# A Bayesian Model For Morpheme and Paradigm Identification

**Matthew G. Snover** and **Michael R. Brent**
Department of Computer Science
Campus Box 1045
Washington University
St. Louis, MO 63130-4899
{ms9, brent}@cs.wustl.edu

## Abstract

This paper describes a system for unsupervised learning of morphological affixes from texts or word lists. The system is composed of a generative probability model and a search algorithm. Experiments on the Wall Street Journal and the Hansard Corpus (French and English) demonstrate the effectiveness of this approach. The results suggest that more integrated systems for learning both affixes and morphographemic adjustment rules may be feasible. In addition, several definitions and a theorem are developed so that our search algorithm can be formalized in terms of the lattice formed by subsets of suffixes under inclusion. This formalism is expected to be useful for investigating alternative search strategies over the same morphological hypothesis space.

## 1 Introduction

Systems for morphological analysis were one of the early successes in computational linguistics (Karttunen and Wittenburg, 1983; Karttunen, 1983). Morphological analyzers are now a key component of most natural language applications and they continue to be a focus of research interest (Sproat, 1992). These systems are knowledge-intensive, requiring a stem lexicon, a list of affixes, morphotactic rules specifying the types of words to which each affix can apply, and spelling adjustment rules such as the English rule that inserts an *e* between a stem-final sybillant and the suffix *-s* in words like *churches*. Adaptation of a morphological analysis system to a new language requires considerable expertise with both the language and the morphological analyzer.

In the 1990's, the focus of research in syntactic parsing shifted from brittle, knowledge-intensive systems to systems based on probabilistic inference and unsupervised learning. Perhaps surprisingly, given its established place in computational linguistics, computational morphology participated in this shift in only a very limited way. Brent (1993) and Brent et al. (1995) described systems for suffix-discovery based on Minimum Description Length (MDL), an approach that is closely related to Bayesian probabilistic modeling. One system attempted to find the set of suffixes in the language using only the spellings of the words; another used the syntactic categories from a tagged corpus as well. Goldsmith (2000) describes another MDL system based on the spellings of words and the set of suffixes that appears with each stem, which Goldsmith calls the *signature* of the word. Gaussier (1999) reports on an explicitly probabilistic system that is based primarily on spellings, while Déjean (1998) describes a heuristic approach. Unlike the methods described above, the memory-based algorithm of van den Bosch and Daelemans (1999) requires supervised training. However, this system has the advantage that its output is a full morphological analysis, with syntactic category labels, not merely a splitting of the word into stem and suffix. Schone and Jurafsky (2000) take a completely different approach based on latent semantic analysis; words that lie near one another in the latent semantic space are treated as more likely to be morphologically related. Yarowsky and Wicentowski (2000) have developed a statistical system that takes a set of regular suffixes as input and learns spelling change rules and irregular morphology. This system uses four different statistical models for identifying words that are morphologically related. One of these models is similar to the latent semantic analysis method of Schone and Jurafsky (2000).

In this paper, we describe an unsupervised learning system that is designed to identify a small number of highly productive suffixes with a very low false positive rate. The system is intended to work cross-linguistically without the need to adjust any free parameters. In particular, we were concerned that the number of suffixes returned by many systems grows as the number of distinct words in the input grows. Thus, input size may serve as a kind of hidden free parameter — good performance may depend on picking exactly the right input size (Brent et al., 1995). Thus, we set out to develop a system that yields very few false positives over a wide range of input sizes in multiple languages. Experiments on French and English suggest that we have made substantial progress toward this goal.

We emphasize a low false positive rate because the suffix-discovery system reported here is intended as a first step toward a system for learning a complete morphological analysis, including irregular morphology, morphosyntax, and spelling adjustment rules. For example, the suffixes discovered by the system described below could be used as one of the inputs to the Yarowsky and Wicentowski (2000) system. We believe that the best strategy for learning a complete morphological analysis is to start with a small set of highly productive suffixes containing very few false positives. For example, a spelling adjustment rule like deletion of stem-final *e* before suffixes beginning in vowels (*rise+ing→rising*, *rise+er→riser*) can be learned on the basis of any suffix beginning in a vowel. Once the rule is learned, it can help identify other suffixes beginning in vowels. Thus, it may be more effective to be conservative at first and count on step-by-step learning to reveal linguistic regularities that are missed early on.

The system described in this paper is based on an explicit probability model and a search algorithm. The model assigns probability to any possible hypothesis about the correct morphological analysis of the input. A hypothesis consists of a division of every word in the input into a stem and suffix, where the suffix may be the empty string. The model described below assigns probability to any given hypothesis based on (1) the hypothesized number of stems and suffixes, (2) the lengths of the hypothesized stems and suffixes, (3) the number of suffixes hypothesized to occur with each stem, and (4) the frequency of each letter in the hypothesized stem and suffix sets. The search algorithm explores a series of hypotheses, replacing its current hypothesis whenever a more probable one is found.

The remainder of this paper is organized as follows. The next section describes the generative probability model. The third section describes the search algorithm. The fourth section describes an experiment on words extracted from the Wall Street Journal and the Hansard French and English corpora. The final section discusses the results and their implications.

## 2 Probability Model

This section introduces a language-independant probability distribution on all possible hypotheses, where a hypothesis is a division of every word in the input into a stem and a (possibly empty) suffix. The distribution is based on a five-step model for the generation of hypotheses. The steps are presented below, along with their probability distributions. What we describe in this section is a mathematical model, not an algorithm that is intended to be run.

1. Choose the number of stems, $M$, according to the distribution:

$$\Pr(M) = \frac{6}{\pi^2}\left(\frac{1}{M}\right)^2 \qquad (1)$$

The $6/\pi^2$ term normalizes the inverse-squared distribution on the positive integers. Choose the number of suffixes, $X$, according to the same distribution. The symbols M for steMs and X for suffiXes are used throughout this paper.

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:

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

For each suffix $i$, choose its length in letters, $L_i^x$, according to the same distribution. Assuming that the suffix lengths are chosen independently, the joint distribution on suffix lengths has the same form as (2).

3. Let $\Sigma$ be the alphabet and let $\{p_1 \ldots p_{|\Sigma|}\}$ be a probability distribution on $\Sigma$. 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,

$$\Pr(\text{STEM}, \text{SUFF} \mid M, L^m, X, L^x) \quad (3)$$
$$= M!X! \prod_{l \in \Sigma} \left(\frac{c_l}{S}\right)^{c_l}$$

The factorial terms reflect the fact that the stems in the set STEM can be generated in any order, as can the suffixes in the set SUFF.

4. For each stem $i$ in STEM, choose the number of suffixes, $\text{Freq}_i$, which $i$ will be paired with in order to generate the lexicon. Since $\text{Freq}_i$ is an integer between one and $X$ we can use a uniform distribution:

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

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

$$\Pr(\text{Freq} \mid M, X) = \left(\frac{1}{X}\right)^M \quad (4)$$

5. For each stem $i$ in STEM, choose a set of suffixes, $D_i$, of size $\text{Freq}_i$, 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 $\text{Freq}_i$ 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:

$$\Pr(D \mid M, X, \text{Freq}) = \prod_{i=1}^{M} \binom{X}{\text{Freq}_i}^{-1} \quad (5)$$

The probability of a hypothesis about how to split every word in the input lexicon can be computed by evaluating equations (1)-(5) and multiplying the results. These equations depend only on the following characteristics of the hypothesis:

- The number of stems ($M$) and suffixes ($X$)
- The lengths of the stems ($L^m$) and suffixes ($L^x$)
- The number of suffixes that occur with each stem (Freq)
- The count of each letter in the combined stem and suffix sets ($\{c_l\}$).

Intuitively, the primary determinants of the probability of a hypothesis are the total number of characters in the stem and suffix sets ($\sum_l c_l$) and the number of suffixes $X$. Every decision about whether to add a suffix to the current hypothesis reflects a tradeoff between these opposing forces. Adding a frequent suffix, such as *-ing*, eliminates many copies of *-ing* from many different stems, but it only adds one copy to the suffix set. This reduces the total number of characters greatly, and therefore increases the probability in (3). However, it also increases the number of suffixes, greatly decreasing the probabilities in (4) and (5). Other determinants of probability of a hypothesis, such as the number of stems and the lengths of the stems, have a lesser impact on the overall probability of a hypothesis.

## 3 Search

This section describes the search algorithm we used to find the most likely segmentations of all the words in the input lexicon $L$. We first define the hypothesis space in terms of a lattice of sets of suffixes. This provides a very general framework within which a variety of search algorithms can be stated. Next, we define the initial hypothesis for our search. Finally, we describe our algorithm for finding a local maximum in the hypothesis space.

### 3.1 The Hypothesis Space

The input to our morphology induction system is a set of words, or lexicon, $L$. We define the hypothesis space for this system 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 $h(\textit{walk}) = \{\epsilon, \textit{s}, \textit{ing}\}$ then the stem *walk* generates the words *walk, walks,* *walking*. If $h(m)$ is the empty set then $m$ generates no words. In this paper, we consider only hypotheses 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 of the input lexicon, $L$.

Our search algorithm is defined in terms of a lattice diagram of the subsets of pSuff under the inclusion relation. Figure 1 shows a portion of this lattice for the lexicon consisting of the words *build*, *builds*, and *building*. The nodes shown are those

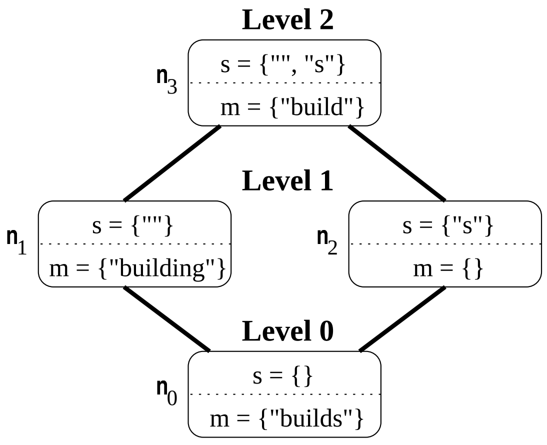

Figure 1: A Simple Graph Representation

whose suffix sets are subsets of $\{\epsilon, s\}$. We designate the set of suffixes corresponding to a node $n$ by $s(n)$. In Figure 1, $s(n)$ is shown above the dotted line in each node $n$. Below the dotted line is the set of stems corresponding to the hypothesis that *build* is the stem for *build* and *builds* and that *building* is the stem for *building*. Node $n_1$ generates the word *building* and node $n_3$ generates *build* and *builds*. Nodes $n_0$ and $n_2$ do not generate any words. Note that the empty suffix (denoted by $\epsilon$ or by “”) is treated like any other suffix: The suffix set containing only $\epsilon$ 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 $n$ ($l(n) = |s(n)|$). 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:

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

## 3.2 The Initial Hypothesis

The search starts with the hypothesis in which all stems corresponding to complete words in the input lexicon $L$ map to the lattice node containing the empty suffix (a node at level 1). All other possible stems map to the lattice node containing the empty set of suffixes (the unique node at level 0). Hence, the initial hypothesis satisfies Invariants 1 and 2.

## 3.3 Search Operators

We now focus on search algorithms 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 $\text{CompM}(m, x, h)$ and $\text{CompX}(m, x, h)$ are the two unique strings such that:

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

The uniqueness of $\text{CompM}(m, x, h)$ and $\text{CompX}(m, x, h)$ 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 $\text{Prom}(m_1, x, h)$ 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 $m_1 x \notin L$, $\text{Prom}(m_1, x, h) = h$.

2. Otherwise, $\text{Prom}(m_1, x, h)(m) =$

$$\begin{array}{l} h(\text{m}) \cup \{x\} \\ \quad \text{if } m = m_1 \\ h(\text{m}) - \{\text{CompX}(m_1, x, h)\} \\ \quad \text{if } m = \text{CompM}(m_1, x, h) \\ h(\text{m}) \\ \quad \text{otherwise} \end{array}$$

Unlike promotion, demotion of a stem $m_1$ from a node containing suffix $x$ to an adjacent node not containing $x$ is always possible. The result is that $m_1$ no longer generates word $m_1 x$, 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 $\text{Dem}(m_1, x, h)$ is the hypothesis such that:

1. If $x \notin h(m_1)$, $\text{Dem}(m_1, x, h) = h$.

2. Otherwise, $\text{Dem}(m_1, x, h)(m) =$

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

*Remark.* $\text{Dem}(m_1, x, h)$ is equivalent to $\text{Prom}(m_1 x, \epsilon, h)$. Thus, Dem is a special case of Prom and all properties that hold for $\text{Prom}(m_1, x, h)$ 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 to node $n$ under hypothesis $h$. Then $\text{Prom}(n, x, h)$ is the result of composing the promotion operators for the $k$ stems:

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

Similarly, $\text{Dem}(n, x, h)$ is defined as the composition of demotion operators for all the stems in node $n$.

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.

A sketch of the proof can be found in Appendix A.

## 3.4 Search Algorithm

The search used in the experiments reported below 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
2:    For each n ∈ lattice(pSuff)
3:      if Pr(Prom(n, x, h)) > Pr(h)
4:          h = (Prom(n, x, h))
5:          Goto 2
```

The demotion phase is exactly analogous.

## 3.5 Suffix Ordering Heuristic

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

| Size | WSJ | Hansard English |
|---|---|---|
| 500 | $\epsilon$ s | $\epsilon$ s |
| 1,000 | $\epsilon$ s | $\epsilon$ s d ed ing ly |
| 2,000 | $\epsilon$ s | $\epsilon$ s d ed ing ly |
| 4,000 | $\epsilon$ s d ed ing s ly | $\epsilon$ s d ed ing ly |
| 8,000 | $\epsilon$ s d ed ing s ly | $\epsilon$ s d ed ing ly |
| 16,000 | $\epsilon$ s d ed ing s ly | $\epsilon$ s d ed ing ly |
| 32,000 | N/A | $\epsilon$ s d ed ing ly |

Table 1: English

| Size | Hansard French |
|---|---|
| 500 | $\epsilon$ s |
| 1,000 | $\epsilon$ s |
| 2,000 | $\epsilon$ s $\underline{\text{r}}$ |
| 4,000 | $\epsilon$ s $\underline{\text{r}}$ |
| 8,000 | $\epsilon$ s $\underline{\text{r}}$ $\underline{\text{nt}}$ ment |
| 16,000 | $\epsilon$ s $\underline{\text{r}}$ $\underline{\text{nt}}$ ment e es $\underline{\text{it}}$ $\underline{\text{ient}}$ |
| 32,000 | $\epsilon$ s $\underline{\text{r}}$ $\underline{\text{nt}}$ ment e es $\underline{\text{it}}$ $\underline{\text{ient}}$ $\underline{\text{ront}}$ |

Table 2: French

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.

## 4 Experiment

We tested a suffix discovery system consisting of the probability model described in Section 2 and the search algorithm described in Section 3. The input consisted of word lists extracted from the TreeBank Wall Street Journal corpus and the French and English versions of set A of the Hansard Corpus. The Hansard corpora are proceedings of the Canadian Parliament, transcribed into French or English and translated into the other language.

### 4.1 Input

We extracted input lexicons from each corpus, excluding words containing capital letters or non-alphabetic characters. The 100 most common words in each corpus were also excluded, since these words tend not to have regular inflection. The system was run on the 500, 1,000, 2,000, 4,000, 8,000, 16,000, and 32,000 most common remaining words. The Treebank corpus only had enough words to run through 16,000.

### 4.2 Results

Table 1 shows the suffixes found in lexicons extracted from each of the English corpora at each lexicon size. All productive inflectional suffixes, as well as the adjective-generating suffix *ly*, are found by the time the input size reaches 1,000 words from the Hansard English corpus and 4,000 words from the WSJ corpus. Two variants of the past tense and past participle suffix are found: *ed*, the general form, and *d*. In reality, these should be treated as single suffix, *ed*, together with a spelling-change rule that deletes the "e" after words ending in a vowel.

Table 2 shows the suffixes found in the lexicon extracted from the Hansard French corpus at each lexicon size. The suffixes $\epsilon$ and $s$ are found at all lexicon sizes, reflecting the regular plural inflection of nouns and adjectives. At 2,000 words the suffixes $r$ is hypothesized, reflecting the fact that verbal infinitives end in "r". Consider, for example, the present indicative paradigm of the verb whose infinitive is *parler*:

| | | | |
|---|---|---|---|
| je | parle | nous | parlons |
| tu | parles | vous | parlez |
| il | parle | ils | parlent |

The system takes *parle* as the stem and $r$ as the suffix on the infinitive. All the errors, which are truncations of true suffixes, can be understood as resulting from the failure to include the "e" in such cases as part of the stem. Errors are underlined in Table 2. However, it should be emphasized that these are relatively benign errors — they are not fictitious suffixes pulled out of thin air, but genuine suffixes that were split from the stem at the wrong place. At 8,000 words *nt* is split off of third person plural verbs and *ment* is split off of adverbs. At 16,000 words two changes occur. First, *e* is split of of feminine singular adjectives and past participles, and *es* is split off of feminine plural adjectives and past participles. (The suffix *e* is not split off of tensed verbs, since the system treats it as part of the stem.) Second, the system splits *it* and *ient* of off imperfects, and *nt* off both third person plural indicatives and present participles. Here, it is treating an "a" as part of the stem when the normal analysis puts it in the suffixes *ait*, *aient*, and *ant*. Thus, it posits a second stem ending in "a", such as *parla*, for many of the verb stems that it treats as ending in "e", such as *parle*. At 32,000 words *ront* is split off of third person plural verbs in the future tense, such as *parleront*. At no point is a single stem, such as *parl*, identified for all forms of the verb. Instead, the system eventually posits two stems, *parle* and *parla*. This behavior is explained in the next sections.

### 4.3 Discussion of the Experiment

Our goal was to develop a system capable of identifying a small number of highly productive suffixes, with very few false positives, across a wide range of input sizes. The low false positive rate was emphasized because this system is intended as the first stage of a much more ambitious system for learning irregular morphology, morphosyntax, and spelling adjustment rules as well as suffixes. It is best be conservative at the first stage, since an early mis-step could be magnified at later stages of processing.

In light of this goal, the results on both of the English corpora were excellent. All of the productive regular inflectional suffixes were discovered by 1,000-4,000 words and remained stable over several binary orders of magnitude. In additon to the standard forms of the regular inflections, *d* was hypothesized to be a suffix and was split off of words like *closed* and *used* — the past forms of stems ending in "e". The correct analysis in this case requires a spelling change: the suffix is *ed*, but when the stem ends in "e" one of the two "e"'s is deleted. A hypothesized suffix set including both *ed* and *d* provides a good starting point for a possible future system aimed at discovering spelling change rules. There are a few moderately productive derivational suffixes, such as *er*, which the sytem did not choose. This is presumably because the probability model favors only the most productive suffixes. The model could easily be changed to identify suffixes such as -it -er, but it is not clear whether this can be done without also outputting other common word-endings that are not morphemes.

The results in French are also good, but they suffer from one persistent error: the first vowel of the suffix is often misanalyzed as part of the stem. This results in hypothesizing two stems, such as *parle* and *parla*, for words that should share a single stem, such as *parl*. In fact, with the current search procedure and initial hypothesis, it is only possible to hypothesize stems that appear as words in the input lexicon. Thus, *parle* and *parla*, both of which appear in the input, can be hypthesized, but *parl*, which is not a word, cannot. To see this, recall that the initial hypothesis maps all stems that occur as words to the lattice node containing the empty suffix; it maps all other stems to the lattice node containing no suffixes (the unique node at level 0). The search never applies the Prom operator to this node directly. Further, Definition 3 specifies that the compensating stem for a demotion always occurs as a word in the lexicon. This limitation to stems that occur as words only became apparent when the system was tested on French, since it has no ill effect on the English results. The next section discusses overcoming this limitation by changing the initial hypothesis.

It worth noting that, although the exact division between stem and suffix may not be right, every suffix the system hypothesized corresponds to an actual productive morpheme.

## 5 General Discussion

The lattice of suffix sets and the invariants and definitions presented above provide a useful formalism for describing alternative search algorithms. Some of the improvements we hope to make are sketched briefly in this section.

*Improving the initial hypothesis* Our immediate goal is to enable the system to hypothesize stems that do not appear in the input, such as the French *parl*. The approach will be to construct a good starting hypothesis in a preprocessing phase by heuristically evaluating individual nodes in the suffix lattice and greedily filling the best ones with as many stems as possible. The invariants will be satisfied only when the initial hypothesis has been completed. Because it is greedy, this process will not find an optimal hypothesis, but it is expected to improve on the current initial hypothesis. This has been implemented and preliminary results are very promising.

*Pulling stems to good nodes* The current search carries out promotions and demotions without any particular goal node toward which stems are being moved. The only goal is to improve the overall probability of the entire hypothesis. Using a heuristic evaluation of individual lattice nodes, however, it should be possible to develop a more directed search that attempts to move stems toward a highly valued lattice node. We believe that it will be possible to efficiently determine whether a given stem can move to a given node without violating the invariants, and if so, what other changes must be made to maintain the invariants. This would make it possible to implement a new search operator, which might be called Pull$(n, h)$. This operator would check each stem to see (1) whether it can be moved to node $n$, (2) if so, what compensating actions are necessary to maintain the invariants, and (3) whether such a move would improve the overall probability of the hypothesis. It would move each stem that can be moved to good effect.

*Relaxing the invariants* Invariant 1 ensures that a hypothesis generates each word in the input in exactly one way and does not generate any other words. This invariant could be relaxed somewhat, allowing the system to hypothesize that a given form of a word exists, but was not seen in the input. One reason a given form might not be in the input is sampling error. We plan to modify the probability model so that it can use word frequency to estimate the probability that a given form was missed due to sampling error.

This is just a sampling of the interesting algorithms that can be investigated using the formalisms developed here. These avenues of research are likely to lead to better, more comprehensive systems for automated morphological analysis.

## References

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

M. R. Brent. 1993. 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. Erlbaum.

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

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

J. Goldsmith. 2000. Unsupervised learning of the morpholgy of a natural language. http://humanities.uchicago.edu/faculty/goldsmith.

L. Karttunen and K. Wittenburg. 1983. A two-level morphological analysis of english. *Texas Linguistics Forum 22*, 22:217–223.

L. Karttunen. 1983. KIMMO: a general morphological processor. *Texas Linguistics Forum 22*, 22:165–186.

P. Schone and D. Jurafsky. 2000. 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.

Richard Sproat. 1992. *Morphology and Computation*. MIT Press, Cambridge, MA.

A. Van den Bosch and W. Daelemans. 1999. Memory-based morphilogical analysis. In *Proc. of the 37th Annual Meeting of the ACL*. ACL.

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

## A Proof of Theorem 1

**Lemma 1** If $m_1 \neq m_2$ then

$$\mathrm{CompM}(m_1, x, h) = \mathrm{CompM}(m_1, x, \mathrm{Prom}(m_2, x, h)),$$
$$\mathrm{CompX}(m_1, x, h) = \mathrm{CompX}(m_1, x, \mathrm{Prom}(m_2, x, h)).$$

*Proof.* By Definition 1,

$$\mathrm{CompM}(m_1, x, h)\mathrm{CompX}(m_1, x, h) = m_1 x,$$

so $\mathrm{CompM}(m_1, x, h)$ and $\mathrm{CompX}(m_1, x, h)$ satisfy the first condition of the definition of $\mathrm{CompM}(m_1, x, h')$ and $\mathrm{CompM}(m_1, x, h')$ for any hypothesis $h'$. To show that they also satisfy the second condition, we must show that

$$\begin{aligned}&\mathrm{CompX}(m_1, x, h)\\ &\qquad \in \mathrm{Prom}(m_2, x, h)(\mathrm{CompM}(m_1, x, h))\end{aligned}$$

By definition 1, we know

$$\mathrm{CompX}(m_1, x, h) \in h(\mathrm{CompM}(m_1, x, h)), \tag{6}$$

so if we are in a case where $\mathrm{Prom}(m_2, x, h) = h$ then the lemma is established. Otherwise, definition 2 gives us:

$$\begin{aligned}&\mathrm{Prom}(m_2, x, h)(\mathrm{CompM}(m_1, x, h)) =\\ &h(\mathrm{CompM}(m_1, x, h)) \cup \{x\} && \text{(a)}\\ &\quad \text{if } \mathrm{CompM}(m_1, x, h) = m_2\\ &h(\mathrm{CompM}(m_1, x, h)) - \{\mathrm{CompX}(m_2, x, h)\} && \text{(b)}\\ &\quad \text{if } \mathrm{CompM}(m_1, x, h) = \mathrm{CompM}(m_2, x, h)\\ &h(\mathrm{CompM}(m_1, x, h)) \quad \text{otherwise} && \text{(c)}\end{aligned}$$

Thus, if $\mathrm{CompM}(m_1, x, h) \neq \mathrm{CompM}(m_2, x, h)$ then

$$\begin{aligned}&h(\mathrm{CompM}(m_1, x, h))\\ &\qquad \subseteq \mathrm{Prom}(m_2, x, h)(\mathrm{CompM}(m_1, x, h))\end{aligned}$$

and the lemma is established. Finally, suppose $\mathrm{CompM}(m_1, x, h) = \mathrm{CompM}(m_2, x, h)$ and hence case (b) applies. Then

$$\mathrm{CompX}(m_1, x, h) \neq \mathrm{CompX}(m_2, x, h); \tag{7}$$

otherwise we would have:

$$\begin{aligned}m_1 x &= \mathrm{CompM}(m_1, x, h)\mathrm{CompX}(m_1, x, h)\\ &= \mathrm{CompM}(m_2, x, h)\mathrm{CompX}(m_2, x, h)\\ &= m_2 x,\end{aligned}$$

and hence $m_1 = m_2$, contradicting the hypothesis of the lemma. Since we are in case (b), (7) and (6) imply that

$$\begin{aligned}&\text{CompX}(m_1, x, h)\\ &\qquad \in \text{Prom}(m_2, x, h)(\text{CompM}(m_1, x, h))\end{aligned}$$

and the lemma is established for all cases.

It is sufficient to show that

$$\text{Prom}(m_1, x, \text{Prom}(m_2, x, h)) \qquad \text{(a)}$$

is invariant under swapping of the subscripts. Space limitations permit only a sketch of the proof. Expanding (a) by two applications of Definition 2 yields 9 cases. If $m_1 = m_2$ it obvious that (a) is invariant under subscript swapping, so we assume $m_1 \neq m_2$. This allows us to drop the first case and apply Lemma 1 to simplify the 8 remaining cases to:

$$\begin{aligned}
&h(m) - \{\text{CompX}(m_2, x, h)\} \cup \{x\} && (2)\\
&\qquad \text{if } m = m_1 = \text{CompM}(m_2, x, h)\\
&h(m) \cup \{x\} && (3)\\
&\qquad \text{if } m = m_1 \neq m_2, m \neq \text{CompM}(m_2, x, h)\\
&(h(m) \cup \{x\}) - \{\text{CompX}(m_1, x, h)\} && (4)\\
&\qquad \text{if } m = \text{CompM}(m_1, x, h) = m_2\\
&h(m) - \{\text{CompX}(m_2, x, h)\} && (5)\\
&\quad - \{\text{CompX}(m_1, x, h)\}\\
&\qquad \text{if } m = \text{CompM}(m_1, x, h) = \text{CompM}(m_2, x, h)\\
&h(m) - \{\text{CompX}(m_1, x, h)\} && (6)\\
&\qquad \text{if } m = \text{CompM}(m_1, x, h),\\
&\qquad m \neq \text{CompM}(m_2, x, h), \text{ and } m \neq m_2\\
&h(m) \cup \{x\} && (7)\\
&\qquad \text{if } m \neq \text{CompM}(m_1, x, h),\ m \neq m_1,\ m = m_2\\
&h(m) - \{\text{CompX}(m_2, x, h)\} && (8)\\
&\qquad \text{if } m \neq m_1,\ m = \text{CompM}(m_2, x, h),\\
&\qquad \text{and } m \neq \text{CompM}(m_1, x, h)\\
&h(m) \quad \text{otherwise} && (9)
\end{aligned}$$

For cases (5), and (9), swapping the subscripts does not change either the condition or the consequent. For cases (3) and (7), swapping subscripts swaps the conditions, but the consequents are identical. For cases (6) and (8), swapping subscripts swaps both the conditions and the consequents, leaving the overall definition unchanged. For cases (2) and (4), swapping the subscripts clearly swaps the conditions; it also swaps the consequents, so long as $x \neq \text{CompX}(m_1, x, h)$ and $x \neq \text{CompX}(m_2, x, h)$. Now suppose one of those two conditions does not hold.

Without loss of generality, suppose $x = \text{CompX}(m_2, x, h)$. By Lemma 2 (see below), $\text{Prom}(m_2, x, h) = h$. Thus,

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

We now establish the theorem by showing that

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

By Lemma 1,

$$\text{CompM}(m_2, x, h) = \text{CompM}(m_2, x, \text{Prom}(m_1, x, h))$$

Thus, we have

$$x = \text{CompM}(m_2, x, \text{Prom}(m_1, x, h)).$$

Now Lemma 2 can be applied, substituting $\text{Prom}(m_1, x, h)$ for $h'$. This yields

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

**Lemma 2** If $x = \text{CompX}(m_2, x, h')$ then $\text{Prom}(m_2, x, h') = h'$. Proof omitted.