Report Number: WUCSE-2002-18

2002-04-18

An Unsupervised Knowledge Free Algorithm for the Learning of Morphology in Natural Languages - Master's Thesis, May 2002

Matthew G. Snover

This thesis describes an unsupervised system to learn natural language morphology, specifically suffix identification from unannotated text. The system is language independent, so that is can learn the morphology of any human language. For English this means identifying “-s”, “-ing”, “-ed”, “-tion” and many other suffixes, in addition to learning which stems they attach to. The system uses no prior knowledge, such as part of speech tags, and learns the morphology by simply reading in a body of unannotated text. The system consists of a generative probabilistic model which is used to evaluate hypotheses, and a directed search and... Read complete abstract on page 2.

An Unsupervised Knowledge Free Algorithm for the Learning of Morphology in Natural Languages - Master's Thesis, May 2002

Matthew G. Snover

Complete Abstract:

This thesis describes an unsupervised system to learn natural language morphology, specifically suffix identification from unannotated text. The system is language independent, so that is can learn the morphology of any human language. For English this means identifying “-s”, “-ing”, “-ed”, “-tion” and many other suffixes, in addition to learning which stems they attach to. The system uses no prior knowledge, such as part of speech tags, and learns the morphology by simply reading in a body of unannotated text. The system consists of a generative probabilistic model which is used to evaluate hypotheses, and a directed search and a hill-climbing search which are used in conjunction to find a highly probably hypothesis. Experiments applying the system to English and Polish are described.

2002-18

An Unsupervised Knowledge Free Algorithm for the Learning of Morphology in Natural Languages - Master's Thesis, May 2002

Authors: Snover, Matthew G.

April 18, 2002

Abstract: This thesis describes an unsupervised system to learn natural language morphology, specifically suffix identification, from unannotated text. The system is language independent, so that it can learn the morphology of any human language. For English this means identifying "-s", "-ing", "-ed", "-tion" and many other suffixes, in addition to learning which stems they attach to. The system uses no prior knowledge, such as part of speech tags, and learns the morphology by simply reading in a body of unannotated text. The system consists of a generative probabilistic model which is used to evaluate hypotheses, and a directed search and a hill-climbing search which are used in conjunction to find a highly probable hypothesis. Experiments applying the system to English and Polish are described.

DEPARTMENT OF COMPUTER SCIENCE

AN UNSUPERVISED KNOWLEDGE FREE ALGORITHM FOR THE LEARNING OF MORPHOLOGY IN NATURAL LANGUAGES

by

Matthew G. Snover, B.S.

Prepared under the direction of Professor Michael R. Brent

A thesis presented to the Sever Institute of
Washington University in partial fulfillment
of the requirements for the degree of

Master of Science

May, 2002

Saint Louis, Missouri

AN UNSUPERVISED KNOWLEDGE FREE ALGORITHM FOR THE LEARNING OF MORPHOLOGY IN NATURAL LANGUAGES

by Matthew G. Snover

ADVISOR: Professor Michael R. Brent

May, 2002

Saint Louis, Missouri

This thesis describes an unsupervised system to learn natural language morphology, specifically suffix identification, from unannotated text. The system is language independent, so that it can learn the morphology of any human language. For English this means identifying "-s", "-ing", "-ed", , "-tion" and many other suffixes, in addition to learning which stems they attach to. The system uses no prior knowledge, such as part of speech tags, and learns the morphology by simply reading in a body of unannotated text. The system consists of a generative probabilistic model which is used to evaluate hypotheses, and a directed search and a hill-climbing search which are used in conjunction to find a highly probable hypothesis. Experiments applying the system to English and Polish are described.

Introduction

1.1 What is Morphology?

Natural language morphology dictate the rules for the the structure of words in a language. Words, in all languages, are constructed of morphemes which combine to give meaning to a word. There are two types of morphemes, stems and affixes. Stems contribute the base meaning of the word, while affixes either add additional meaning. There are three types of affixes, prefixes, which attach to the beginning of words, suffixes, which attach to end of words, and infixes which are inserted into the middle of words. The change in meaning from affixes comes either as a part of speech change, such as from adjective to adverb when the suffix *-ly* is used, or as semantic alteration, such as when the prefix *un-* in English is used to negate meaning.

Knowledge of morphology is useful for several reasons in Natural Language Processing. Limited solely to the field of linguistics, morphology is an integral part of language, directly related to phonology and syntax. From a purely computational viewpoint, knowledge of morphology can be used to reduce the size of language lexicons. Rather than having to store a stem with all possible affixes, one can simply store the stem, and the rules which generalize the morphology to all the stems. Morphology is also useful in natural language processing for its aid in part of speech tagging, speech segmentation, and parsing.

An obvious solution to the problem is the automatic generation of morphological analyses, a task which intuitively should be feasible as children learn morphology without direct supervision. Such an algorithm would need to be unsupervised so that no initially annotated data is required, as well as being knowledge free and language independent. The knowledge free requirement dictates that the algorithm should not require any other annotated data from the language, such as part of speech tags, as such annotations are likely to be just as difficult and costly to acquire as the morphological analysis. The algorithm should be language independent so it can run on any language without detailed knowledge of language being required.

The algorithm could then be used to annotate a new corpus for languages which do not already have good morphological lexicons. The annotation need not be perfect, but should be highly reliable with few mistakes. This annotation could then be improved using other methods, either by first aiding in the bootstrapping of part of speech tags and then refining the morphological analysis, or by use of adjustment rule learning algorithms.

1.3 The Goal

The general goal of this work is to take a list of common words from a language and deduce the morphological structure of those words without any annotated data or prior knowledge, such as part of speech tags. In specific these algorithms only address final suffix detection, and do not attempt to learn prefix, infix or reduplicative morphology. The algorithm described does not attempt to learn stem or suffix adjustment rules and only predicts concatenative morphology. Each word is only allowed one morphological analysis by the algorithm, which would be a concern in agglutinative languages such as Turkish, where there are frequently multiple morphological analyses of a word, each with separate meaning, that can only be distinguished on the basis of context.

Ideally, the algorithm would produce the output words given the input words such as shown in Table 1.1. The ϵ that attaches to *build* represents the empty string and is reported as the suffix when no actual suffix is detected. The algorithm actually treats ϵ as a real suffix for detection purposes. Because the algorithm is only concerned with final suffix detection, it should find that *buildings* should be analyzed as *building* + *s* and not as *build* + *ing* + *s*, which is in fact the correct analysis.

1.4 Previous Work

1.4.1 Supervised Learning

Van den Bosch and Daelemans[16] approach morphological learning with an instance or memory based learning algorithm. Such an approach is a useful way to generalize from a preestablished morphological lexicons, but it cannot help in the annotation of new lexicons. Instance based learning could be applied to a preliminary hypothesis generated by some of the unsupervised algorithms described below, allowing for ease of prediction of the morphological forms of new words.

1.4.2 Unsupervised Learning

Minimum Description Length Models

Minimum Description Length, MDL,[13] has been shown to be effective in supervised machine learning, such as Quinlan and Rivest's[12] use of MDL to help prune decision trees. It is also highly suited for unsupervised learning and has been used in several algorithms for the unsupervised learning of morphology. The Minimum Description Length Principle lends itself to morphology, since the problem can be viewed as finding the smallest set of stems and affixes to describe the observed words.

uses Expectation Maximization (EM) as the search method. In addition Goldsmith's model incorporates the notion of suffix signatures to the MDL model. A suffix signature, similar to the notion of a paradigm which is described later, is a group of suffixes which all attach to the same stems. His algorithm, also known as Linguistica, is considered one of the most successful unsupervised techniques to learn morphology. It does not focus solely on productive morphology but also detects word concatenation, as well as more historical morphological features. The algorithm predicts large numbers of suffixes and often misanalysizes many rarer words.

The system described in this paper is a probabilistic model and is not based on MDL, but could be translated into such a framework with minor mathematical manipulation.

Other Models

In addition to MDL driven algorithms, there have been several other unsupervised algorithms developed.

Déjean[7] uses a bootstrapping method which detects frequent morphemes and then attempts to use those to learn more morphemes and apply to them to the list of words. The end goal of his algorithm is to aid in chunking. He also uses the notion of suffix signatures, but the approach is rather ad hoc, and it is unclear how well it would generalize to other languages.

Gaussier[8] uses an explicitly probabilistic model to derive morphology from word pairs. The algorithm uses part of speech tags to learn derivational morphology. Because of this requirement the algorithm is not knowledge free and would be unsuitable for morphology discovery in unannotated languages where part of speech information is not readily available. If a highly reliable method for unsupervised learning of part of speech was developed then the algorithm might have more practical value.

to be a very promising technique, but Schone and Jurafsky emphasis the need to incorporate it with more sophisticated orthographic methods for learning morphology.

1.5 Preview

The algorithm described in this thesis is composed of two components, a probabilistic model and a search algorithm. The search algorithm is used to find a local maxima in the probability space defined by both the model and the input words.

Chapter 2 describes several versions of a probabilistic model that is used to evaluate hypotheses. The formulation of the models are described in a generative framework.

Chapter 3 discusses algorithms used to search the hypothesis space and maximize the probabilities described in Chapter 2. A lattice representation of the hypothesis is described and then used in the formulation of a a simple search. This search proves inadequate though, and necessitates the introduction of a better initial hypothesis.

Experimental testing of the algorithm is presented in Chapter 4 as well as empirical comparisons between different variants of the probability model and search techniques. The algorithm is also compared to Goldsmith's Linguistica morphology learning system[9], considered to be the best current unsupervised system for learning morphology. Section 4.2 details the methods used for evaluation of empirical results. Evaluation of morphological results is difficult due to the high degree of ambiguity present in morphology, and the inherent limitations of concatenative models. Two methods of evaluation, which measure the accuracy of the stems and suffixes identified separately are described.

Chapter 5 summarizes the work, draws conclusions from it and presents ideas for future development and research.

A Generative Probabilistic Model of Concatenative Morphology

This chapter introduces several variants of a language-independent prior probability distribution on all possible hypotheses, where a hypothesis is a set of words, each of which is marked with a stem-suffix boundary. The distributions are based on a multiple step model for the generation of hypotheses. Associated with each step is a probability distribution for the various choices that could be made at that step. By taking the product over all steps of the generative model, one can calculate the prior probability for any given hypothesis. The steps described are a mathematical model used to calculate a probability for a hypothesis, not an algorithm intended to be run.

This technique for calculating a probability model is fairly common and has been successfully used in other computational linguistics tasks, such as word segmentation[3].

In addition, a naive model, which is based on some simple heuristics is described. It is not a probabilistic model, but provides some intutions that motivate models described later.

2.1 Naive Model

The most naive scoring method for morphological hypotheses is to minimize the sum of the number of characters in the stem and suffix sets. While such a model performs very poorly as it fails to identify many of the morphological patterns that exist in language, it does give a good starting point for constructing a better scoring system.

This scoring system will make horrible mistakes. For instance given the words *at*, *accomplishment*, and *abolish*, the best hypothesis from this naive model would be

Insertions and deletions are given a cost of 1 and substitutions are given a cost of 2. For every pair of words that are postulated to be morphologically related, the min-edit-distance is calculated and added to the cost of the model. Two words are said to be morphologically related if they have the same stem. The naive model can thus be reformulated such that cost of a hypothesis is the number of characters in the stem and suffix sets, plus the minimum edit distance of every pair of words which have the same stem. The search will then attempt to minimize the cost of the hypothesis.

The hypothesis given above for *at*, *accomplishment*, and *abolish*, in which all words have the stem *a*, will have a cost of 46, (19 for the number of characters, 11 for the min-edit-dist of *at* and *accomplishment*, 7 for the min-edit-dist of *at* and *abolish* and 9 for the min-edit-dist of *accomplishment* and *abolish*). This is far worse than the hypothesis where all of the words are not morphologically related and are their own stems, which has a cost of 21. The best hypothesis for this set of words according to the new model has a cost of 20 and is where *t* is a suffix on the stems *a* and *accomplishmen*, and *abolish* is its own stem with no suffix. There is no edit distance between *a* and *accomplishment* because the two words do not share a stem. This is still not perfect but it is a large improvement on the original naive formulation.

The modification is useful because it discourages long suffixes, and it penalizes heavily for stems with lots of suffixes, due to the number of minimum edit distances that would be calculated.

2.2 Simple Model

The following is a five step process describing how a hypothesis could be constructed along with associated probability distributions.

throughout this paper.
Example: $M = 5$. $X = 3$.

2. For each stem i, choose its length in letters, L_i^m, according to the inverse squared distribution on the positive integers. Assuming that the stem lengths are chosen independently and multiplying together their probabilities, we have:

$$\Pr(L^m \mid M) = \left(\frac{6}{\pi^2}\right)^M \prod_{i=1}^{M} \left(\frac{1}{L_i^m}\right)^2 \tag{2.2}$$

For each suffix i, choose its length in letters, L_i^x, according to a similar distribution to (2.2), which differs in that suffixes of length 0 are allowed, by offsetting the length by one.
Example: $L^m = 4, 4, 4, 3, 3$. $L^x = 2, 0, 1$.

3. Let Σ be the alphabet and let $\{p_1 \ldots p_{|\Sigma|}\}$ be a probability distribution on Σ. For each i from 1 to M, generate stem i by choosing L_i^m letters at random, according to the probabilities $\{p_1 \ldots p_{|\Sigma|}\}$. Call the resulting stem set STEM. Similarly, for each i from 1 to X, generate suffix i by choosing L_i^x letters at random, according to the probabilities $\{p_1 \ldots p_{|\Sigma|}\}$. Call the resulting suffix set SUFF. To compute the joint probability of hypothesized stem and suffix sets under this model we use the maximum likelihood estimates of the letter probabilities:

$$\hat{p}_l = \frac{c_l}{S}$$

where c_l is the frequency count of letter l among all the hypothesized stems and suffixes, and $S = \sum_l c_l$. Thus,

$$\Pr(\text{STEM}, \text{SUFF} \mid M, L^m, X, L^x) = M!X! \prod_{l \in \Sigma} \left(\frac{c_l}{S}\right)^{c_l} \tag{2.3}$$

$$\Pr(\text{Freq}_i \mid M, X) = \frac{1}{X}$$

Assuming all these choices are made independently and multiplying together their probabilities yields:

$$\Pr(\text{Freq} \mid M, X) = \left(\frac{1}{X}\right)^M \tag{2.4}$$

Example: $Freq = \{3, 3, 2, 1, 2\}$

5. For each stem i in STEM, choose a set of suffixes, D_i, of size Freq_i, that i will be paired with in order to generate the lexicon. The number of subsets of a given size is finite, so we can again use the uniform distribution. This implies that the probability of each individual subset of size Freq_i is the inverse of the total number of such subsets:

$$\Pr(D_i \mid M, X, \text{Freq}_i) = \binom{X}{\text{Freq}_i}^{-1}$$

Assuming that all these choices are independent yields:

$$\Pr(D \mid M, X, \text{Freq}) = \prod_{i=1}^{M} \binom{X}{\text{Freq}_i}^{-1} \tag{2.5}$$

Example: D_{walk} ={ *ed*, ϵ, *s* }, D_{look} ={ *ed*, ϵ, *s* }, D_{door} ={ ϵ, *s* }, D_{far} ={ ϵ }, D_{cat} ={ ϵ, *s* }

From the results of step 5 one can see that running example would yield the hypothesis consisting of the set of words with suffix breaks, {*walk+*ϵ, *walk+s, walk+ed, look+*ϵ, *look+s, look+ed, far+*ϵ, *door+*ϵ, *door+s, cat+*ϵ, *cat+s*}. Removing the breaks in the words results in the set of input words. To find the probability for this hypothesis just take the product of the probabilities from equations (2.1)-(2.5).

- The number of suffixes that occur with each stem (Freq)
- The count of each letter in the combined stem and suffix sets ($\{c_l \mid l \in \Sigma\}$).

Using this generative model, we can assign a probability to any hypothesis. Typically one wishes to know the probability of the hypothesis given the data, however in our case such a distribution is not required. Equation (2.6) shows how the probability of the hypothesis given the data could be derived from Bayes law.

$$\Pr(\text{Hyp} \mid \text{Data}) = \frac{\Pr(\text{Hyp}) \Pr(\text{Data} \mid \text{Hyp})}{\Pr(\text{Data})} \tag{2.6}$$

The search algorithm described later only considers hypotheses consistent with the data. The probability of the data given the hypothesis, Pr(Data|Hyp), is always 1, since if you remove the breaks from any hypothesis, the input data is produced. This would not be the case if our search considered inconsistent hypothesis. The prior probability of the data is unknown, but is constant over all hypotheses. Thus the probability of the hypothesis given the data reduces to $\Pr(\text{Hyp})/c$. The probability of the hypothesis is given by the above generative process. Thus, among all consistent hypotheses, the one with the greatest prior probability also has the greatest posterior probability.

When considering varying hypotheses for a given input the most probable hypothesis will be determined based on several forces. The simple model favors hypotheses with less stems and suffixes, and then those hypotheses with shorter stems and suffixes. The primary force of the model is to minimize the number of characters in the stem and suffix sets.

2.3 Paradigm Model

The simple model described above ignores some vital information about morphology, namely that the suffixes that attach to stems do so in regular and consistent groups.

suffixes of their paradigm and no others.

The Paradigm Model uses the same initial steps, 1-3 as the Simple Model for generating the stem and suffix sets but uses a different method to combine stems and suffixes. Steps 4-5 are replaced in the Paradigm model with the following steps:

4. Pick the number of paradigms, P. Each stem is in exactly one paradigm, and each paradigm has at least one stem, therefore the number of paradigms, P, can range from 1 to M. We pick P according to the following uniform distribution:

$$\Pr(P \mid M) = \frac{1}{M} \tag{2.7}$$

   Example: $P = 3$.

5. We choose the number of suffixes in the paradigms, R, according to a uniform distribution. The distribution for picking R_i, suffixes for paradigm i is:

$$\Pr(R_i \mid XP) = \frac{1}{X}$$

   The joint probability over all paradigms, R is therefore:

$$\Pr(R \mid XP) = \prod_{i=1}^{P} X^{-1} = \left(\frac{1}{X}\right)^{P} \tag{2.8}$$

   Example: $R = \{2, 1, 2\}$.

6. For each paradigm i, choose the set of R_i suffixes, PARA_i^x that the paradigm will represent. The number of subsets of a given size is finite so we can again use the uniform distribution. This implies that the probability of each individual subset of size R_i, is the inverse of the total number of such subsets. Assuming

possible distributions that could be used at this step, a uniform distribution, which makes all paradigms equally likely and a biased distribution that favors paradigms with more stems. Under the uniform distribution the probability of picking any particular paradigm is $\frac{1}{P}$. Taking the product over all stems yields:

$$\Pr(\text{PARA}^m \mid MP) = \left(\frac{1}{P}\right)^M \tag{2.10}$$

Using the biased distribution, the probability of choosing a paradigm i, for a stem is calulcated using a maximum likelihood estimate:

$$\Pr(\text{PARA}_i^m | MP) = \frac{|\text{PARA}_i^m|}{M}$$

where PARA_i^m is the set of stems in paradigm i. Assuming that all these choices are made independently yields the following:

$$\Pr(\text{PARA}^m \mid MXP) = \prod_{i=1}^{P} \left(\frac{|\text{PARA}_i^m|}{M}\right)^{|\text{PARA}_i^m|} \tag{2.11}$$

Example: $PARA_1^m = \{walk, look\}$. $PARA_2^m = \{far\}$. $PARA_3^m = \{door, cat\}$.

The hypothesis generated by the running example for the Paradigm Model is the same as the example for the Simple Model, though different probabilities would have been calculated. To find the probability for this hypothesis just take the product of the probabilities from equations (2.1)-(2.3) and equations (2.7)-(2.10). Equation (2.11) can be substituted for equation (2.10) depending on which distribution is used for step 7.

The two distributions described in step 7 can be thought of as two different priors on the selection of paradigms by stems. The biased distribution in equation 2.11 favors hypothesis where more stems are in a smaller number of paradigms, whereas

using the uniform distribution.

The main force on the probability space exerted by the Paradigm Model is the minimization of the number of paradigms. The hypotheses that are favored are more structured. The minimization of paradigms is actually a liberal force when compared to the Simple Model. In the Simple Model there is a large cost to assigning a stem to a large number of suffixes, and this cost is paid for every stem in that paradigm. In the Paradigm Model, that cost is only paid once, in the creation of the Paradigm. The Paradigm Model actually allows for the creation of more paradigms than the Simple Model, allowing the prediction of potentially more suffixes.

Probability Maximization

3.1 Lattice Formulation

The input to the morphology induction system is a set of words, or lexicon, L. The hypothesis space for this system is defined in terms of the set of all possible stems in L, pStem, and the set of all possible suffixes in L, pSuff. The empty string is considered to be a possible suffix but not a possible stem. A hypothesis h is a function from the set of possible stems to sets of suffixes:

$$h : \text{pStem} \mapsto 2^{\text{pSuff}}$$

where $h(m)$ is interpreted as the set of suffixes that occur with stem m in the input. The set of words generated by a stem m under hypothesis h is simply the concatenation of m with all the suffixes in $h(m)$. For example, if $h(walk) = \{\epsilon, s, ing\}$ then the stem *walk* generates the words *walk, walks, walking*. If $h(m)$ is the empty set then m generates no words. In the searches described, the only hypotheses considered are those in which (a) each word of the input is divided into stem and suffix in exactly one way, (b) no other words are generated. Under such a hypothesis the sets of words associated with all possible stems form a partition of the input lexicon. Thus, the following invariant holds:

Invariant 1 For all hypotheses h,

$$\{\{mx \mid x \in h(m)\} \mid m \in \text{pStem}\}$$

is a partition of the input lexicon, L.

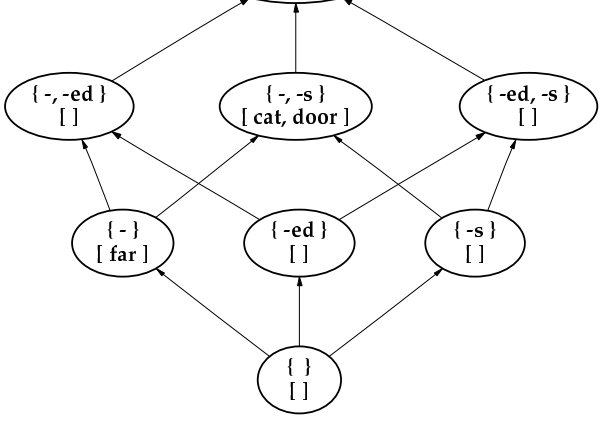

Figure 3.1: A Simple Lattice Representation

The nodes shown are those whose suffix sets are subsets of $\{\epsilon, s, ed\}$. The set of suffixes corresponding to a node n are designated by $s(n)$. In Figure 3.1, $s(n)$ is shown in the curly braces in each node n, and the stems of a paradigm are shown in square brackets. The node representing the suffix set $\{\epsilon, s\}$ has two stems, *cat* and *door*, and generates the words *cat*, *cats*, *door*, and *doors*. Those nodes with no suffixes or no stems, such as the node representing $\{s, ed\}$, generate no words. Note that the empty suffix (denoted by ϵ or by -) is treated like any other suffix: The suffix set containing only ϵ has size one and is distinct from the empty set.

Each node is also assigned a level $l(n)$ corresponding to the number of suffixes in n ($l(n) = |s(n)|$). The node repreeenting the empty set of suffixes is at level 0 in the lattice. If hypothesis h maps stem m onto node n at level i then m generates i words. As a direct consequence of Invariant 1, the following invariant holds:

Invariant 2 Let L be an input lexicon with possible stem set pStem. For all hypotheses h, the sum of the node levels of all stems is the number of input words:

$$\sum_{m \in \text{pStem}} l(h(m)) = |L|$$

the empty set of suffixes (the unique node at level 0). Hence, the initial hypothesis satisfies Invariants 1 and 2.

3.2.2 Search Operators

We now focus on search operators that move stems up and down the lattice by a single step. We first define operations that either promote or demote a single stem. That is, they move a stem up to a node whose level is one greater or down to a node whose level is one less than the node the stem occupies in the current hypothesis. In order to maintain Invariant 1, a stem m can be promoted from a node not containing suffix x to a node containing suffix x only if mx is one of the input words ($mx \in L$). If m is promoted to a node containing x some other stem must be demoted (Invariant 2). In particular, the stem that previously generated the word mx must be demoted so that it no longer does (Invariant 1). This stem is called the *Compensating Stem*, and the suffix with which it combined to generate mx is called the *Compensating Suffix*. The following formal definitions will prove useful.

Definition 1 Let L be an input lexicon. Let m and x be such that m is a stem in pStem, x a suffix in pSuff, and $mx \in L$. Let h be a hypothesis. Then CompM(m, x, h) and CompX(m, x, h) are the two unique strings such that:

1. CompM(m, x, h)CompX$(m, x, h) = mx$
2. CompX$(m, x, h) \in h($CompM$(m, x, h))$

The uniqueness of CompM(m, x, h) and CompX(m, x, h) follows from Invariant 1.

Definition 2 Let L be an input lexicon, m a stem in pStem, x a suffix in pSuff, and h a hypothesis. Then Prom(m_1, x, h) is the hypothesis that results from starting with h and promoting stem m_1 to a node that contains suffix x:

1. If $x \in h(m_1)$ or $m_1 x \notin L$, Prom$(m_1, x, h) = h$.

an adjacent node not containing x is always possible. The result is that m_1 no longer generates word m_1x, so some other stem must be moved up to a node in which it generates mx. There is always at least one stem that can be moved up so that it generates mx — namely, the stem that is equal to mx can be moved up one level from its current node to the node that also contains the empty suffix. (Its current node cannot contain the empty suffix because otherwise the word mx would have been generated twice).

Definition 3 Let L be an input lexicon, m_1 a stem in pStem, x a suffix in pSuff, and h a hypothesis. Then $\mathrm{Dem}(m_1, x, h)$ is the hypothesis such that:

1. If $x \notin h(m_1)$, $\mathrm{Dem}(m_1, x, h) = h$.
2. Otherwise, $\mathrm{Dem}(m_1, x, h)(m) =$

$$\begin{cases} h(m) - \{x\} & \text{if } m = m_1 \\ h(m) \cup \{\epsilon\} & \text{if } m = m_1 x \\ h(m) & \text{otherwise} \end{cases}$$

Remark. $\mathrm{Dem}(m_1, x, h)$ is equivalent to $\mathrm{Prom}(m_1x, \epsilon, h)$. Thus, Dem is a special case of Prom and all properties that hold for $\mathrm{Prom}(m_1, x, h)$ regardless of m_1, x, or h also hold for Dem.

The search algorithm used in the experiments reported below applies the promotion and demotion operators sequentially to all stems that map to a given node. It is convenient to define this operation by overloading the functions Prom and Dem as follows:

Definition 4 Let L be an input lexicon, n a node in the subset lattice of pSuff, x a suffix in pSuff, and h a hypothesis. Let $\{m_1, \ldots, m_k\}$ be the set of stems that map

Theorem 1 guarantees that the order of composition does not affect the result, and hence that these functions are well defined.

Theorem 1 Let x a suffix in pSuff, h a hypothesis, and $\{m_1, \ldots, m_k\}$ a set of stems in pStem. The hypothesis

$$\text{Prom}(m_1, x, \text{Prom}(m_2, x, \ldots \text{Prom}(m_k, x, h)))$$

is invariant under permutation of the subscripts.

3.2.3 Search Algorithm

The hill climbing search alternates between a promotion phase and demotion phase until neither phase can improve the score further.

```
Search
1:  h = InitialHypothesis
2:  While (Probability increases)
3:    PromotionPhase
4:    DemotionPhase
```

The promotion phase loops through all possible suffixes. For each suffix, it loops through all nodes that contain at least one stem and one suffix (the node for the empty set of suffixes is not included). For each suffix/node combination, it evaluates the hypothesis that would result from applying the promotion operator to that node and that suffix. If the probability of the hypothesis resulting from promotion is greater than the probability of the current hypothesis then the promotion is carried out. For each suffix x, the loop through all nodes is restarted whenever a promotion is carried out.

```
PromotionPhase
1:  For each x ∈ pSuff
```

The order in which `PromotionPhase` and `DemotionPhase` iterate through possible suffixes is determined by the relative frequency of the suffix, as a string, divided by the relative frequencies of its component letters. For example if c is the total number of characters in the lexicon, the rank of *-ing* would be $\frac{f(\text{-ing})/(c-2)}{f(\text{i})f(\text{n})f(\text{g})/c^3}$. This formula reflects the degree to which the observed relative frequency of a suffix exceeds what would be expected under a null model in which letters are chosen independently. Suffixes are processed in order from highest rank to lowest in order to give priority to those that are most likely to be productive.

3.2.5 Search Limitations

There are severe limitations to the search as described above, which may not be readily noticeable. The most important of them is that each stem considered by the search algorithm must itself be a word in the input lexicon.

Each word is initially generated by the node containing only the suffix ϵ. For a word to be generated by another node, its stem must be promoted from this node. All stems in this node must themselves be words, however. Since all stems can only be promoted from the node containing no suffixes into the node representing the ϵ, there is no way for a stem that is not itself a word in the input lexicon to enter a node containing suffixes. All stems therefore must also be words.

This limitation is not a significant impairment in English, where almost all stems are words, but it is devastating in other languages, including those in the Romance and Slavic language families. French for example, has the verb *parler*, whose inflected forms include *parle*, *parles*, *parlons*, and *parla*. The ideal stem to identify would be *parl*, as it is the longest common substring for all of the inflected forms, however *parl* is not a word in French and thus the hill-climbing search on its own could not identify it and would miss many of the inflected forms of *parler*.

search is a valid hypothesis, and can be used without the hill climbing search detailed above, though a combination of the two will yield a more probable hypothesis than the directed search would on its own. The directed search does not directly attempt to find the most probable hypothesis consistent with the input, but finds a highly probable and consistent hypothesis.

The directed search algorithm attempts to find highly productive nodes in the lattice and maximally fill them with stems. The set of highly productive nodes and the stems in them are then pruned to produce a valid hypothesis consistent with the lattice invariants.

The directed search is accomplished in two steps. First sub-hypotheses, each of which is a hypothesis about a subset of the lexicon, are examined and ranked. The sub-hypotheses are initially each nodes in the lattice. The N best sub-hypotheses are then incrementally combined until a single sub-hypothesis remains. The remainder of the input lexicon is added to this sub-hypothesis at which point it becomes the final hypothesis.

3.3.1 Ranking Sub-Hypotheses

The set of possible suffixes is defined to be the set of terminal substrings, including the empty string ϵ, of the words in L. Each subset of the possible suffixes has a corresponding sub-hypothesis. The sub-hypothesis, h, corresponding to a set of suffixes SUFF_h, has the set of stems STEMS_h. For each stem m and suffix x, in h, the word $m + x$ must be a word in the input lexicon. STEM_h is the maximal sized set of stems that meets this requirement. The sub-hypothesis, h, is thus the hypothesis over the set of words formed by all pairings of the stems in STEM_h and the suffixes in SUFF_h with the corresponding morphological breaks. One can think of each sub-hypothesis as initially corresponding to a maximally filled paradigm. Only sub-hypotheses which have at least two stems and two suffixes are considered.

3.3.2 Combining Sub-Hypotheses

The highest N scoring sub-hypotheses are incrementally combined in order to create a hypothesis over the complete set of input words. The selection of N should not vary from language to language and is simply a way of limiting the computational complexity of the algorithm. Changing the value of N does not dramatically alter the results of the algorithm. We let N be 100 in the experiments reported here.

Let S be the set of the N highest scoring sub-hypotheses. We remove from S the sub-hypothesis, s', which has the highest score. The words in s' are now added to each of the remaining sub-hypotheses in S, and their counter hypotheses. Every sub-hypothesis, s, and its counter, $\bar{s}$, in S are modified such that they now contain all the words from s' with the morphological breaks those words had in s'. If a word was already in s and $\bar{s}$ and it is also in s' then it now has the morphological break from s', overriding whatever break was previously attributed to the word.

All of the sub-hypotheses now need to be rescored, as the words in them will likely have changed. If, after rescoring, none of the sub-hypotheses have scores greater than one, then we use s' as our final hypothesis. Otherwise, we repeat the process of selecting s' and adding it in. We continue doing this until all sub-hypotheses have scores of one or less or there are no sub-hypotheses left.

The final sub-hypothesis, s', is now converted into a full hypothesis over all the words. All words in L, that are not in s' are added to s' with ϵ as their suffix. This results in a hypothesis over all the words in L.

Empirical Results

This chapter describes experiments evaluating the performance of the algorithm described in Chapter 2 and Chapter 3 conducted on English and Polish data. Several variants of the algorithm are compared and the performance of the algorithm is compared with that of Goldsmith's Linguistica algorithm[9].

4.1 Input Lexicons

Various sized word lists were extracted from English and Polish corpora. The English word lists were extracted from set A of the Hansard corpus, which is a parallel English and French corpus of the proceedings of the Canadian Parliament. No standard corpus of Polish could be found, so one was developed from online documents consisting of old Polish stories. The sources for the Polish corpus were somewhat older texts and thus our results correspond to a slightly antiquated form of the language, though not significantly divergent from modern Polish. The English in the Hansard corpus corresponds to British English and not American English.

The most frequent words in both corpora were extracted, excluding words with non-alphabetic characters, to build the input lexicons. The 100 most common words in each corpus were also excluded, as these words tend to be function words and do not tend to have regular inflection. Testing does not reveal significantly different results when the one hundred most common words are included. Lexicons consisting of the 500, 1000, 2000, 4000, 8000 and 16,000 most common words were then constructed and used as input to the morphology learning algorithms. The Polish data set did not include a lexicon with 16,000 words as the corpus was too small and did not contain a sufficient number of word types.

break in the word, "location" should be placed. It seems that the stem "locate" is combined with the suffix "tion", but in terms of simple concatenation it is unclear if the break should be placed before or after the "t". When "locate" is combined with the suffix "s", simple concatenation seems to work fine, though a different stem is found from "location" and the suffix "es" could be argued for. One solution is to develop an evaluation technique which incorporates the adjustment or spelling change rules, such as the one that deletes the "e" in "locate" when combining with "tion".

None of the systems being evaluated attempt to learn adjustment rules, and thus it would be difficult to analyze them using such a measure. In an attempt to solve this problem we have developed two new measures of performance, neither of which specifies the exact morphological split of a word. The accuracy of stem and suffix identification are evaluated separately, but when considered together they give a robust measure of the overall performance. The accuracy of the stems predicted is analyzed by examining whether pairs of words are related by having the same immediate stem. We measure suffix identification by comparing against groups of suffixes which have the same underlying form but differ in surface form due to adjustment rules.

4.2.2 Stem Relation

Two words are related if they share the same immediate stem. For example the words "building", "build", and "builds" are related since they all have "build" as a stem, just as "building" and "buildings" are related as they both have "building" as a stem. The two words, "buildings" and "build" are not directly related since the former has "building" as a stem, while "build" is its own stem. Irregular forms of words are also considered to be related even though such relations would be very difficult to detect with a simple concatenation model.

We say that a morphological analyzer predicts two words as being related if it attributes the same stem to both words, regardless of what that stem actually is.

Lexicon Size	English	Polish
500	99	348
1,000	321	891
2,000	1,012	2,062
4,000	2,749	4,352
8,000	6,762	9,407
16,000	15,093	-

Table 4.1 shows the correct number of stems relations at each lexicon size in both English and Polish. Because Polish is morphological richer than English and uses more inflected forms, the number of relations in Polish is consistently higher than in English for same number of words.

4.2.3 Suffix Identification

Suffix accuracy is measured by gathering all of the surface forms of all of the suffixes from a given input and grouping those with the same underlying form together. We only grouped together those suffixes which seemed to be simple adjustment-rule variations of each other. Only suffix spelling changes were included. For example, "s", "es", and "ies" were grouped together, as were several variations of "tion", such as "ation", "sion", and "ion". The suffix "ing" was not grouped with any other suffixes, despite the presence of words such as "running", where the "n" was duplicated, as this was deemed to be a stem spelling change.

We only scored against those suffix groups whose suffixes occurred in at least two of the stem relations for the same input. If an analyzer predicts any suffix in a suffix group, it is said to predict that group. Predicting multiple suffixes in a group yields no additional score increase, though it will likely improve the stem relation score. Each incorrect suffix predicted counts as a false suffix group. The precision for suffix identification is the number of correct groups predicted divided by the total

be able to score high in one measure but at the expense of the other measure.

Table 4.2: Correct Number of Suffixes

Lexicon Size	English	Polish
500	10 (31)	19 (29)
1,000	14 (40)	34 (46)
2,000	24 (65)	51 (70)
4,000	33 (85)	63 (86)
8,000	39 (92)	86 (109)
16,000	45 (99)	-

Table 4.2 shows the correct number of suffixes at each lexicon size in both English and Polish. The first number is the number of suffix groups and the number in parentheses is the total number of surface forms of suffixes. Again the number of suffixes in Polish is much higher than in English, but the ratio of suffix surface forms to suffix groups is much lower. This is because there are fewer suffix spelling change rules in Polish than in English, so there are fewer surface forms for each suffix. There are in fact a roughly equal number of surface forms in both languages.

4.2.4 Example Evaluation

It might be of some assistance to consider an example evaluation of a hypothesis in English. Consider the 21 words in Table 4.3, with the morphological breaks shown. These words and the morphological splits shown are not from actual experiments and are purely for explanation of the evaluation method.

There are obviously several incorrect analyses of words in Table 4.3. The preferred analysis for *justice* would be *justice* + ϵ, and the analysis for *justices* should be *justice* + *s*. Similarly for *office* and *offices*. In addition the hypothesis fails to find that *buildings* should be *building* + *s*. The analysis of *operate* as *operat* + *e* is

ponder + ing	quick + ϵ	quick + ly

debatable, as there should be no morphological split in *operate*, but such an analysis identifies that *operate* and *operation* have the same stem.

Table 4.4: Evaluation of Example Hypothesis

	TP	FP	FN	Precision	Recall	Fscore
Stem Relation	12	4	1	0.75	0.923	0.828
Suffix Identification	5	3	0	0.625	1.0	0.769

Table 4.4 shows the results of evaluating the example hypothesis in Table 4.3. TP is the number of true positives, which is the number of relations or suffixes correctly identified. FP is the number of false positives and FN is the number of false negatives. Precision, a measure of the accuracy of the predictions is defined as:

$$\text{Precision} := \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Recall, a measure of how many of the correct predictions were found, is defined as:

$$\text{Recall} := \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Fscore is a measure that combines precision and recall in an unbiased manner, which favors an equal precision and recall. The fscore allows a single value by which to evaluate the performance, in either stem relation or suffix identification, and is defined as:

$$\text{Fscore} := \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

There are nine predicted suffixes, ϵ, *-ing*, *-s*, *-ly*, *-e*, *-tion*, *-ice*, *-ices*, and *ion*, of which three are not really suffixes, *-e*, *-ice* and *-ices*. Two of the suffixes, *-ion* and *-tion*, belong to the same suffix group, that is they are really the same suffix but have different surface forms. There are no suffixes in the data that were not identified in the hypothesis, so there are 5 correct suffix groups. The number of correct suffixes found is 5 and the number of incorrect suffixes identified is 3. The hypothesis thus has a suffix precision of 0.625 and a suffix recall of 1.0, giving an fscore of 0.769.

No hypothesis for this set of words would ever result in a perfect score for stem relations, due to *build*, *building*, and *buildings*. To get a perfect stem relation score, *build* and *building* would have to have the same stem, *building* and *buildings* would have to have the same stem, and *build* and *buildings* would have to have different stems. However if any two of those relationships hold then the third must be false, since no ambiguity is allowed.

Additionally if *operate* and *operating* are correctly found to have the same stem then one of the words must have an incorrect suffix, as *-ing* has only one correct surface form and *-e* is not a correct suffix. Only a system that could learn spelling change rules would be succeed at a perfect score with these two words. A balance between the two evaluation methods must be reached for optimal performance.

4.3 Experimental Comparisons

4.3.1 Comparison of Model Variants

Comparisons of results using the Naive Model from section 2.1, the Simple Model from section 2.2, and the Paradigm Model from section 2.3. The search through the probability space was accomplished by first finding a good hypothesis using the directed initial hypothesis algorithm described in section 3.3, and then maximizing the probability with the hill climbing search from section 3.2.

Figure 4.1: Number of Suffixes Predicted by Model Variants

Figure 4.1 shows the number of suffixes predicted by the different model variants in both English and Polish. This is the number of surface forms of suffixes, not the number of suffix groups predicted. The Naive Model predicts far more suffixes at larger corpus sizes than any of the other models, due mostly to the small constraints on the suffix set. The Simple Model is the most conservative due to the high cost of pairing a suffix with a stem. The difference between the two versions of the Paradigm Model is fairly small though it appears that the uniform prior distribution causes the model to be slightly more liberal in predicting new suffixes.

Stem Relation Accuracy

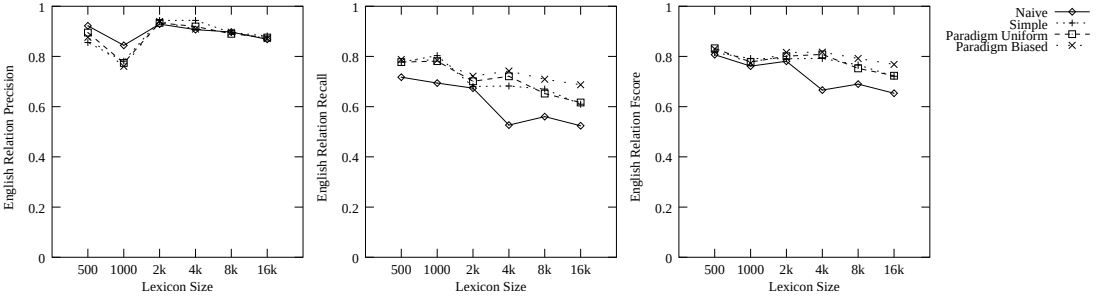

Figure 4.2: English Stem Relation Accuracy across Model Variants

Figure 4.2 and figure 4.3 show the stem relation accuracy for the different models in English and Polish, respectively. The precision for all the models are relatively equal, making it impossible to judge which is better. The recall is slightly more informative, as it shows the consistently worse performance of the Naive Model. The other models are still too similar to differentiate. The performance across both languages is relatively consistent.

Figure 4.3: Polish Stem Relation Accuracy across Model Variants

Suffix Identification Accuracy

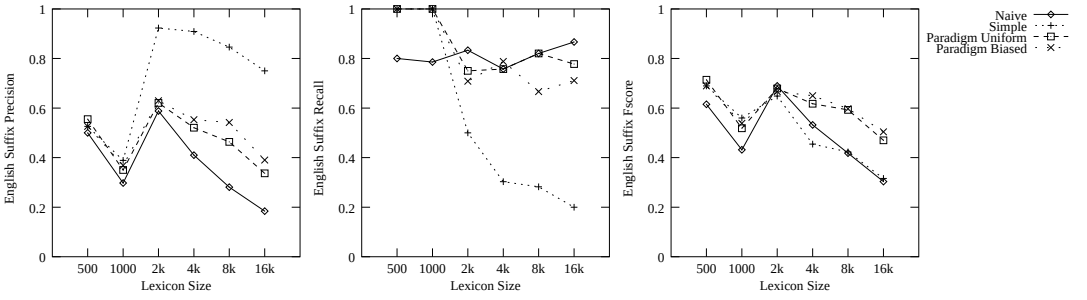

Figure 4.4: English Suffix Accuracy across Model Variants

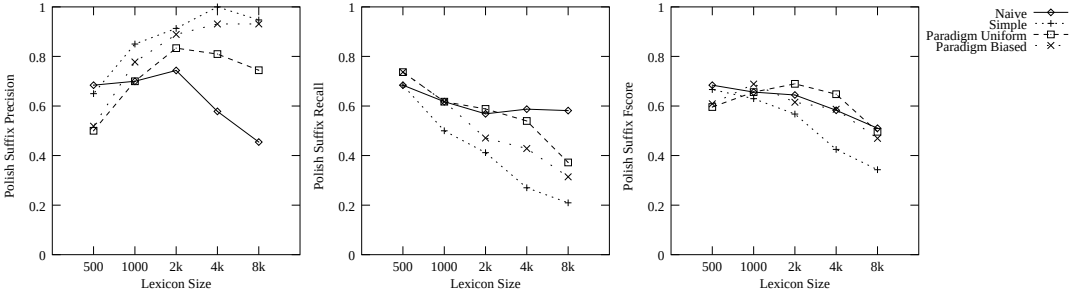

Figure 4.5: Polish Suffix Accuracy across Model Variants

Figure 4.4 and figure 4.5 show the suffix identification accuracy for the model variants in English and Polish, respectively. The Simple Model has very high precision in both English and Polish at the expense of its recall. This is reasonable as the Simple Model predicts a very small set of suffixes, though those suffixes that it does predict are very accurate. Both versions of the Paradigm Model have very similar Fscores. The biased paradigm prior causes a slightly higher precision, balanced by a reduction in recall. Since our goal is to generate a conservative model that has higher precision, it is fair to say that the biased prior slightly outperforms the uniform prior.

evidence that the suffix occurs and more words are required for the models to favor the introduction of new suffixes. At much larger lexicon sizes the number of suffixes detected in Polish should exceed the number detected in English. This phenomenon can also be seen in the lower recall of the models in Polish.

The Paradigm Model with a biased paradigm prior has the best performance and balance between both stem relation identification and suffix identification. The Paradigm Model with a uniform paradigm prior has only slightly worse performance, indicating that the prior used for that distribution only causes slight differences in performance with the searches used. Therefore the Paradigm Model with the biased paradigm prior has be used for the future experiments in this chapter.

4.3.2 Comparison of Search Variants

This section will compare the search starting from the original initial hypothesis from section 3.2.1 to the directed initial hypothesis without search, as well as the combination of the two. The Paradigm Model with a biased paradigm prior was used with all of the experiments in this section.

Number of Suffixes Predicted

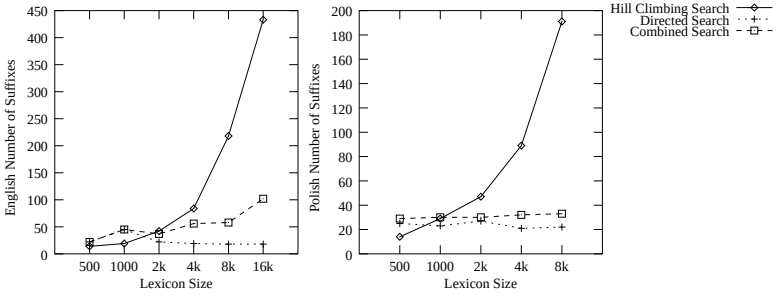

Figure 4.6: Number of Suffixes Predicted by Search Variants

growth rate of the number of correct suffixes.

The results in Polish for the Hill Climbing search are especially interesting and unusual. Both the Simple Model and the Paradigm Model with a uniform paradigm prior perform quite differently when applied to Polish using only the Hill Climbing search, in that neither of those models identifies any suffixes, except ϵ. Using only the Hill Climbing search those models completely fail to analyze the Polish input lexicons. Using the Directed and Combined Searches they succeed in producing an analysis as previously shown in Figure 4.1.

Stem Relation Accuracy

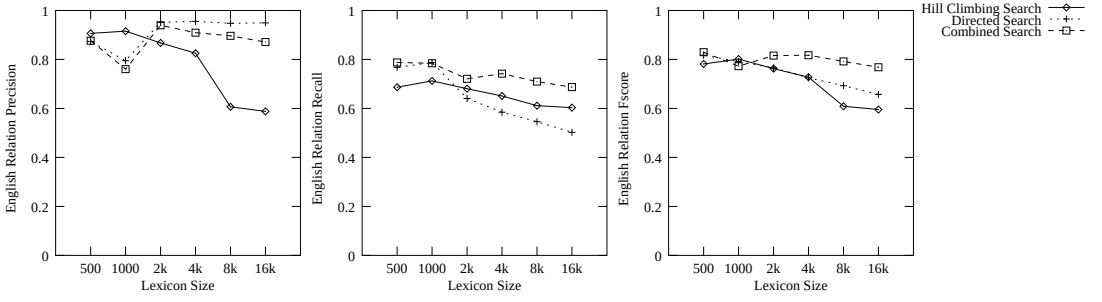

Figure 4.7: English Stem Relation Accuracy across Search Variants

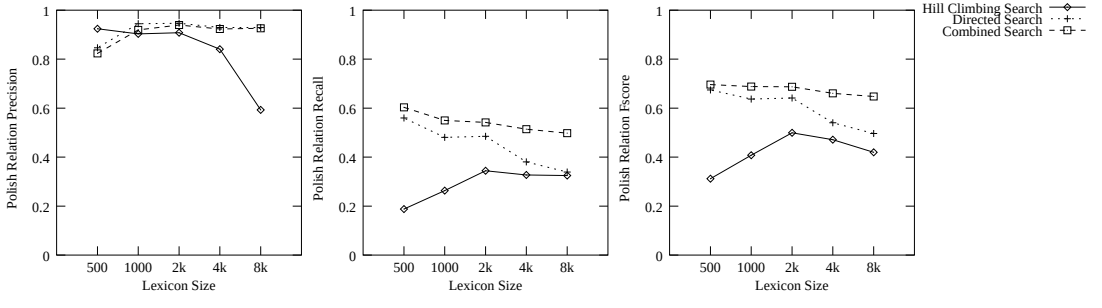

Figure 4.8: Polish Stem Relation Accuracy across Search Variants

misanalyzing many words as the lexicon sizes grow larger indicating difficulties scaling up to larger datasets.

Suffix Identification Accuracy

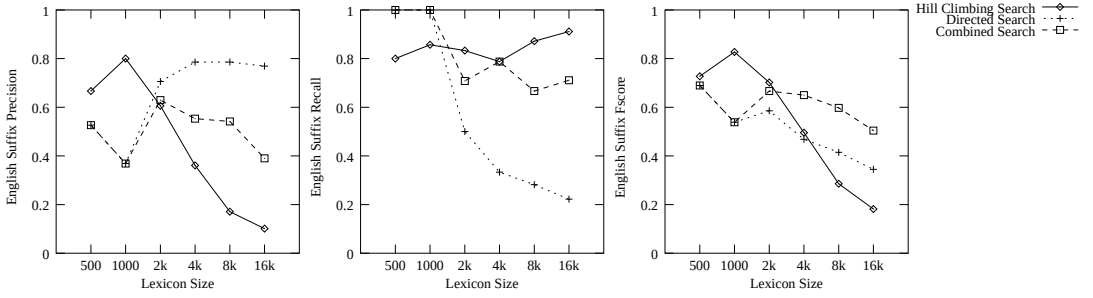

Figure 4.9: English Suffix Accuracy across Search Variants

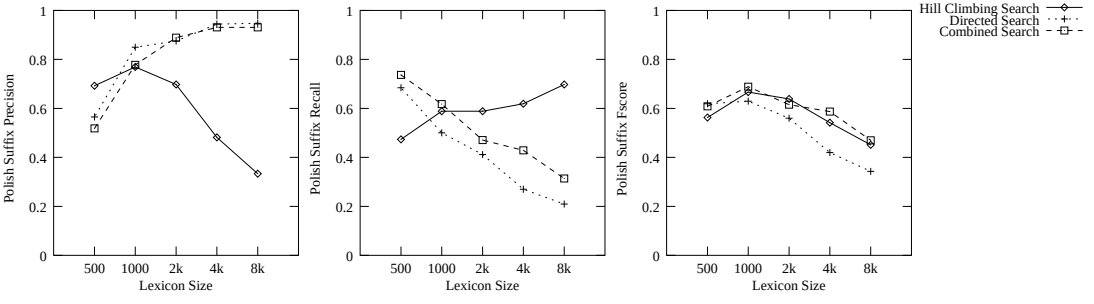

Figure 4.10: Polish Suffix Accuracy across Search Variants

Figure 4.9 and figure 4.10 show the suffix identification accuracy for the search variants in English and Polish, respectively. The Directed Search shows an excellent increase in precision as the datasets increase in size, indicating that while it may not be predicating many more suffixes as the lexicon size increases, those suffixes that it does predict are more and more accurate. The lack of new suffixes predicted by the Directed Search is evident in the rapidly descending recall score. The combination of

the addition of the Hill Climbing Search to find decent hypotheses. It is even slightly more conservative than the combination of the two. The performance of the combined system, as shown by the fscores, does seem to outweigh the slight loss of conservation. The Directed Search appears to be too conservative, a problem solved by performing the Hill Climbing Search from hypothesis generated by the Directed Search. As a result the combined search technique will used for the remainder of the experiments in this chapter.

4.3.3 Comparison with Linguistica

A version of Goldsmith's Linguistica system available on the world wide web[1] was used for these experiments. Word-list corpus mode and the method A suffix detection were used, while all other parameters were left at their default values. The use of the word-list corpus mode may have slightly hampered the performance of the system, as it is an EM algorithm that uses the frequency of word types in assigning initial weights. Due to software difficulties Linguistica was unable to run on the 500, 1000, and 2000 word English Lexicons. The system ran without difficulties on the larger English lexicons and on all of the Polish input lexicons.

Comparisons of Linguistica to the combined search with the Paradigm Model with a biased paradigm prior are shown below. For purposes of simplicity the combined search with the Paradigm Model with a biased prior shall be referred to as the sMorph system.

Number of Suffixes Predicted

Figure 4.11 shows the number of suffixes predicted by Linguistica and sMorph. Linguistica clearly identifies far more suffixes than the sMorph system. The sMorph system is very consistent in the number of suffixes it predicts, with slight upward

[1] available at http://humanities.uchicago.edu/faculty/goldsmith

Figure 4.11: Number of Suffixes Predicted in Linguistica Comparison

trends. As Linguistica is predicting over 700 suffixes when exposed to 16,000 word types in English it is clearly over predicting. Surprisingly Linguistica predicts far fewer suffixes in Polish than in English, for the same reasons as the sMorph system, as described above in section 4.3.2.

Stem Relation Accuracy

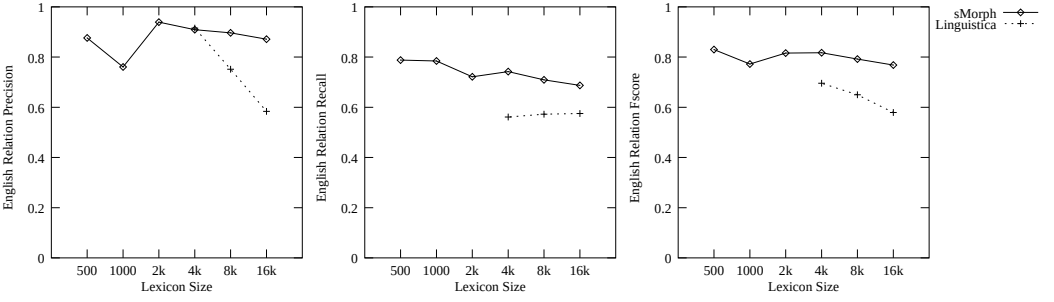

Figure 4.12: English Stem Relation Accuracy in Linguistica Comparison

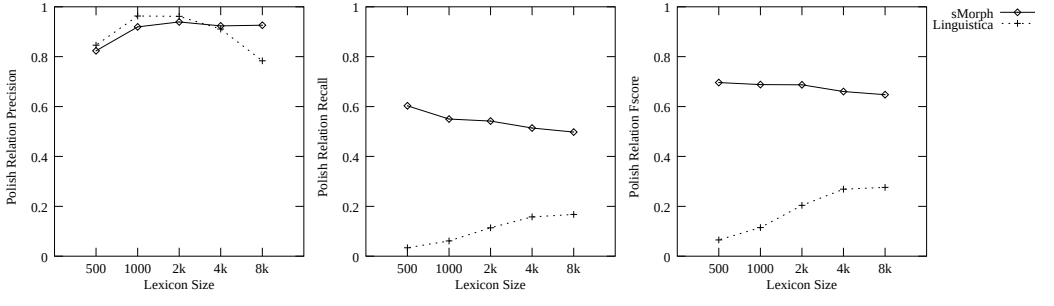

Figure 4.13: Polish Stem Relation Accuracy in Linguistica Comparison

Figure 4.12 and figure 4.13 show the differences in stem relation accuracy between sMorph and Linguistica in both English and Polish. The relation precision

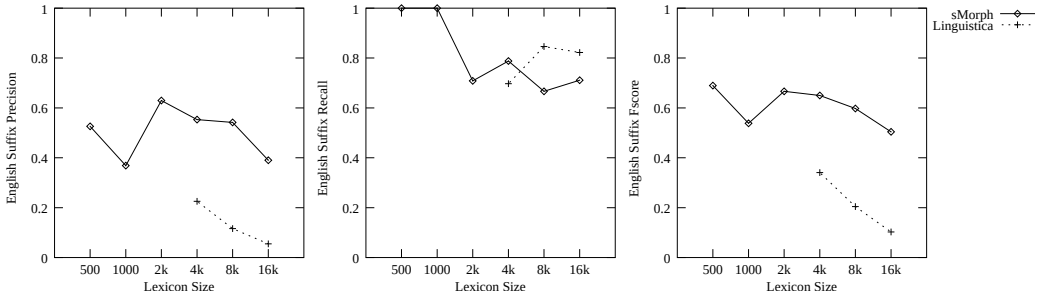

Figure 4.14: English Suffix Accuracy in Linguistica Comparison

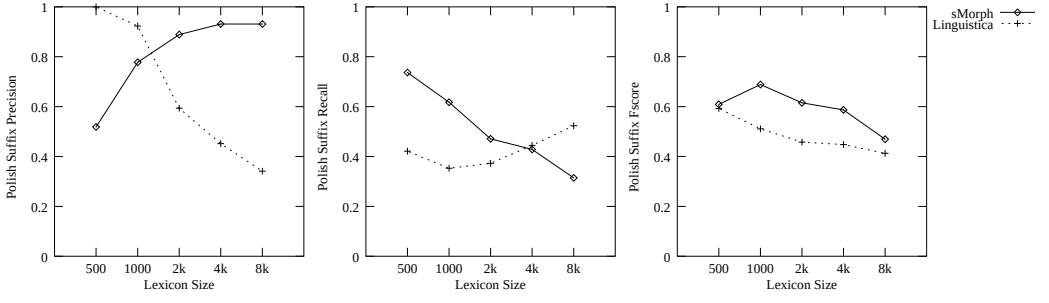

Figure 4.15: Polish Suffix Accuracy in Linguistica Comparison

Figure 4.14 and figure 4.15 show the suffix identification accuracy between Linguistica and sMorph in various sized lexicons of English and Polish. These results are especially interesting as the number of suffixes predicted by the two systems is so divergent. The exceptionally low suffix identification precision of the Linguistica system in English shows that very few of the large number of suffixes predicted by that system are correct. Due to the large number of suffixes predicted by Linguistica it does retain a high recall, though the balance of precision and recall is slightly worse than the sMorph system. Since we are aiming for high precision for this task the loss of precision at the benefit of recall is not desired.

Conclusions

I have presented and compared several generative probability models, which are used to evaluate hypotheses, as well a non-probabilistic naive scoring system. The most sophisticated of these systems, the Paradigm Model with a biased paradigm prior, consistently has the best performance when applied to both English and Polish datasets. This model is used to score hypotheses by a search which looks for the highest scoring, or most probable, hypothesis. The combination of a directed and hill-climbing search is shown to slightly outperform the directed search on its own in both English and Polish. This system, sMorph, was then compared to Goldsmith's Linguistica system, which is another unsupervised morphology learning algorithm. Linguistica over predicts the number of suffixes causing it make far more mistakes, while sMorph makes fewer, but more accurate, predictions.

The sMorph system succeeds in its goal of conservative morphology learning, though there is much room for increased accuracy. The hypotheses generated appear to be good starting points for sophisticated systems, which could learn part of speech tags, adjustment rules and more advanced morphology. Further testing is required to show the language independence of the system, though the main difficulty here lies in evaluation. Morphological analyses of languages are difficult to obtain, especially for the rarer languages upon which this algorithm is designed to be used, and it is difficult to test the system without the ability to quantitatively evaluate the results.

The primary power of the system is its exploitation of the notion of a paradigm. By both modeling paradigms and searching specifically for paradigms in both searches, the system hones in on the decisive characteristic of natural language morphology. Any system that doesn't capture the notion of a paradigm cannot succeed at learning morphology. The modular nature of the system is also highly useful, as one could use

and multiple suffixes, such finding that *buildings* should be *build + ing + s*. The current system is also unable to handle the ever popular word *antidisestablishmentarianism*, which has many layers of prefixes and suffixes. The current models and searches would probably need some significant improvement, though not enough to destroy their fundamental character. The current system could be trivially altered to detect prefixes by simply reversing all of the words in the lexicon on input and output, such that it would treat all prefixes as suffixes. This does not aid in the task of detecting prefixes and suffixes at the same time however.

The learning of adjustment or spelling change rules would be an invaluable improvement to the current model. These rules would need to be learned for both spelling change rules in the suffixes, such as the rule in English stating "-s" $\rightarrow$ "-es" after a stem ending in "h". A minimally supervised system for learning these rules has already been developed by Yarowsky and Wicentowski[18]. The system takes a preliminary morphological hypothesis, part of speech tags and token counts for the input words, and learns a set of adjustment rules. Integrating such an algorithm into the current sMorph system could allow for significant improvement in performance, especially in English where suffix spelling changes are frequent.

A minimally supervised version of the sMorph algorithm seems very feasible. Rather than presenting only unlabelled data to the algorithm a few examples of words with morphological splits could also be given to aid in the initial searching. The expense of annotating a small set of examples by hand is much more practical than annotating the entire lexicon of a language, and would help limit the search space. Such a modification to the system could be accomplished by including the supervised training data with the unsupervised data, and not allowing the search routines to modify the morphological breaks of the annotated data. A similar approach was used by Brent and Tao [5] to make their unsupervised word segmentation algorithm, minimally supervised. The incorporation of training data might not significantly improve performance, since the current algorithm does not have difficulty identifying

modeling for speech recognition tasks[1]. Schone and Jurafsky use singular value decomposition (SVD) to calculate the semantic relatedness of words from the cooccurance data, but such a method does not lend itself to incorporation into a probabilistic model. Probabilistic Latent Semantic Analysis[10] is a variation on LSA that attempts to model the data using a multinomial distribution with maximum likelihood estimates. This probabilistic model has shown success when integrated into language modeling[15], and attempts have been made to integrate a similar model into the morphology system. I am confident that the incorporation of semantic information could succeed and provide greater accuracy.

Perhaps the most useful improvement to this system would be the development of an incremental directed search. The system currently needs to be applied to the entire set of input words, which is computationally very expensive if the input size is extremely large, making scaling up the system difficult. The system would benefit if it analyzed only a subset of the lexicon, such as the 500 most frequent words, and then added in more and more words incrementally so that the search space was more limited. The most fascinating part of the directed search is that it in effect targets subsets of the lexicon and analyzes them, making it ideal for working on subsets of the data. An alternative approach would be to break the lexicon into several pieces, use the directed search on each part, and then merge the resulting sub-hypotheses together.

[1] Jerome Bellegarda. Exploiting latent semantic information in statistical language modeling. In *Proceedings of the IEEE*, volume 88, pages 1279–1296. IEEE, 2000.

[2] Michael R. Brent. Minimal generative models: A middle ground between neurons and triggers. In *Proceedings of the 15th Annual Conference of the Cognitive Science Society*, pages 28–36, Hillsdale, NJ, 1993. Erlbaum.

[3] Michael R. Brent. An efficient, probabilistically sound algorithm for segmentation and word discovery. *Machine Learning*, 34:71–106, 1999.

[4] Michael R. Brent, Sreerama K. Murthy, and Andrew Lundberg. Discovering morphemic suffixes: A case study in minimum description length induction. In *Proceedings of the Fifth International Workshop on Artificial Intelligence and Statistics*, Ft. Laudersdale, FL, 1995.

[5] Michael R. Brent and Xiaopeng Tao. Chinese text segmentation with mbdp-1: Making the most of training corpora. In *Proceedings of the 39th Annual Meeting of the Association of Computational Linguistics*, pages 82–89. ACL, 2001.

[6] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[7] Hervé Déjean. Morphemes as necessary concepts for structures: Discovery from untagged corpora. http://www.info.unicaen.fr/ DeJean/travail/articles/pg11.htm, 1998.

[8] Éric. Gaussier. Unsupervised learning of derivational morphology from inflectional lexicons. In *ACL '99 Workshop Proceedings: Unsupervised Learning in Natural Language Processing*. ACL, 1999.

1997.

[12] J. Ross Quinlan and Ronald L. Rivest. Inferring decision trees using the minimum description length principle. *Information and Computation*, 80:227–248, 1989.

[13] Jorma Rissanen. *Stochastic Complexity in Statistical Inquiry*. World Scientific Publishing, Singapore, 1989.

[14] Patrick Schone and Daniel Jurafsky. Knowledge-free induction of morphology using latent semantic analysis. In *Proceedings of the Conference on Computational Natural Language Learning*. Conference on Computational Natural Language Learning, 2000.

[15] Ronald Rosenfield Shaojun Wang and Yunxin Zhao. Latent maximum entropy principle for statistical language modeling. In *Proceedings of the IEEEAutomatic Speech Recognition and Understanding Workshop*. IEEE ASRU, 2001.

[16] Antal Van den Bosch and Walter Daelemans. Memory-based morphological analysis. In *Proc. of the 37th Annual Meeting of the ACL*. ACL, 1999.

[17] R. A. Wagner and M. J. Fischer. The string-to-string correction problem. *Journal of the Association for Computing Machinery*, 21:168–173, 1974.

[18] David Yarowsky and Richard Wicentowski. Minimally supervised morphological analysis by multimodal alignment. In *Proceedings of ACL-2000*, pages 207–216. ACL, 2000.

Degrees	B.S., Computer Science, Washington University, 2000
Professional Societies	Member, Association for Computational Linguistics
Publications	Snover, M. G. and Brent, M. R. (2001). A Bayesian Model for Morpheme and Paradigm Identification. In *Proceedings of the Association for Computational Linguistics 2001*, pages 482-490.

May 2002