Proceedings of the Workshop of the
ACL Special Interest Group on Computational Phonology (SIGPHON)
Association for Computations Linguistics
Barcelona, July 2004

Priors in Bayesian Learning of Phonological Rules

Sharon Goldwater and **Mark Johnson**
Department of Cognitive and Linguistic Sciences
Box 1978
Brown University
Providence, RI 02912
USA
{sharon_goldwater, mark_johnson}@brown.edu

Abstract

This paper describes a Bayesian procedure for unsupervised learning of phonological rules from an unlabeled corpus of training data. Like Goldsmith's *Linguistica* program (Goldsmith, 2004b), whose output is taken as the starting point of this procedure, our learner returns a grammar that consists of a set of *signatures*, each of which consists of a set of *stems* and a set of *suffixes*. Our grammars differ from Linguistica's in that they also contain a set of *phonological rules*, specifically insertion, deletion and substitution rules, which permit our grammars to collapse far more words into a signature than Linguistica can. Interestingly, the choice of Bayesian prior turns out to be crucial for obtaining a learner that makes linguistically appropriate generalizations through a range of different sized training corpora.

1 Introduction

Unsupervised learning presents unusual challenges to the field of computational linguistics. In supervised systems, the task of learning can often be restricted to finding the optimal values for the parameters of a pre-specified model. In contrast, an unsupervised learning system must often propose the structure of the model itself, as well as the values for any parameters in that model. In general, there is a trade-off between the structural complexity of a model and its ability to explain a set of data. One way to deal with this trade-off is by using Bayesian learning techniques, where the objective function used to evaluate the overall goodness of a system takes the form

$$\Pr(H)\Pr(D|H)$$

where $\Pr(H)$ is the prior probability of the hypothesized model H, and $\Pr(D|H)$ is the likelihood of the data D given that model. In a Bayesian system, we want to find the hypothesis H for which $\Pr(H)\Pr(D|H)$ is highest (or, equivalently, where $-\log\Pr(H) - \log\Pr(D|H)$ is lowest). While calculating the likelihood of the data given a particular hypothesis is generally straightforward, the more difficult question in Bayesian learning is how to determine the prior probabilities of various hypotheses.

In this paper, we compare the results of using two different prior distributions for an unsupervised learning task in the domain of morpho-phonology. Our goal is to learn transformation rules of the form $x \rightarrow y \ / \ C$, where x and y are individual characters (or the empty character ϵ) and C is some representation of the context licensing the transformation. Our input is an existing segmentation of words from the Penn Treebank (Marcus et al., 93) into stems and suffixes. This segmentation is provided by the Linguistica morphological analyzer (Goldsmith, 2001; Goldsmith, 2004b), itself an unsupervised algorithm. Using the transformation rules we learn, we are able to output a new segmentation that more closely matches our linguistic intuitions.[1]

We are not the first to apply Bayesian learning techniques for unsupervised learning of morphology and phonology. Several other researchers have also pursued these methods, usually within a Minimum Description Length (MDL) framework (Rissanen, 1989). In MDL approaches, $-\log\Pr(H)$ is taken to be proportional to the length of H in some standard encoding, and $-\log\Pr(D|H)$ is the length of D using the encoding specified by H. MDL-based systems have been relatively successful for tasks including word segmentation (de Marcken, 1996; Brent and Cartwright, 1996), morphological

[1] Since we use ordinary text, rather than phonological transcriptions, as input, the rules we learn are really spelling rules, not phonological rules. We believe that the work discussed here would be equally applicable, and possibly more successful, with phonological transcriptions. However, since we wish to have an entirely unsupervised system and we require a morphological segmentation as input, we are currently limited by the capabilities of Linguistica, which requires standard textual input. For the remainder of this paper, we use "phonology" and "phonological rules" in a broad sense to include orthography as well.

$$\left\{\begin{array}{c}\text{lift}\\ \text{jump}\\ \text{roll}\\ \text{walk}\\ \dots\end{array}\right\} \times \left\{\begin{array}{c}\epsilon\\ -\text{s}\\ -\text{ed}\\ -\text{ing}\\ \dots\end{array}\right\}$$

Figure 1: An example signature

segmentation (Goldsmith, 2001; Creutz and Lagus, 2002), discovery of syllabicity and sonority (Ellison, 1993), and learning constraints on vowel harmony and consonant clusters (Ellison, 1994). However, our work shows that a straightforward MDL approach, where the prior $-\log \Pr(H)$ depends on the length of the phonological rules and the rest of the grammar in the obvious way, does not result in a successful system for learning phonological rules. We discuss why this is so, and then present several changes that can be made to the prior in order to learn phonological rules successfully. Our conclusion is that, although Bayesian techniques can be successful for unsupervised learning of linguistic information, careful choice of the prior, with attention to both linguistic and statistical factors, is important.

In the remainder of this paper, we first review the basics behind Goldsmith's Linguistica program, which serves as the starting point for our own work. We then explain the additional framework necessary for learning phonological rules, and describe our search algorithm. In Section 5, we describe the results of two experiments using our search algorithm, first with an MDL prior, then with a modified prior. We discuss why the modified prior works better for our task, and implications for other Bayesian learners.

2 Linguistica

Since our algorithm is designed to take as input a morphological analysis produced by Linguistica, we first briefly review what that analysis consists of and how it is arrived at. Linguistica is based on the MDL principle, which states that the optimal hypothesis to explain a set of data is the one that minimizes the total number of bits required to describe both the hypothesis and the data under that hypothesis. Information theory tells us that the description length of the data under a given hypothesis is simply the negative log likelihood of the data, so the MDL criterion is equivalent to a Bayesian prior favoring hypotheses that can be described succintly.

Linguistic hypotheses (grammars) all contain some primitive types. Linguistica uses three primitive types in its grammar: stems, suffixes, and signatures. [2] Each signature is associated with a set of stems, and each stem is associated with exactly one signature representing those suffixes with which it combines freely. For example, *walk* and *jump* might be associated with the signature $\langle \epsilon.ed.ing.s \rangle$ (see Figure 1), while *bad* might be associated with $\langle \epsilon.ly \rangle$. Unanalyzed words can be thought of as belonging to the $\langle \epsilon \rangle$ signature. A possible grammar under this scenario consists of a set of signatures, where each signature contains a set of stems and a set of suffixes. Rather than modeling the probability of each word in the corpus directly, the grammar assumes that each word consists of a stem and a (possibly empty) suffix, and assigns a probability to each word w according to

$$\Pr(w = t + f) = \Pr(\sigma)\Pr(t|\sigma)\Pr(f|\sigma),$$

where σ is the signature containing stem t. (We have adopted Goldsmith's notation here, using f to denote suffixes, t for stems, and σ for signatures.) Clearly, grouping words into signatures will cause their probabilities to be modeled less well than modeling each word individually. The negative log likelihood of the corpus will therefore increase, and this portion of the description length will grow. However, listing each word individually in the grammar requires as many stems as there are words. Assigning words to signatures significantly reduces the number of stems, and thus the length of the grammar. If the stems are chosen well, then the length of the grammar will decrease more than the length of the encoded corpus will increase, leading to an overall win. Goldsmith (2001) provides a detailed description of the exact grammar encoding and search heuristics used to find the optimal set of stems, suffixes, and signatures under this type of model.

Goldsmith's algorithm is not without its problems, however. We concern ourselves here with its tendency to postulate spurious signatures in cases where phonological constraints operate. For example, many English verb stems ending in *e* are placed in the signature $\langle e.ed.es.ing \rangle$, while stems not ending in *e* have the signature $\langle \epsilon.ed.ing.s \rangle$. This is due to the fact that the stem-final *e* deletes before suffixes beginning in *e* or *i*. Similarly, words like *match* and *index* are likely to be given the signature $\langle \epsilon.es \rangle$, whereas most nouns would be $\langle \epsilon.s \rangle$. The toy grammar G_1 in Figure 2 illustrates the sort of analysis produced by Linguistica.

Goldsmith himself has noted the problem of spurious signatures (Goldsmith, 2004a), and recent ver-

[2] Linguistica actually can perform prefix analysis as well as suffix analysis, but in our work we used only the suffixing functions.

σ_1 = ({work, roll}×{ϵ, ed, ing, er})
σ_2 = ({din, bik}×{e, ed, ing, er})
σ_3 = ({wait}×{ϵ, ed, er})
σ_4 = ({carr}×{y, ied, ier})
σ_5 = ({carry}×{ϵ, ing})
σ_6 = ({bike, booth, worker}×{ϵ, s})
σ_7 = ({beach, match}×{ϵ, es})

Figure 2: G_1: A Sample Linguistica Grammar

sions of Linguistica include some functionality devoted to detecting allomorphs. Superficially, our work may seem similar to Goldsmith's, but in fact it is quite different. First of all, the allomorphic variation detected by Linguistica is suffix-based. That is, suffixes are proposed that operate to delete certain stem-final material. For example, a suffix *(e)ing* could be proposed in order to include both *hope* and *walk* in the signature $\langle$*ϵ.(e)ing.s*$\rangle$. This suffix is actually separate in the grammar from the ordinary *ing* suffix, so there is no recognition of the fact that *any* occurrence of *ing* in any signature should delete a preceding stem-final *e*. Moreover, this approach is not really phonological, in the sense that other suffixes beginning with *i* might or might not be analyzed as deleting stem-final *e*. While many languages do contain some affix-specific morpho-phonological processes, our goal here is to find phonological rules that apply at all stem-suffix boundaries, given certain context criteria.

A second major difference between the allomorphy detection in Linguistica and the work presented here is that a Linguistica suffix such as *(e)ing* is assumed to delete any stem-final *e*, without exception. While this assumption may be valid in this case, there are other suffixes and phonological processes that are not categorical. For example, the English plural *s* requires insertion of an *e* after certain stems, including those ending in *x* or *s*. However, there is no simple way to describe the context for this rule based solely on orthography, because of stems such as *beach (+es)* and *stomach (+s)*. For this reason, and to add robustness against errors in the input morphological segmentation, we allow stems to be listed in the grammar as exceptions to phonological rules.

In addition to these theoretical differences, the work presented here covers a wider range of phonological processes than does Linguistica. Linguistica is capable of detecting only stem-final deletion, whereas our algorithm can also detect insertion (as in *match + s* $\rightarrow$ *matches*) and stem-final substitution (as in *carry + ed* $\rightarrow$ *carried*). In the following section we discuss the structure of the grammar we use to describe the words in our corpus.

3 A Morpho-Phonological Grammar

Since the morphology we use as input to our program is obtained directly from Linguistica, our grammar is necessarily similar to the one in that program. As discussed above, Linguistica contains three primitives types in its grammar: signatures, stems, and suffixes. We add one more primitive type to our grammar, the notion of a *rule*.

Each rule consists of a transformation, for example $\epsilon \rightarrow e$ or $y \rightarrow i$, and a conditioning context. A context consists of a string of four characters $X_t y_t y_f X_f$, where $X_i \in \{C, V, \#\}$ (consonant, vowel, end-of-word) and y_i is in the set of characters in our text.[3] The first half of the context is from the end of the stem, and the second half is from the beginning of the suffix. For example, the stem-suffix pair *jump + ed* has the context $CpeC$. All transformations are assumed to occur stem-finally, i.e. at the second context position (or after the second position, for insertions). Of course, these contexts are more detailed than necessary for certain phonological rules, and don't capture all the information required for others. In future work, we plan to allow for different types of contexts and generalization over contexts, but for the present, all contexts have the same form.

Using these four primitives, we can construct a grammar in the following way: As in Goldsmith's work, we list a set of signatures, each of which contains a set of stems and suffixes. In addition, we list a set of phonological rules. In many cases, only one rule will apply in a particular context, in which case it applies to all stem-suffix pairs that meet its context. If more than one rule applies, we list the rule with the most common transformation first and assume that it applies unless a particular stem specifies otherwise. Stems can thus be listed as exceptions to rules by using a non-default *no-change* rule with the appropriate context. Note that the more exceptions a rule has, the more expensive it is to add to the grammar: each new type of transformation in a particular context must be listed, and each stem requiring a non-default transformation must specify the transformation required. Any prior preferring short grammars will therefore tend

[3]The knowledge of which characters are consonants and which are vowels is the only information we provide to our program, other than the text corpus and the Linguistica-produced morphology. Aside from the C/V distinction, our program is entirely knowledge-free.

σ_1 = ({work, roll, dine, carry}$\times${ϵ, ed, er, ing})
σ_2 = ({bike}$\times${ϵ, ed, er, ing, s})
σ_3 = ({wait}$\times${ϵ, ed, er})
σ_4 = ({booth (r_5), worker, beach, match}$\times${ϵ, s})

r_1 = $e \rightarrow \epsilon$ / $CeeC$
r_2 = $e \rightarrow \epsilon$ / $CeiC$
r_3 = $y \rightarrow i$ / $CyeC$
r_4 = $\epsilon \rightarrow e$ / $Chs\#$
r_5 = *no-change* / $Chs\#$

Figure 3: G_2: A Sample Grammar with Transformation Rules

to reject rules requiring many exceptions (i.e. those without a consistent application context).

Grammar G_2, in Figure 3, shows a sample of the kind of grammar we use. This grammar generates exactly the same wordforms as G_1, but using fewer signatures due to the effects of the phonological rules. All the stem-suffix parings in this grammar undergo the default rules for their contexts except for the stem *booth*, which is listed as an exception to the *e*-insertion rule. For *booth* + *s*, the grammar therefore generates *booths*, not *boothes*.

Our model generates data in much the same way as Goldsmith's: a word is generated by selecting a signature and then independently generating a stem and suffix from that signature. This means that the likelihood of the data takes the same form in our model as in Goldsmith's, namely $\Pr(w) = \Pr(\sigma)\Pr(t|\sigma)\Pr(f|\sigma)$, where the word w consists of a stem t and a suffix f both drawn from the same signature σ. Our model differs from Goldsmith's in the way that stems and suffixes are produced; because we use phonological rules a great many more stems and suffixes can belong to a single signature. We defer discussion of how we define the prior probability over grammars to Section 5, and assume for the moment that we are given prior and likelihood functions that can evaluate the utility of a grammar and training data.

4 Search Algorithm

Since it is clearly infeasible to evaluate the utility of every possible grammar, we need a search algorithm to guide us toward a good solution. Our algorithm uses certain heuristics to make small changes to the initial grammar (the one provided by Linguistica), evaluating each change using our objective function, and accepting or rejecting it based on the result of evaluation. Our algorithm contains three major components: a procedure to find signatures that are similar in ways suggesting phonological change, a procedure to identify possible contexts for phonological change, and a procedure to collapse related signatures and add phonological rules to the grammar. We discuss each of these components in turn.

4.1 Identifying Similar Signatures

An important first step in simplifying the morphological analysis of our data using phonological rules is to identify signatures that might be related via such rules. Since our algorithm considers three different types of possible phonological processes (deletion, substitution, and insertion), there are three different ways in which signatures may be related. We need to look for pairs of signatures that are similar in any of these three ways.

Insertion We look for potential insertion rules by finding pairs of signatures in which all suffixes but one are common to both signatures. The distinct pair of suffixes must be such that one can be formed from the other by inserting a single character at the beginning. Example pairs found by our algorithm include $\langle \epsilon.s \rangle / \langle \epsilon.es \rangle$ and $\langle \epsilon.y \rangle / \langle \epsilon.ly \rangle$. In searching for these pairs (as well as deletion and substitution pairs), we consider only pairs where each signature contains at least two stems. This is partly in the interests of efficiency and partly due to the fact that signatures with only one stem are often less reliable.

Deletion Signature pairs exhibiting possible deletion behavior are similar to those exhibiting insertion behavior, except that one of the suffixes not common to both signatures must be the empty suffix. Examples of possible deletion pairs include $\langle \epsilon.ed.ing \rangle / \langle e.ed.ing \rangle$ and $\langle \epsilon.ed.ing \rangle / \langle ed.ing.s \rangle$.

Substitution In a possible substitution pair, one signature (the one potentially exhibiting stem-final substitution) contains suffixes that all begin with one of two characters: the basic stem-final character, and the substituted character. The signature $\langle ied.ier.y \rangle$ from G_1 is such a signature. The other signature in a possible substitution pair must contain the empty suffix, and the two signatures must be identical when the first character of each suffix in the first signature is removed. Possible substitution pairs include $\langle ied.ier.y \rangle / \langle \epsilon.ed.er \rangle$ and $\langle ous.y \rangle / \langle \epsilon.us \rangle$.

Using the set of similar signatures we have detected, we can propose a set of possible phonological processes in our data. Some transformations, such as $e \rightarrow \epsilon$, will be suggested by more than one pair of signatures, while others, such as $y \rightarrow o$, will occur with only one pair. We create a list of all the possible transformations, ranked according to the number of signature pairs attesting to them.

4.2 Identifying possible contexts

Once we have found a set of possible transformations, we need to identify the contexts in which those transformations might apply. To see how this works, suppose we are looking at the proposed *e*-deletion rule and our input grammar is G_1. Using one of the signature pairs attesting to this rule, such as $\langle \epsilon$*.ed.er.ing*$\rangle$/$\langle$*e.ed.er.ing*$\rangle$, we can find possible conditioning contexts by examining the set of stems and suffixes in the second signature. If we want to reanalyze the stems *din* and *bik* as *dine* and *bike*, we hypothesize that each wordform generated using the suffixes present in both signatures (*ed*, *er*, and *ing*) must have deleted an *e*. We can find the context for this deletion by looking at these suffixes together with the reanalyzed stems. The contexts for deletion that we would get from {*bike, dine*} $\times$ {ed, ing} are $\{CeeC, CeiC\}$.[4]

Our methods for finding possible contexts for substitution and insertion rules are similar: reanalyze the stems and suffixes in the signature hypothesized to require a phonological rule, combine them, and note the context generated. In this way, we can get contexts such as $CyeC$ for the $y \rightarrow i$ rule (from *carry* + *ed*) and $Vxs\#$ for the $\emptyset \rightarrow e$ rule (from *index* + *s*).

Just as we ranked the set of possible phonological rules according to the number of signature pairs attesting to them, we can rank the set of contexts proposed for each rule. We do this by calculating $r = \Pr(X_t y_t | y_f X_f)/\Pr(X_t y_t)$, the ratio between the probability of seeing a particular stem context given a particular suffix context to the prior probability of the stem context. If a stem context (such as Ce) is quite common overall but hardly ever appears before a particular suffix context (iC), this is good evidence that some phonological process has modified the stem in the context of that suffix. Low values of r are therefore better evidence of conditioning for a rule than are high values of r.

4.3 Collapsing signatures

Given a set of similar signature pairs, the rules relating them, and the possible contexts for those rules, we need to determine which rules are actually phonologically legitimate and which are simply accidents of the data. We do this by simply considering each rule and context in turn, proceeding from the most attested to least attested rules and from most likely to least likely contexts. For each rule-context pair, we add the rule to the grammar with that context and collapse any pairs of signatures related by the rule, as long as all stem-suffix pairs contain a context at least as likely as the one under consideration. Collapsing a pair of signatures means reanalyzing all the stems and suffixes in one of the signatures, and possibly adding exceptions for any stems that don't fit the rule. We have found that exceptions are often required to handle stems that were originally misanalyzed by Linguistica. For that reason, we prune the rules added to the grammar, and for each rule, if fewer than 2% of the stems require exceptions, we assume that these are errors and de-analyze the stems, returning the wordforms they generated to the $\langle\epsilon\rangle$ signature. We then evaluate the new analysis using our objective function, and accept it if it scores better than our previous analysis. Otherwise, we revert to the previous analysis and continue trying new rule-context pairs. Pseudocode for our algorithm is presented in Figure 4. We use the notation $\sigma_i \underset{r}{\rightarrow} \sigma_j$ to indicate that σ_i and σ_j are similar with respect to rule r, with σ_j being the more "basic" signature (i.e. adding r to the grammar would allow us to move the stems in σ_i into σ_j).

Note that collapsing a pair of signatures does not always result in an overall reduction in the number of signatures in the grammar. To see why this is

```
FINDPHONORULES()
 1  G ← grammar produced by Linguistica
 2  R ← ordered set of possible rules
 3  for each r ∈ R
 4  do
 5     C_r ← ordered set of possible contexts for r
 6     C ← ∅
 7     while C_r ≠ ∅
 8     do c ← next c ∈ C_r
 9        C_r ← C_r \ {c}
10        C ← C ∪ {c}
11        G' ← collapseInContext(G, r, C)
12        G' ← pruneRules(G')
13        if score(G') < score(G)
14          then G ← G'
15  return G

COLLAPSEINCONTEXT(G, r, C)
1  for each σ_i ∈ G
2  do for each σ_j ∈ G
3     do if (σ_i →_r σ_j) ∧ (∀(t, f) ∈ σ_i, ctx(t, f) ∈ C)
4           then collapseSigs(σ_i, σ_j)
```

Figure 4: Pseudocode for our search algorithm

[4] The reasoning we use to finding conditioning contexts for deletion rules was also described by Goldsmith (2004a), and is similar to the much earlier work of Johnson (1984).

	Morph Only		Morph+Phon	
	Small	Large	Small	Large
Tokens	100k	888k	100k	888k
Types	11313	35631	11313	35631
σs	435	1634	404	1594
Singleton σs	280	1231	259	1215
Stems	8255	24529	8186	24379
Non-ϵ Stems	2363	7673	2286	7494

Table 1: Grammatical Analysis of our Corpora

	Init. Grammar	Change
# σs	1617	-10
# Stems	24374	-17
Grammar Size:	1335425	+520
σs, Suffixes	53933	-253
Stems	1280617	+493
Phonology	875	+279
Likelihood:	6478490	+166
Total:	7813915	+686

Table 2: Effects of adding $y \rightarrow i$ rules under MDL prior (large corpus).

so, consider the effect of collapsing σ_1 and σ_2 and adding r_1 and r_2 (the *e*-deletion rules) to G_1. When the stem *bik* gets reanalyzed as *bike*, the algorithm recognizes that *bike* is already a stem in the grammar, so rather than placing the reanalyzed stem in σ_1, it combines the reanalyzed suffixes $\{\epsilon$, *ed*, *er*, *ing*$\}$ with the suffixes $\{\epsilon$, *s*$\}$ from σ_6 and creates a new signature for the stem *bike* — $\langle \epsilon$.*ed*.*er*.*ing*.*s*$\rangle$. The two stems *carr* and *carry* are also combined in this way, but in that case, the combined suffixes form a signature already present in the grammar, so no new signature is required.

5 Experiments

For our experiments with learning phonological rules, we used two different corpora obtained from the Penn Treebank. The larger corpus contains the words from sections 2-21 of the treebank, filtered to remove most numbers, acronyms, and words containing puctuation. This corpus consists of approximately 900,000 tokens. The smaller corpus is simply the first 100,000 words from the larger corpus. We ran each corpus through the Linguistica program to obtain an initial morphological segmentation. Statistics on the results of this segmentation are shown in the left half of Table 1. "Singleton signatures" are those containing a single stem, and "Non-ϵ stems" refers to stems in a signature other than the $\langle \epsilon \rangle$ signature, i.e. those stems that combine with at least one non-ϵ suffix.

The original function we used to evaluate the utility of our grammars was an MDL prior very similar to the one described by Goldsmith (2001). This prior is simply the number of bits required to describe the grammar using a fairly straightforward encoding. The encoding essentially lists all the suffixes in the grammar along with pointers to each one; then lists the phonological rules with their pointers; then lists all the signatures. Each signature is a list of stems and their pointers, and a list of pointers to suffixes. Each exceptional stem also has a pointer to a phonological rule.[5]

Our algorithm considered a total of 11 possible transformations in the small corpus and 40 in the large corpus, but using this prior, only a single type of transformation appeared in any rule in the final grammar: $e \rightarrow \epsilon$, with seven contexts in the small corpus and eight contexts in the large corpus. In analyzing why our algorithm failed to accept any other types of rules, we realized that there were several problems with the MDL prior. Consider what happens to the overall evaluation when two signatures are collapsed. In general, the likelihood of the corpus will go down, because the stem and suffix probabilities in the combined signature will not fit the true probabilities of the words as well as two separate signatures could. For large corpora like the ones we are using, this likelihood drop can be quite large. In order to counterbalance it, there must be a large gain in the prior.

But now look at Table 2, which shows the effects of adding all the $y \rightarrow i$ rules to the grammar for the large corpus under the MDL prior. The first two lines give the number of signatures and stems in each grammar. The next line shows the total length (in bits) of each grammar, and this value is then broken down into three different components: the overhead caused by listing the signatures and their suffixes, the length of the stem list (not including the length required to specify exceptions to rules), and the length of the phonological component (including both rules and exception specifications). Finally, we have the negative log likelihood under each grammar and the total MDL cost (grammar plus likelihood).

As expected, the likelihood term for the grammar

[5]There are some additional complexities in the grammar encoding that we have not mentioned, due to the fact that stems can be recursively analyzed using shorter stems. These complexities are irrelevant to the points we wish to make here, but are described in detail in Goldsmith (2001).

	Init. Grammar	Change
# σs	1601	-7
# Stems	24386	-7
Grammar Size:	1249629	-318
σs, Suffixes	241465	-493
Stems	1005887	-111
Phonology	2277	+286
Likelihood:	6478764	+39
Total:	7728393	-279

Table 3: Effects of adding $y \rightarrow i$ rules under modified prior (large corpus).

with $y \rightarrow i$ rules has increased, indicating a drop in the probability of the corpus under this grammar. But notice that the total grammar size has also increased, leading to an overall evaluation that is worse than for the original grammar. There are two main reasons for this increase in grammar size. Initially, the more puzzling of the two is the fact that the number of bits required to list all the stems has increased, despite the fact that the number of stems has decreased due to reanalyzing some pairs of stems into single stems. It turns out that this effect is due to the encoding used for stems, which is simply a bitwise encoding of each character in the stem. This encoding means that longer stems require longer descriptions. When reanalysis requires shifting a character from a suffix onto the entire set of stems in a signature (as in $\{$*certif, empt, hurr*$\} \times \{$*ied, y*$\} \rightarrow \{$*certify, empty, hurry*$\} \times \{\epsilon$, *ed*$\}$), there can be a large gain in description length simply due to the extra characters in the stems. If the number of stems eliminated through reanalysis is high enough (as it is for the $e \rightarrow \epsilon$ rules), this stem length effect will be outweighed. But when only a few stems are eliminated relative to the number that get longer, the overall length of the stem list increases.

However, even without the stem list, the grammar with $y \rightarrow i$ rules would still be slightly longer than the grammar without them. In this case, the reason in that under our MDL prior, it is quite efficient to encode a signature and its suffixes. Therefore the grammar reduction caused by removing a few signatures is not enough to outweigh the increase caused by adding a few phonological rules.

Using these observations as a guideline, we redesigned our prior by assigning a fixed cost to each stem and increasing the overhead cost for signatures. The new overhead function is equal to the sum of the lengths of all the suffixes in the signature, times a constant factor. This function means there is more incentive to collapse two signatures that share several suffixes, such as $\langle$*e.ed.er.ing*$\rangle$/$\langle\epsilon$*.ed.er.ing*$\rangle$, than to collapse signatures sharing only a single suffix, such as $\langle$*ing.s*$\rangle$/$\langle\epsilon$*.ing*$\rangle$. This behavior is exactly what we want, since these shorter pairs are more likely to be accidental. Table 3 shows the effects of adding the $y \rightarrow i$ rules under this new prior. The starting grammar is somewhat different from the one in Table 2, because more rules have already been added by the time the $y \rightarrow i$ rules are considered. The important point, however, is that the cost of each component of the grammar changes in the direction we expect it to, and the total grammar cost is reduced enough to more than make up for the loss in likelihood.

With this new prior, our algorithm was more successful, learning from the large corpus the three major transformations for English ($e \rightarrow \epsilon$, $\epsilon \rightarrow e$, and $y \rightarrow i$) with a total of 22 contexts. Eight of these rules, such as $\epsilon \rightarrow e \ / \ Vxs\#$ and $y \rightarrow i \ / \ CyeC$, had no exceptions. Of the remaining rules, the exceptions to six of the rules were correctly analyzed stems (for example, *unhappy* + *ly* $\rightarrow$ *unhappily* and *necessary* + *ly* $\rightarrow$ *necessarily* but *sly* + *ly* $\rightarrow$ *slyly*), while the remaining eight rules contained misanalyzed exceptions (such as *overse* + *er* $\rightarrow$ *overseer*, which was listed as an exception to the rule $e \rightarrow \epsilon \ / \ CeeC$, rather than being reanalyzed as *oversee* + *er*). In the small corpus, no $y \rightarrow i$ rules were learned due to the fact that no similar signatures attesting to these rules were found.

Using these phonological rules, a total of 31 signatures in the small corpus and 57 signatures in the large corpus were collapsed, with subsequent reanalysis of 225 and 528 stems, respectively. This represents 7-10% of all the non-ϵ stems. The final grammars are summarized in the right half of Table 1.

6 Conclusion

The work described here is clearly preliminary with respect to learning phonological rules and using those rules to simplify an existing morphology. Our notion of context, for example, is somewhat impoverished; our system might benefit from using contexts with variable lengths and levels of generality, such as those in Albright and Hayes (2003). We also cannot handle transformations that require rule ordering or more than one-character changes. One reason we have not yet implemented these additions is the difficulty of designing a heuristic search that can handle the additional complexity required. We are therefore working toward implementing a more general search procedure that will allow us to explore a larger grammar space, allowing greater flex-

ibility with rules and contexts. Once some of these improvements have been implemented, we hope to explore the possibilities for learning in languages with richer morphology and phonology than English.

Our point in this paper, however, is not to present a fully general learner, but to emphasize that in a Bayesian system, the choice of prior can be crucial to the success of the learning task. Learning is a trade-off between finding an explanation that fits the current data (maximizing the likelihood) and maintaining the ability to generalize to new data (maximizing the prior). The MDL framework is a way to formalize this trade-off that is intuitively appealing and seems straightforward to implement, but we have shown that a simple MDL approach is not the best way to achieve our particular task. There are at least two reasons for this. First, the obvious encoding of stems actually penalizes the addition of certain types of phonological rules, even when adding these rules reduces the number of stems in the grammar. More importantly, the type of grammar we want to learn allows two different kinds of generalizations: the grouping of stems into signatures, and the addition of phonological rules. Simply specifying a method of encoding each type of generalization may not result in a linguistically appropriate trade-off during learning. In particular, we discovered that our MDL encoding for signatures was too efficient relative to the encoding for rules, leading the system to prefer *not* to add rules. Our large corpus size already puts a great deal of pressure on the system to keep signatures separate, since this leads to a better fit of the data. In order to learn most of the rules, we therefore had to significantly increase the cost of signatures.

We are not the first to note that with an MDL-style prior the choice of encoding makes a difference to the linguistic appropriateness of the resulting grammar. Chomsky himself (Chomsky, 1965) points out that the reason for using certain types of notation in grammar rules is to make clear the types of generalizations that lead to shorter grammars. However, our experience emphasizes the fact that very little is still known about how to choose appropriate encodings (or, more generally, priors). As researchers continue to attempt more sophisticated Bayesian learning tasks, they will encounter more interactions between different kinds of generalizations. As a result, the question of how to design a good prior will become increasingly important. Our primary goal for the future is therefore to investigate exactly what assumptions go into deciding whether a grammar is linguistically sound, and to determine how to specify those assumptions explicitly in a Bayesian prior.

Acknowledgements

The authors would like to thank Eugene Charniak and the anonymous reviewers for helpful comments. This work was supported by NSF grants 9870676 and 0085940, NIMH grant 1R0-IMH60922-01A2, and an NDSEG fellowship.

References

A. Albright and B. Hayes. 2003. Rules vs. analogy in english pass tenses: a computational/experimental study. *Cognition*, 90:119–161.

M. Brent and T. Cartwright. 1996. Distributional regularity and phonotactic constraints are useful for segmentation. *Cognition*, 61:93–125.

N. Chomsky. 1965. *Aspects of the Theory of Syntax*. MIT Press, Cambridge.

M. Creutz and K. Lagus. 2002. Unsupervised discovery of morphemes. In *Proceedings of the Workshop on Morphological and Phonological Learning at ACL '02*, pages 21–30.

C. de Marcken. 1996. *Unsupervised Language Acquisition*. Ph.D. thesis, Massachusetts Institute of Technology.

T. M. Ellison. 1993. *The Machine Learning of Phonological Structure*. Ph.D. thesis, University of Western Australia.

T. M. Ellison. 1994. The iterative learning of phonological constraints. *Computational Linguistics*, 20(3).

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

J. Goldsmith. 2004a. An algorithm for the unsupervised learning of morphology. Preliminary draft as of January 1.

J. Goldsmith. 2004b. Linguistica. Executable available at http://humanities.uchicago.edu/faculty/goldsmith/.

M. Johnson. 1984. A discovery procedure for certain phonological rules. In *Proceedings of COLING*.

M. Marcus, B. Santorini, and M. A. Marcinkiewicz. 93. Building a large annotated corpus of english: the penn treebank. *Computational Linguistics*, 19(2).

Rissanen. 1989. *Stochastic Complexity and Statistical Inquiry*. World Scientific Co., Singapore.