \frac{2 \cdot \mathit{Precision} \cdot \mathit{Recall}}{\mathit{Precision} + \mathit{Recall}} \\ &= \frac{2 \cdot \mathit{hits}}{2 \cdot \mathit{hits} + \mathit{insertions} + \mathit{deletions}} \end{aligned}$$

In this evaluation, word frequency is not considered; all terms are given equal importance, whether they occur in the corpus very commonly or very rarely. The 2005 Morpho Challenge morphological analyses that were made publicly available were a sample of the gold standard used in the actual competition and much smaller in size than the wordlists, as shown by Table 1. This sample gold standard was created with the intention of helping Morpho Challenge participants develop their solutions prior to the competition by testing them on a sample evaluation.

Language	Size of Word List	Size of Test Set
English	167377	532
Finnish	1636336	660
Turkish	582923	774

Table 1: The number of words in the 2005 Morpho Challenge word lists and test sets.

Language	Size of Word List	Size of Test Set
English	33737	23741
German	35053	11048

Table 2: The number of words in the CELEX word lists and test sets.

A more thorough evaluation is done using the CELEX data sets. While the wordlists provided in CELEX are much smaller, there are many words that have been morphologically analyzed, as shown in Table 2. The vocabulary data in CELEX are not entirely clean, however, and some measures were taken when creating the wordlists and test sets from these data. The largest difficulty with CELEX is that in addition to segmenting words into their component morphemes, they modify the root to match the original word from which it was derived. For example, `happily` would be segmented into `happy` + `ly` instead of `happi` + `ly`. Because such word segmentations would unfairly penalize any pure segmentation technique, all terms which CELEX modified while segmenting were dropped from our evaluation. Further, as CELEX was generated automatically and then hand-corrected, there are some oddities, such as words with no associated morphemes and terms that consist of multiple words. In both these cases the terms in question were dropped.

5. Trials

The trials for this paper were conducted on English, Finnish, Turkish, and German using the word lists provided by the 2005 Morpho Challenge (MorphoChallenge, 2006) and CELEX (Baayen et al., 1996). Tables 4 and 3 give the results for trials using the Swordfish2 algorithm in which we varied the number of partitions in the word length interval and the size of the bandwidth used for the kernel density estimation.

Partitions	Bandwidth	Precision	Recall	F-measure
English				
10	1.00	38.36	59.21	46.56
10	1.50	38.71	58.76	46.67
10	2.00	38.67	58.39	46.53
10	2.50	38.59	58.11	46.38
10	3.00	38.54	57.90	46.28
German				
10	1.00	32.89	39.85	36.04
10	1.50	33.30	39.64	36.19
10	2.00	33.27	39.57	36.15
10	2.50	33.53	39.74	36.37
10	3.00	33.53	39.77	36.38

Table 3: Results for Swordfish2 Algorithm on the CELEX data sets, varying the bandwidth used in the kernel density estimation.

Partitions	Bandwidth	Precision	Recall	F-measure
English				
3	1.00	60.12	39.00	47.31
3	1.50	58.43	37.45	45.65
3	2.00	57.14	37.07	44.96
3	2.50	56.14	37.07	44.65
3	3.00	54.39	35.91	43.26
10	1.00	58.33	40.86	48.05
10	1.50	58.76	40.47	47.93
10	2.00	59.02	41.54	48.76
10	2.50	59.34	41.54	48.87
10	3.00	59.02	41.54	48.76
Finnish				
3	1.00	72.09	24.22	36.26
3	1.50	71.64	23.85	35.79
3	2.00	71.02	23.29	35.08
3	2.50	71.54	23.42	35.28
3	3.00	70.57	23.23	34.95
10	1.00	70.57	25.45	37.41
10	1.50	70.03	25.37	37.25
10	2.00	69.56	25.37	37.18
10	2.50	68.88	25.12	36.82
10	3.00	68.43	24.88	36.49
Turkish				
3	1.00	59.06	16.17	25.39
3	1.50	60.49	16.58	26.03
3	2.00	60.73	16.74	26.25
3	2.50	60.49	16.58	26.03
3	3.00	60.33	16.47	25.88
10	1.00	60.59	17.27	26.88
10	1.50	61.76	17.63	27.43
10	2.00	61.48	17.68	27.47
10	2.50	60.77	17.68	27.39
10	3.00	61.08	17.72	27.47

Table 4: Results for Swordfish2 Algorithm on the 2005 Morpho Challenge data sets, varying both the number of partitions in the word length interval and the bandwidth used in the kernel density estimation.

Algorithm	Precision	Recall	F-measure
English			
Baseline	14.56	100.00	25.42
Morfessor 1.0	74.19	26.64	39.20
Porter's Algorithm	67.53	59.54	63.29
Swordfish	56.97	31.02	40.17
Swordfish2	63.40	43.81	51.81
Finnish			
Baseline	19.71	100.00	32.92
Morfessor 1.0	83.66	29.51	43.63
Swordfish	70.39	23.58	35.33
Swordfish2	68.88	25.12	36.82
Turkish			
Baseline	25.87	100.00	41.11
Morfessor 1.0	76.33	24.17	36.71
Swordfish	58.90	16.79	26.13
Swordfish2	60.77	17.68	27.39

Table 5: Comparison of algorithms on the 2005 Morpho Challenge data sets. The results shown for Swordfish2 are for the trials using a word length interval partitioned into ten blocks and a 2.5 bandwidth for the kernel density estimation.

Tables 5 and 6 give a comparison of Swordfish2 with approximately the best parameter setting against the results from a baseline approach, the Morfessor 1.0 algorithm, a variation of Porter's algorithm that does morpheme segmentation, and the original Swordfish algorithm. The baseline approach is the result of placing a split between every character for every term; refer to Section 2 for a discussion of the Morfessor, Porter's, and Swordfish algorithms. The variation of Porter's algorithm is used only in trials over English, because it is a rule-based approach designed only to work on English.

The results in both Tables 5 and 6 show that for each of the trials, with the exception of Turkish, Swordfish2 gives better results for precision and the F-measure evaluation metrics than the baseline. The results also seem to indicate that when the number of partitions in the word length interval is low, kernel density estimation with a small bandwidth to smooth the N-gram histograms gives better results than with a larger one. For the trials using three partitions, a bandwidth of 1.0 gives the best results for English and Finnish in terms of precision, recall and the F-measure. When ten partitions are used, having a larger bandwidth of 2.0 to 2.5 tends to give the most solid results over all metrics.

The results for Swordfish2 in Tables 5 and 6 are using a ten block partition for the word length interval with a bandwidth of 2.5 for the kernel density estimation. These values were selected because they gave reasonable performance compared to the other parameter settings that were evaluated in this paper. The results for Morfessor over the 2005 Morpho Challenge data sets differ from what is posted on the Morpho Challenge Website: `http://www.cis.hut.fi/morphochallenge2005/results.shtml`. This difference is due to this work using the publicly available sample gold standard provided by the 2005 Morpho Challenge instead of the full gold standard used in the actual competition. As well, no parameter tuning was done for Morfessor in this work. The implementation of Morfessor that was used in this

Algorithm	Precision	Recall	F-measure
English			
Baseline	8.08	100.00	14.95
Morfessor 1.0	69.70	29.18	41.14
Porter's Algorithm	28.33	28.18	28.25
Swordfish	45.60	61.59	52.41
Swordfish2	38.04	56.41	45.44
German			
Baseline	7.43	100.00	13.84
Morfessor 1.0	71.24	29.15	41.37
Swordfish	31.46	35.05	33.16
Swordfish2	32.06	39.48	35.39

Table 6: Comparison of algorithms on the CELEX data sets. The results shown for Swordfish2 are for the trials using a word length interval partitioned into ten blocks and a 2.5 bandwidth for the kernel density estimation.

work is provided on the Helsinki University of Technology Morpho project Website: `http://www.cis.hut.fi/projects/morpho/`.

Swordfish2 outperforms Morfessor in terms of recall and the F-measure on English, but Morfessor has higher precision. On the 2005 Morpho Challenge data sets, Table 5, the variation of the rule-based Porter's algorithm achieves the best results, outperforming all the other algorithms. On the CELEX English data set, however, Table 6, the variation of Porter's algorithm returns much poorer results over all metrics than do either Swordfish or Swordfish2. This may be at least partly explained by the fact that the subset of the CELEX data set that was used here did not contain any terms that performed character substitution during segmentation. Obviously this subset of terms was much more difficult for Porter's algorithm to analyze correctly. For rule-based algorithms such as Porter's to deal with this particular subset, they would need to have their rules modified. This illustrates one of the limitations of rule-based approaches. Also recall that such a rule-based approach is language-dependent, and thus can only be applied to the English data sets. Morfessor, Swordfish, and Swordfish2 all have the advantage of being language-independent and have had trials run on each of the language data sets.

Statistical analyses of these summary results in Tables 5 and 6 are shown in Figures 2, 3, and 4. These figures indicate the significant differences in precision, recall and F-measure respectively for each algorithm and data set. A bootstrap (Efron and Tibshirani, 1993) is performed in order to generate non-parametric confidence intervals around results. The idea behind the bootstrap is to sample, with replacement, the words for which we know the correct morpheme splits. This process continues until the new sample contains as many words as were contained within the initial set. This ensures that approximately $\frac{2}{3}$ of our data will be present within each sample. Precision, recall and F-measure scores are generated for each such sample. This process is repeated with each score treated as a draw from some underlying non-parametric distribution. A large number of bootstrapped samples results in a good approximation of the underlying distribution. For the analyses that we present here, we used 10,000 samples. By employing this empirical approximation, a 95% confidence interval can be taken to be the region which contains approximately 95% of the observed measurements.

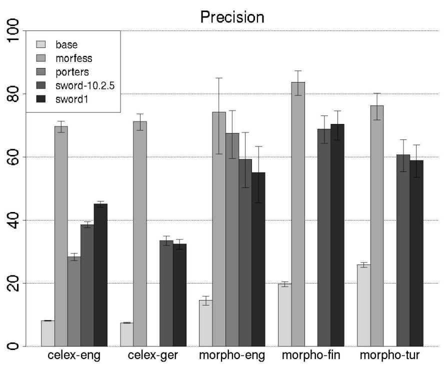

Figure 2: Comparisons of precision between the baseline (base), Morfessor (morfess), Porter's (porters), Swordfish 1 (sword1), and Swordfish2 using ten blocks and a bandwidth of 2.5 (sword-10.2.5). The data were bootstrapped and the bars represent the mean precision with 95% confidence intervals around each bar.

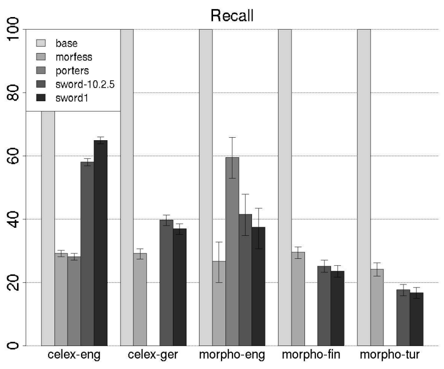

Figure 3: Comparisons of recall between the baseline (base), Morfessor (morfess), Porter's (porters), Swordfish 1 (sword1), and Swordfish2 using ten blocks and a bandwidth of 2.5 (sword-10.2.5). The data were bootstrapped and the bars represent the mean recall with 95% confidence intervals around each bar.

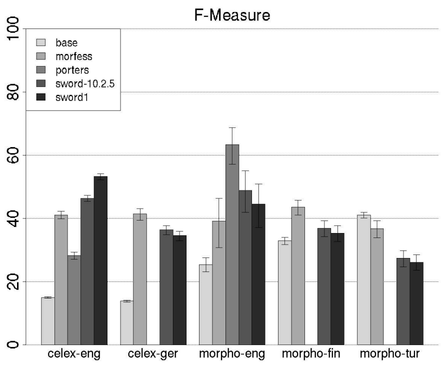

Figure 4: Comparisons of F-measure between the baseline (base), Morfessor (morfess), Porter's (porters), Swordfish 1 (sword1), and Swordfish2 using ten blocks and a bandwidth of 2.5 (sword-10.2.5). The data were bootstrapped and the bars represent the mean F-measure with 95% confidence intervals around each bar.

Algorithm	Precision	Recall	F-measure
Swordfish	49.02	46.90	47.94
Swordfish2	41.56	41.95	41.75

Table 7: Comparison between Swordfish and Swordfish2 using the 2005 Morpho Challenge data set for building the models and the CELEX English data set for evaluation.

Figures 2, 3, and 4 implicitly compare the results of the four algorithms (five in the case of English). In order to account for the implicit multiple comparisons, we use the Bonferroni correction (Miller, 1981) to inflate our confidence intervals to take into account these multiple comparisons. In terms of F-measure, Porter's algorithm was significantly better than the other algorithms over the English Morpho Challenge data, but significantly worse over the English CELEX data. Swordfish and Swordfish2 are only significantly different over the English CELEX data. Morfessor is significantly better than the Swordfish algorithm over the non-English data. Both the Swordfish algorithms, however, are significantly better than Morfessor over the English CELEX data.

Perhaps the most significant result of these trials is that for all but one of the data sets, there is not enough evidence to indicate that Swordfish2 is significantly different from Swordfish; over the English CELEX data, Swordfish is significantly better than Swordfish2. This result is somewhat disappointing, as the summary results from Tables 5 and 6 seem to indicate that Swordfish2 was an improvement over Swordfish. For instance, on the English Morpho Challenge data, the F-Measure for Swordfish2 is 11% greater than it is for Swordfish. One of the reasons for the lack of significant difference in the trials over the Morpho Challenge data is that the sizes of its test sets, the gold standards, are quite small. The test sets in CELEX, on the other hand, are much larger, thus making it easier to find significant differences.

One of the issues with the CELEX data, however, is that the word lists are quite small. As the models constructed here depend on term frequencies, smaller word lists will have a negative effect on them. This effect is much greater on Swordfish2 than on Swordfish; Swordfish2 divides N-grams frequencies into different word length interval partitions, while Swordfish does not. Table 7 displays the results of using the 2005 Morpho Challenge data set for building the probabilistic models and CELEX for evaluation. Only English was examined, because that is the only language that the Morpho Challenge and CELEX data sets have in common. For both Swordfish and Swordfish2, the precision improves, while recall drops in value. These scores, however, are less than those from the evaluation using just the Morpho Challenge data shown in Table 5. This result indicates that increasing the number of words used for building the models improves precision, but at the expense of recall. As well, there appear to be some differences between the words in Morpho Challenge and CELEX test sets in terms of morphological analysis. It was expected that by building the model using the Morpho Challenge data, the performance on the CELEX test set would be similar to the performance on the Morpho Challenge test set.

Further analysis of Swordfish and Swordfish2 reveals that they do not agree on every word split. Table 8 displays the percentage of word splits Swordfish makes that Swordfish2 also makes. For all languages, Swordfish2 agrees with Swordfish more than 55% of the time. The precision of those splits is slightly higher than the precision of both Swordfish and Swordfish2. Table 9 displays the precision of the word splits on which Swordfish and Swordfish2 do not agree. The precision of these splits is lower than the precision of either algorithm respectively.

Data Set	Agreement	Precision
2005 Morpho Challenge		
English	76.0%	66.77
Finnish	56.1%	70.76
Turkish	73.0%	64.08
CELEX		
English	73.6%	49.86
German	69.6%	34.64

Table 8: The percentage on which Swordfish2 agrees with the split that Swordfish makes across all data sets, and the precision of those splits.

Data Set	Swordfish Precision	Swordfish2 Precision
2005 Morpho Challenge		
English	26.00	58.41
Finnish	58.90	57.69
Turkish	44.93	53.06
CELEX		
English	33.72	13.97
German	24.21	27.68

Table 9: The precision of the word splits on which Swordfish and Swordfish2 disagree.

Thus, the word splits that the Swordfish algorithms disagree on tend to be incorrect.

Despite the lack of significant difference in performance and the levels of disagreement over word splits, we feel that we have successfully incorporated N-gram within word location into the Swordfish2 algorithm. This additional information about N-grams is used to detect morphemes in words. We believe that one of the main reasons why many of these results are not significantly different is because of the deficiencies in the data sets. Even with these issues, these results do provide some new evidence that supports our hypothesis that the most probable N-grams are morphemes.

Note that neither memory usage nor computation time is considered in these evaluations.

6. Conclusion

Swordfish2 is an adaptation of Swordfish, an unsupervised N-gram approach to morphological analysis. As with the original Swordfish algorithm, this approach uses unsupervised learning based on N-grams to separate a word into morphemes. Swordfish2, however, incorporates information about the location of N-grams in a word when determining the likelihood of a particular morpheme split. In addition, Swordfish2 uses a kernel density smoothing technique when building the histogram representations for each N-gram. Unfortunately, the results given in Section 5 indicate that these extensions to the original Swordfish algorithm do not improve the morphological analysis significantly.

The Swordfish2 algorithm is language-independent, making it an attractive alternative to traditional language-dependent rule-based approaches. Future work will include trials of

Swordfish2 on other languages. Future areas of investigation include performing trials that limit the length of the N-grams used in the probability comparison. Hence instead of comparing N-grams that compose a word, N-grams that compose a substring of the word will be examined. In theory, this should help deal with less commonly occurring words. Incorporating rule-based feedback to filter known non-morpheme N-grams has the potential to improve the model greatly. We believe that it is not entirely reasonable to have a completely unsupervised approach for doing morphological analysis. It may be more suitable for Swordfish2 to provide a solid starting model which a linguist can begin with and modify quickly to achieve higher levels of performance. Investigation of an effective feedback mechanism needs to be done.

The Swordfish2 algorithm is an extension of the original Swordfish algorithm, which is simply a generative model created with the maximum likelihood estimate. While this new approach creates multiple generative models to represent different word locations, it too is simple in nature. Both Swordfish algorithms show that N-gram probabilities can be used to identify morphemes in words. Exploring different ways to extend these N-gram models should lead to greater refinements and improved detection of morphemes.

It is important to realize the significance of this work, which is fairly easy to lose sight of when examining performance metrics. The importance is not that we have devised a method of conducting morphological analysis in an unsupervised manner; it is that we have provided evidence as to why character N-grams are semantically important. Until recently, character N-grams have often been disregarded in information retrieval and their usefulness questioned. Both the results of Swordfish and Swordfish2 indicate that the most prominent N-grams are also morphemes. We have not completely proven our hypothesis to be true, but we have provided significant evidence that supports it. Definitively showing that the most probable N-grams are indeed morphemes will require better probabilistic models. How to build these models remains an open question. Discovering how to create them will lead to a greater understanding of how our language works and how we can understand our own vocabulary.

Acknowledgement

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

References

M. Adler and M. Elhadad. 2006. An Unsupervised Morpheme-Based HMM for Hebrew Morphological Disambiguation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 665–672, Sydney, Australia, July. Association for Computational Linguistics.

R. H. Baayen, R. Piepenbrock, and L. Gulikers. 1996. *CELEX2.* Linguistic Data Consortium, Philadelphia, PA.

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

D. Bernhard. 2006. Unsupervised Morphological Segmentation Based on Segment Predictability and Word Segments Alignment. In *Proceedings of the PASCAL Challenge Workshop on Unsupervised segmentation of words into morphemes.*

M. Creutz and K. Lagus. 2005. Unsupervised Morpheme Segmentation and Morphology Induction from Text Corpora using Morfessor 1.0. Technical report, Helsinki University of Technology.

B. Efron and R. J. Tibshirani. 1993. *An introduction to the bootstrap.* Chapman & Hall.

W. B. Frakes and R. A. Baeza-Yates, editors. 1992. *Information Retrieval: Data Structures & Algorithms.* Prentice-Hall.

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

T. Hastie, R. Tibshirani, and J. Friedman. 2001. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer.

T. Hirsimäki, M. Creutz, V. Siivola, M. Kurimo, S. Virpioja, and J. Pylkkönen. 2006. Unlimited vocabulary speech recognition with morph language models applied to finnish. *Computer, Speech and Language*, 20(4):515–541.

C. Jordan, J. Healy, and V. Keselj. 2006a. Swordfish – An Unsupervised Ngram Based Approach to Morphological Analysis. In *SIGIR '06: Proceedings of the 29rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, New York, NY, USA. ACM Press.

C. Jordan, J. Healy, and V. Keselj. 2006b. Swordfish – Using Ngrams in an Unsupervised Approach to Morphological Analysis. In *Proceedings of the PASCAL Challenge Workshop on Unsupervised Segmentation of Words into Morphemes.*

M. Kantrowitz, B. Mohit, and V. Mittal. 2000. Stemming and its effects on TFIDF ranking (poster session). In *SIGIR '00: Proceedings of the 23rd annual international ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 357–359, New York, NY, USA. ACM Press.

S. Keshava and E. Pitler. 2006. A Simpler, Intuitive Approach to Morpheme Induction. In *Proceedings of the PASCAL Challenge Workshop on Unsupervised Segmentation of Words into Morphemes.*

R. Krovetz. 1993. Viewing Morphology as an Inference Process. In *SIGIR '93: Proceedings of the 16th Annual International Conference on Research and Development in Information Retrieval*, pages 191–203.

J. Lovins. 1968. Development of a Stemming Algorithm. *Mechanical Translation and Computational Linguistics*, 11:22–31.

E. C. Marsi, A. van den Bosch, and A. Soudi. 2005. Memory-based morphological analysis generation and part-of-speech tagging of Arabic. In *Proceedings of the ACL 2005 workshop on computational approaches languages to Semitic languages*, Ann Arbor, MI, USA. Association for Computational Linguistics.

R. G. Miller. 1981. *Simultaneous Statistical Inference.* Springer-Verlag.

MorphoChallenge. 2006. Unsupervised segmentation of words into morphemes challenge 2005. http://www.cis.hut.fi/morphochallenge2005/.

M. Popovic and P. Willett. 1992. The effectiveness of stemming for natural-language access to slovene textual data. *Journal of the American Society for Information Science*, 43(5):384–390.

M. F. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

M. F. Porter. 2001. Snowball: A language for stemming algorithms. http://snowball.tartarus.org/, October. Last access in April 2007.

E. J. Schuegraf and H. S. Heaps. 1976. Query processing in a retrospective document retrieval system that uses word fragments as language elements. *Information Processing and Management*, 12(4):283–292.

R. Wicentowski. 2004. Multilingual Noise-Robust Supervised Morphological Analysis using the WordFrame Model. In *Proceedings of Seventh Meeting of the ACL Special Interest Group on Computational Phonology (SIGPHON)*, pages 70–77.

M. Yamamoto and K. W. Church. 2001. Using suffix arrays to compute term frequency and document frequency for all substrings in a corpus. *Computational Linguistics*, 27(1):1–30.

C. Zhai and J. Lafferty. 2001. A study of smoothing methods for language models applied to Ad Hoc information retrieval. In *SIGIR '01: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 334–342, New York, NY, USA. ACM Press.