Natural Language Engineering **12** (3): 1–19. © 2006 Cambridge University Press
doi:10.1017/S1351324905004055 Printed in the United Kingdom

An algorithm for the unsupervised learning of morphology

JOHN GOLDSMITH
Departments of Linguistics and Computer Science, 1010 East 59th St., The University of Chicago, Chicago, IL 60637, USA
e-mail: goldsmith@uchicago.edu

(*Received 15 July 2004; revised 1 June 2005*)

Abstract

This paper describes in detail an algorithm for the unsupervised learning of natural language morphology, with emphasis on challenges that are encountered in languages typologically similar to European languages. It utilizes the Minimum Description Length analysis described in Goldsmith (2001), and has been implemented in software that is available for downloading and testing.

1 Scope of this paper

This paper describes in detail an algorithm used for the unsupervised learning of natural language morphology which works well for European languages and other languages in which the average number of morphemes per word is not too high. It has been implemented and tested in Linguistica, and is based on the theoretical principles described in Goldsmith (2001).[1] The executable for this program, and the source code as well, is available at http://linguistica.uchicago.edu.

Section 2 of this paper gives a brief overview of the theory that lies behind this work; sections 3 through 10 discuss the details of the algorithm in considerable detail. Section 11 presents an evaluation of the algorithm in an application to a corpus of English, and section 12 addresses briefly the theoretical implications of work on the unsupervised learning of linguistic structure more generally.

2 Brief overview of the theoretical framework

2.1 Introduction

The work described in this paper is part of a project aimed at the automatic learning of natural language morphology, as one aspect of the larger challenge of unsupervised learning of natural language grammar – a larger goal which may be

[1] I am grateful for comments, suggestions, and criticisms from Yu Hu, Irina Matveeva, Colin Sprague, Jeni Parham, and the anonymous reviewers of this journal.

unreachable in practical terms, but one which informs much of the work described here. We focus in the present work on morphological analysis based purely on distributional information, and in particular on the task of segmenting a word into distinct, successive morphs – rather than the assignment of morphosyntactic features, for example, which is the goal of many other morphological parsers under development today. Our goal is partly a practical one: good morphological parsers for many of the world's languages would be useful for a number of functions, ranging from document retrieval to automatic machine translation, all of which would arguably be superior if trained from a corpus in which words were morphologically segmented. At the same time, the goal has a theoretical side, because we seek to understand how much prior (or "innate") knowledge needs to be given to the morphological induction device in order to be able to find an analysis that matches up well to linguists' considered opinions regarding linguistic structure.

Goldsmith (2001) proposes a natural division of the process of morphology discovery into a set of heuristics, on the one hand, and an MDL (Minimum Description Length) evaluation process, on the other. The heuristics, in turn, divide naturally into an initial bootstrapping heuristic that is able to determine a first pass analysis of the words of the corpus into stem and affix, and a set of incremental heuristics, which modify the analysis, leaving the decision to the MDL component as to whether the modifications are worth maintaining or should be dropped. The exposition in this paper follows that division; it begins with a discussion of the MDL analysis, and continues with a discussion of the bootstrapping heuristic and the individual incremental heuristics. This is followed by a discussion of how to evaluate the performance of the approach, and some final concluding observations.[2]

2.2 Minimum Description Length (MDL)

Minimum Description Length (MDL) analysis (see Rissanen (1989); see also Wallace and Georgeff (1983) and Wallace and Dowe (1999) for a related approach, Minimum Message Length) is a form of analysis rigorously based on information theory. Given a corpus, an MDL model defines a *description length* of the corpus, given a probabilistic model of the corpus: the description length is the sum of the most compact statement of the model expressible in some universal language of algorithms (by which one would mean a program for a universal Turing machine), plus the length of the optimal compression of the corpus, when we use the probabilistic model to compress the data (see (1)). The length of the optimal compression of the corpus is the base 2 logarithm of the reciprocal of the probability assigned to

[2] There is no natural home in the analysis presented in this paper for the distinction between inflectional and derivational morphology. This question is addressed, however, in Goldsmith and Hu (2005), in which an analysis of the distinction is offered in terms of the geometry of a finite state automaton for the morphology.

the corpus by the model; we return to this notion (a standard one in information theory) below. Since we are concerned with morphological analysis, I will henceforth use the more specific term the *morphology* rather than *model*; one can read the M in (1) as referring specifically to a morphology.

(1) $$DescriptionLength(CorpusC, ModelM) = length(M) + \log_2 \frac{1}{prob(C|M)}$$

MDL analysis proposes that the morphology M which minimizes the objective function in (1) is the best morphology of the corpus. Intuitively, the first term (the length of the model, in bits) expresses the *conciseness* of the morphology, giving us strong motivation to find the simplest morphology possible, while the second term expresses how *well* the model describes the corpus in question. The morphology M spreads probability mass over a wide universe of possible words (by assigning a probability to all possible words in the language, and by being subject to the requirement that the probabilities sum to 1), and we want one that assigns as much of it as possible to the words of the particular corpus which we happen to be looking at. Instead of considering the probability of the corpus, we consider the log of the reciprocal of that probability, because this is a quantity which is expressible in information theoretic *bits*, and which can then be added to the first term in (1); that is, by multiplying the log probability of the corpus by −1, we can reasonably add the two terms and attempt to find the analysis which *minimizes* the sum of the two terms. Hence the term: *minimum* description length.

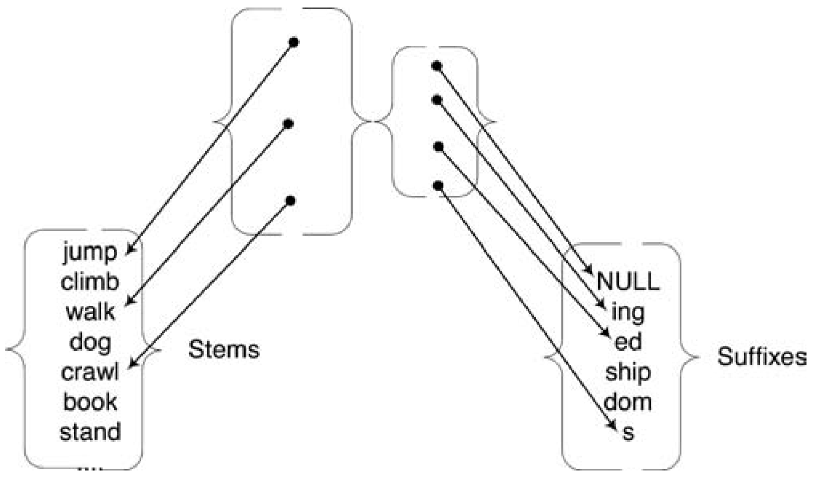
Thus we need to design a morphology M which assigns a distribution D over words such that the observed words in the corpus lie in the support of D (the set to which D assigns non-zero probability), and we need to do this in a way which allows us to easily calculate the length of M.

2.3 Calculating the length of the morphology

And what *is* the information that composes the morphology of a language such as English? Most of the information is to be found in the phonological (or logographic) content of the morphemes, but some of the information is contained in information regarding the ordering of possible morphemes in the language. We condense all of this information into essentially three components of the morphology: a list of stems, a list of affixes, and a list of *signatures*, which are structures indicating which stems may appear with which affixes. A signature can be visualized as in (2a), or better as in (2b).

(2)a $$\left\{\begin{matrix} crawl \\ jump \\ walk \end{matrix}\right\}\left\{\begin{matrix} NULL \\ ed \\ ing \\ s \end{matrix}\right\}$$

(2)b



As the structure in (2b) suggests, the role of pointers in the construction of the formal morphology is critical. We must ask precisely how *long* a pointer is in such a diagram, and we must get an answer to that question expressed in units of bits. Information theory provides an answer to this question, or rather, to the question: what is the *shortest* encoding system that we can set up for pointers in such a situation, measured in bits? The answer is the base 2 logarithm of the reciprocal of the frequency of the item being pointed to, or $-\log_2 freq(\bullet)$ (see, e.g., Bell *et al.* (1990)). Intuitively, this means that up to this limit, we can find an encoding that allows frequently used items to be more easily accessed, if by "easily" we mean pointed to in a more concise fashion. It is possible to quite literally encode a pointer to an object X by a string of binary digits no larger than 1 greater than the base 2 logarithm of the reciprocal of the frequency of X, and thus this quantity is often referred to simply as the "length of the pointer" to X. When we speak of the "length" of a pointer, then, one may paraphrase that as the length of the optimal encoding of the pointer.

2.4 Calculating the probability of the corpus, using the morphology

As we noted above, we must also compute the probability of the corpus – or rather, in order to derive a unit in the natural unit of bits, we compute the base 2 logarithm of the reciprocal of the probability of the corpus. The log probability of a given word w, analyzed as belonging to signature σ with stem t and suffix f, is as given in (3); the probability of the corpus, as given in (4).

(3) $$\log prob(w) = \log prob(\sigma) + \log prob(t|\sigma) + \log prob(f|\sigma)$$

(4) $$\log_2 \frac{1}{prob(Corpus)} = -\sum_{w \in Corpus} \log prob(w)$$

3 Searching for the right morphology: begin by bootstrapping

We have framed the problem of morphology learning, given a specific corpus, as one of finding a morphology which leads to a minimum value of the expression in (1). In order to be able to find this morphology in a reasonable period of time, it is most effective to have a bootstrapping algorithm that gets us reasonably close to the correct morphology quickly. After that, we can use alternative methods to modify the morphology in relatively small steps, that is, to modify it plank by plank to achieve the lowest possible description length.

The bootstrapping algorithm that we have found to be most effective in dealing with a language such as English or French consists of two parts, the first based loosely on a method proposed by Zellig Harris (1955, 1967), utilizing his notion of *successor frequency* (and see now Xanthos (2003)), and the second consisting of identifying successful *signatures* from among the cuts proposed by the first part.[3]

3.1 Successor frequency

Harris originally made a proposal as a *solution* to the general problem of morpheme discovery, despite the fact that even early implementations of it established quite clearly that it was not adequate for that end (Hafer and Weiss (1974)).

Harris used the term *successor frequency* in the following way: after the n^{th} letter in a word W, in a given corpus, the successor frequency is the number of distinct *letters* that appear immediately after the string prefix defined by $W[1\ldots n]$, the first n letters of W, where space and punctuation count as a single, distinct letter. For example, after the string prefix "*gover*" in most English corpora, only one letter will be found to follow, and that letter is "*n*"; hence the successor frequency of the position after *gover* is 1, while in a particular corpus, the successor frequency after the string prefix *govern* is 6, if we find the words *governed*, *governing*, *government*, *governor*, *governs*, and *govern* in that corpus.

Harris' basic insight can be easily implemented if we organize the words of a lexicon in the form of the data structure called a *trie* – a tree in which there is a node for each shared string prefix, in the following sense. Each node in the trie is responsible for a set of strings which share a common string prefix, and each node contains a (distinct) pointer to a (distinct) node for each letter which immediately follows the common string prefix in the set of strings that the node is responsible for. Thus the root node is responsible for all the strings, and has a pointer to a node for each letter that begins at least one word in the string set. If "*a*" is such a letter, then there is a node "*a**" which responsible for (and which we may think of as dominating) all strings beginning with "*a*", and if there is more than one string in that set, then that node has pointers to nodes corresponding to each of the letters which appear immediately after word-initial "*a*", etc. Such data structures are widely used today, and will be familiar to most readers of this paper, especially

[3] An extension of this heuristic is discussed in Hu and Goldsmith (2005).

in the context of a Patricia trie, a trie in which all nodes with only unary branching beneath them are merged with their daughters.

Harris proposed that peaks in successor frequency would be suitable detectors for the discovery of morpheme breaks. As Hafer and Weiss (1974) note, Harris's apparent proposal is actually a family of closely related proposals, and none of them work anywhere close to perfectly, for various reasons, some of which we will review here. There are a number of parameters that one can modify in the actual implementation of Harris's suggestion, and we adopt a set of parameters that increases the precision, while decreasing its recall. In short, we adjust Harris' proposal so that it is makes fewer analytical claims about the words, but those that it makes are relatively trustworthy. We do this in the following way.

Looking at peaks in the successor frequency in the first three letters of a word tends to give rise to a large number of spurious peaks, in the sense that the peaks do not signal morpheme boundaries. Since there are more consonants than vowels, and since vowels tend to follow consonants, just as consonants tend to follow vowels, there is a strong tendency for the successor frequency to be larger after a vowel than after a consonant within the first three letters of a word, and hence for this algorithm to find a (spurious) morpheme break after any vowel in the first 3 letters of a word. Since we are at this point looking for "stem-suffix" breaks, we restrict our attention to candidate stems that are at least three letters in length, recognizing that there are some shorter stems (e.g., *be*) which will only be discovered at a later point.

We actually place a more stringent requirement on the cuts motivated by a peak in successor frequency at this point: we require that to make a cut after the *i*th letter, the successor frequency must be exactly 1 after both the $i\text{–}1$ th letter and the $i+1$ th letter. This decision is a conservative one, in the following sense. The two most common reasons to find a successor frequency greater than 1 in two successive positions are these: either both peaks are accurate indicators of morpheme breaks, and the first morpheme is one letter long (for example, with the words *petit*, *petits*, *petite*, *petites*, a successor frequency of 3 is found after *petit* and a successor frequency of 2 is found after *petite*), or a morpheme break is found after the first position, and two of the suffixes that occur begin with the same letter (e.g., many stems are followed by both *–ing* and *–ion*, in addition to *–ed* and *–s*). It is difficult to be certain which is the correct cut at this point; by putting this condition on the bootstrapping heuristic, no cut is made for the *–ing* and *–ion* words at this point. The algorithm will very soon have considerable knowledge about the morphology of the language, and it will know that *–ing* and *–ion* are common suffixes, but that *–ng* and *–on* are not common suffixes, so it will be able to make a much more informed choice than it can right now.

3.2 Organization into signatures

After finding appropriate cuts of some of the words into two pieces, we treat the first piece as a stem and the second as a suffix, and for each stem, we organize the entire set of suffixes with which it appears in the corpus as an alphabetized list: a *suffix-list*.

We then create a list of such suffix-lists, and associate with each such list the set of stems that appears with precisely that set of suffixes. This association is exactly a signature, as described earlier in this paper, as in (2). Each stem is associated with exactly one signature. Common signatures in English include *NULL.s* (primarily nouns), *NULL.ed.ing.s* (verbs), and *NULL.er.est.ly* (adjectives).

We then apply a set of filters in order to eliminate certain implausible signatures, because our goal in this first heuristic is to prefer precision over recall, in the sense that we would rather fail to uncover some morphological structure than detect spurious or false structure. We set a threshold (of 3) for the minimum number of words an affix may appear in; a hypothetical suffix occurring less often than that is eliminated.

The second heuristic we use to eliminate signatures is based on an apriori probability of the length of a suffix being just one letter in length. NULL is a likely affix in general (in the sense that languages often build words with no overt affixes), but suffixes with only one letter (phoneme) are both rare and suspect. Even if we did not know English, we would be wise to be suspicious of a morphological analysis which posits a stem *car* that can be followed by the affixes NULL, *e*, *t*, *p*, *b*, and *d*. These are really distinct stems (in English: *car*, *care*, *cart*, *carp*, and *card*). As noted by Brent (1999), natural languages do act as if they select their morphemes with an eye to keeping their mean length to the neighborhood of 5, with the average less for affixes than for stems, but with a relatively low probability of morphemes of length 1. To be sure, *NULL.s* is the most common signature in English, French, and Spanish, so we can take this length consideration only as a tendency, and be willing to accept a signature such as *NULL.s* as legitimate if it is found in association with a sufficient number of examples.

A certain amount of experimentation has led us to the following heuristic.[4] Any signature with a large number of stems (defined as 25) is permitted, while those with fewer are subject to the following test. A signature must have at least two affixes that are of length at least 2 (where a NULL affix is considered to be of length 2 for these purposes); otherwise it is dropped. Thus by this latter criterion, *NULL.t*, or *b.p*, would be eliminated, while *br.tr* and *NULL.br* would be accepted.

What is the connection between finding signatures and MDL? Each signature represents a considerable savings in the number of letters that are needed in the stem lists. We may think of the null morphology as being the morphology in which there are no affixes, and the only structure present is that the words of the corpus are each individually represented in the list of stems. When we are able to reduce a set of t stems and f affixes to a description as a signature (and such a signature represents t times f words altogether), we are able to save $f–1$ copies of each stem, and $t–1$ copies of each affix. If the average length of a stem is S and the average length of an affix is F, the signature will save approximately $\log_2(27)[S(F-1)+F(S-1)]$, measured in bits – which is a considerable amount of savings in practice. In an

[4] The value chosen for this parameter has been chosen somewhat arbitrarily, and here, as at a few other places in this paper, experiments with a large number of gold standards for different languages might lead to somewhat different optimal settings.

intuitively straightforward sense, the quantity $[S(F-1)+F(S-1)]$ is the number of letters saved by the use of the signature, and we may refer to this as the *robustness* of the signature. In presentation of a language's signatures to the user, this quantity is used to sort the signatures, with the signature with greatest robustness ordered first.

4 Check signatures

The *Check signatures* function directly incorporates the insights of the Minimum Description Length perspective on grammar induction. It examines each signature in turn, and attempts to determine if the transfer of material (letters, phonemes) from stem to suffix will improve the overall description length of the morphology. For example, if there is a large set of words ending in *–ion* and *–ive*, the function described in the prior section will draw the conclusion that there are suffixes *–on* and *–ve* in these cases, and place the *–i* in the stems, not in the suffixes. The purpose of this function is to identify and correct that error.

Now, each signature consists of a list of pointers to stems, and pointers to suffixes, and in most cases, there are more stems than there are suffixes in a signature. When we examine a signature, we typically expect a healthy variety of different final letters: while there may be a skew in the distribution of letters that may appear stem-finally, there should nonetheless be a good variety. *Check signatures* computes the entropy of the set of stem-final letters. If that entropy is greater than the threshold value (experimentally set at 1.4), the function returns, performing no change. If the entropy is less than the threshold amount, it considers the entropy of the set of stem-final bigrams, and performs the same check for measure against the entropy threshold. The function successively considers the entropy of stem-final strings of up to 4 characters, and determines what the largest k is for which the set of k-long stem-final letters has an entropy less than the threshold.[5]

It then considers each of these restructurings of the signature, and calculates an approximation of the change in the morphology's description length brought about by the change in the cuts between stem and suffix that would be caused by shifting a certain amount of stem-final material to the beginning of the suffixes, such as the –i– alluded to above.

The first step is to calculate how much length the signature σ is responsible for in the overall morphology – so that we can compare that length to the length of the alternative signatures which attempt to handle the same data. Now, a signature is composed essentially of the following: a list of pointers to stems, and a list

[5] It should be clear that this strategy is just a heuristic, and a more complex heuristic may prove worthwhile in more complex cases. Testing the entropy of the last k letters of the stems is a rough test as to whether we have wrongly cut up one or a small number of suffixes between the stem and the affix, but it works well in practice.

of pointers to suffixes. From here on out, it will be convenient to have a good notation to indicate the frequency of a word or morpheme in the corpus, and we shall henceforth indicate it as square brackets; thus, [*the*] indicates the number of occurrences of the morpheme *the* in the corpus in question. As we have seen, the length of (the encoding of) a pointer to C (where C is an element in a list labeled by the category A) is $\log_2 \frac{[A]}{[C]}$, so the length of the pointers to stems is the sum of the inverse log frequencies of the stems, and in a parallel fashion, the length of the pointers to suffixes is the sum of the inverse log frequencies of the suffixes, though there is a difference in that a suffix will typically be associated with several signatures.

Indeed, a suffix which is associated with only a single signature is a bit suspect; being able to reanalyze a signature (such as *on.ve*) so that it is replaced by a signature that consists only of suffixes that "already" and "independently" exist is a good thing, as it decreases the description length of the morphology by increased use of a smaller inventory of parts. In order to be able to keep track of the possibility of making such a move, when we calculate the bit-length (information content) of a signature, we assign to it a portion of the information content of the suffix entry itself that is proportional to the relative use made of the suffix by that signature. For example, if signature σ is the only signature to use the suffix *on*, and storage of the suffix *on* takes 9.2 bits, then signature σ is charged the full 9.2 bits at this point, in addition to the length of the pointer to *on* which the signature needs in order to do its work. If, however, there was another signature σ' which used the suffix *on* to cover an equal number of tokens in the corpus, then signature σ would only be responsible for 9.2/2 (= 4.6) bits in the present calculation.

In sum, we can thus calculate an approximation of the description length (DL) of an individual signature σ, and we have a resulting value in bits. This DL consists of three terms: the sum of the lengths of the pointers to stems, the sum of the lengths of the pointers to suffixes, and the *partial* responsibility of each signature for the information content of the suffix entries of the suffixes it uses. However, since under most circumstances the lengths of the pointers to the stems will not change when we restructure the signature, we leave this consideration out of the calculation at this point. As we shall see in a moment, there are cases where the length of the pointer to the stem changes (because the stem "already" existed), and we shall then calculate the difference directly. We expect that each stem associated with a signature will contain a pointer to its signature, so we also include the cost of all of these pointers, which is equal to the number of stems $[Stems(\sigma)]$ times the log frequency of the signature. As we will see, under certain conditions, each of these may vary in the modified forms of the signature.

We now move on to consider alternative signatures to σ' which will be constructed by shifting increasingly long sections of material from the stems to the suffixes, stopping when the entropy of the set of transferred material exceeds the threshold. For example, a set of stems ending in *a* and in *i* might be associated with the signature *–ble* by the Successor Frequency function, and the present algorithm would calculate the total description length of the two signatures that would be created by shifting all stem-final *a*'s to form a suffix *–able*, and all of the stem-final *–i*'s to form a suffix *–ible*, meanwhile shortening all of the stems by one letter.[6]

The description length of one of these alternative signatures is calculated as follows: to determine whether the restructuring is preferable, we must total each of the description lengths, and compare them to the original description length, opting for the situation in which the description length is the least.

Consider first the length of the pointers to stems. Since by design, each stem T is associated with exactly one signature, these numbers will not generally change when we restructure the signature–whether the stem is *positi-* or *posit-* will not change the number of occurrences of *position* and *positive* in the overall corpus; but as this example suggests, the removal of a portion of material from stem T (in this case, the material i) may well give rise to a "new" stem T' which independently occurs elsewhere in the corpus (for example, as an unanalyzed word). Indeed, that discovery should speak in *favor* of this reanalysis, for the stem *posit* is being used more often. Restructuring the entire morphology in order to calculate the overall effects of this change would be the most accurate way to proceed; however, we accept a simplification, and merely decrease the length of the stem-pointer in the signature by increasing the frequency of the stem in question: it becomes the sum of the number of occurrences of the stem T in the present signature σ, plus the number of occurrences of the stem T' in its other signature or its unanalyzed occurrences. Thus the length of the pointer to T will shift from $\log \frac{[W]}{[T]}$ to $\log \frac{[W]}{[T]+[T']}$ (where [W] is the total number of words in the corpus), a difference equal to $\log(1 + \frac{[T']}{[T]})$ (the reader may recall that $\log(1+x)$ is approximately $x - x^2/2 + x^3/3$ for small x), and similarly change in the length of the pointer to T' will be equal to $\log(1 + \frac{[T]}{[T']})$. Furthermore, the stem T is now entirely removable from the list of stems, and therefore an additional savings equal to approximately $|T| * \log(27)$ occurs, which is likely to be a considerably larger amount.

Even when a new stem is created which did not exist before (e.g., *posit-* instead of *positi-*), if it is shorter, then the amount of information in the stem list decreases; hence if the number of stems associated with signature σ is $[Stems(\sigma)]$, and a final string of length k is removed from them, there is a total savings of approximately $[Stems(\sigma)] * k * \log(27)$ bits associated with the new signature.

And what of the list of suffixes in this new signature? In the first place, it is possible that this list of suffixes already exists in the morphology as an independently needed signature $\sigma*$, and if that is the case, then a considerable simplification can be achieved by simply merging the signature σ with $\sigma*$. Let us construct a list of all the places in the morphology where this merger will give rise to a simplification. First, the length of a pointer to $\sigma*$ will shift from $\log \frac{[W]}{[\sigma*]}$ to $\log \frac{[W]}{[\sigma*]+[\sigma]}$, and that difference is $\log(1 + \frac{[\sigma]}{[\sigma*]})$; in parallel fashion, the length of the pointers to σ will change by an amount equal to $\log(1 + \frac{[\sigma*]}{[\sigma]})$. Using the approximation mentioned above, we see

[6] As this example illustrates, it may be that the best analysis would be one where one of these letters was transferred to the suffix, and the other was not. This possibility is not currently considered by the algorithm.

that this means a savings of about $\frac{[\sigma*]}{[\sigma]})$ bits for every place where the signature σ was mentioned in the grammar previously. Thus the savings are considerable when a signature is replaced, or superseded, by a signature which occurs considerably more times. And these savings will indeed accrue quite a few times, for there are many places in the grammar where pointers to signatures occur: minimally, there is a pointer to a signature associated with each stem. The *savings* to $\sigma*$ that occur when signature σ can be replaced by an already existing signature $\sigma*$ due to the collapsing procedure alone are thus $[Stems(\sigma*)] * \log(1 + \frac{[\sigma]}{[\sigma*]})$, while savings of the cost of the pointers to $\sigma*$ from the stems of σ is equal to $[Stems(\sigma*)] * \log(1 + \frac{[\sigma*]}{[\sigma]})$. One is tempted to see this as a quantitative evaluation of Meillet's classic dictum that a language is a *système où tout se tient*.

If the new signature $\sigma*$ did not independently occur, we must calculate the relevant parts of its description length: the length of its pointers to its individual suffixes. The length of the pointer to suffix f is $\log \frac{[W]}{[f]}$. We continue, as we noted above, to prorate the information content of the actual phonological material of the suffix between this new signature and all the other signatures that also point to this suffix. The more signatures point to the suffix, the less any of them will have to be responsible for that suffix's phonological content.

5 Extending known stems to known suffixes

One of the conditions that we placed on the successor frequency bootstrapping algorithm blocked it from associating a stem with a particular suffix if there were two or more suffixes that began with the same letter (e.g., *conservation* and *conservative* could not be analyzed as *conserv-ation* and *conserv-ative*, even in the presence of *conserve* and *conserving*). We now make up for this initial conservatism, by scanning through our list of discovered stems and looking to see if there are any unanalyzed words which consist of such a stem followed by a suffix that had been discovered elsewhere. When such words are found, they are analyzed and divided into *stem* and *suffix*. If there should be two such ways found, the one with the more common stem is preferred.

6 Extending known signatures

We now consider all signatures containing at least two stems and two suffixes, and scan through the words unanalyzed so far, to see if they fall into any such signatures. We sort the signatures by robustness, and look for the most robust signatures first. When we find that a signature matches a set of words, we analyze the words into stem and affix with that signature. One of the consequences of this is that we now can find stems whose length is shorter than the limit we placed on stems in the initial boostrap heuristic, because our knowledge of the morphological patterns is now greater.

7 Extending known stems

As we pass through each successive function, our formal analysis of the morphology of the corpus has improved. We now consider whether the stems which we have analyzed up to now can serve as a means to finding new suffixes.

We consider each stem t (optionally setting a lower bound on stem length), and consider the signature σ_t that it is currently associated with. The robustness of σ_t (as defined above) is a rough MDL-based measure of how good a signature σ_t is, and we set an empirical threshold of 10 for σ_t's robustness. If the stem t passes this test, we consider all words that begin with t but whose continuation is not in σ_t, and we put all of these continuations into a tentative suffix collection. When we have considered all of the stems, we eliminate from the tentative collection any suffix which has occurred fewer than 3 times, and accept all other suffixes, integrating them into their new stems' signatures.

8 Extending known suffixes ("loose fit")

Our knowledge of morphology is encoded in our knowledge of signatures, stems, and affixes, and the degree of secure knowledge in each of these is greatest for signatures, and least for affixes. In the preceding functions, we have leveraged our knowledge of stems and of signatures to deepen the analysis; now we use our knowledge of suffixes to consider the possibility of finding both new stems and new signatures.

We look at all words which have not yet been morphologically analyzed, and find all divisions of such a word w into two pieces $t{+}f$ such that the second piece is a known suffix f. For such a case, we consider *all* words that begin with t that have not yet been analyzed, and tentatively analyze them as being based on a stem t associated with a signature which is composed of all of these observed continuations. These constitute the new hypothetical signatures which we now analyze.

For each such signature σ, if it is already a recognized signature (which rarely happens), we accept the new stem introduced into to the discussion in the preceding paragraph. In all of the other cases, we calculate the effect that its inclusion would have on the description length. If that effect would be salutary – that is, it would decrease the total description length – then we accept the new signature and the new word-divisions that it indicates; if not, we reject it and its word-divisions. We can rapidly approximate the effect on description length in the following terms: as above, the size of the set of words in the morphology is $[W]$; then the cost of the current analysis of a particular word w (i.e., no analysis at all) is equal to the "graphological information" of the entire word (see above: $\log_2(27)$ times length of each word), plus one pointer from the null signature to this word (of length $\log \frac{[W]}{[w]}$) plus one pointer from the word to the null signature (of length $\log \frac{[W]}{[NullSignature]}$).[7] The cost of the new analysis, with the new signature, is equal to the sum of three terms: a

[7] In this paper, we have simplified things slightly by assuming each observed word occurs once (so that stem frequencies can be derived from number of different affixes they are observed with); in general, this is a simplifying assumption that does not need to be made, but if it is, then $[w]$ here is equal to 1.

stem term, a suffix term, and a pointer to the signature itself. The stem term is the sum, over all of the stems in the signature, of the graphological information of the stem and a pointer to that stem; the suffix term is the sum, over all of the suffixes in the signature, of the graphological information of the suffix if the suffix did not already exist, and a pointer to that suffix. If the cost of the new analysis is less than the cost of the current analysis, we select the new analysis and its concomitant word-analyses.

Approximate information content of an analysis, where F is the set of suffixes which already existed in the morphology, and G is the set which did not:

$$(5) \qquad |t| \log_2 27 + \sum_{f \in F \bigcup G} \log \frac{1}{freq(f)} + \sum_{f \in G} \left(|f| \log_2 27 + \log_2 \frac{[W]}{[f]} \right)$$

Following this, we use the MDL-based check-signatures function (see section 4).

9 Finding singleton signatures

We have known since Zipf that a high proportion of words found in a corpus have very low frequency, and the same is true of stems. Given a word formed by a stem that occurs only once in a corpus, it is not always easy to know whether the word is morphologically analyzed, and if it is, which is the correct analysis. Given the word *pringles*, is it to be analyzed as *pringle* + *s*? Given the word *framness* should it be analyzed at all, and if so, is the suffix *–ness*, *–s*, or something in between?

Here we have to do the best we can with considerable less certainty. Our algorithm considers each unanalyzed word which ends in a known suffix (including the null suffix) and assigns it a probability based on a model that assumes that a stem is chosen based solely on its length, and then a suffix is chosen based on the suffix's frequency. We must, therefore, compute the probability of each stem length, using the frequency of known stem-lengths in the stem collection, and taking the unanalyzed words to be members of the set of stems that have null suffixes.

10 Allomorphy

Determining the correct segmentation of an arbitrary word is only the first step in analyzing the morphology of a language: in addition, virtually all languages display allomorphy, that is, variation in the realized form of a given morpheme. In English, the same morpheme appears as *love* (in the word *love*, *lovesick*, and *loves*) and *lov* (in the words *loving*, *lover*, and *loved*). More generally, word-final *–e* in English deletes before a range of suffixes, including *–ed*, *–ing*, and *–ity*. Suffixes too take different forms: the plural *–s* in English appears as *–es* after stems that end in *s*, *sh*, or *ch* (*hisses*, *masses*, *hitches*, etc.). It is often not obvious to the analyst or to the native speaker just where this allomorphy begins and ends (a point we discuss in greater detail in section 11 below). For example, it is reasonable to assert that the stems *receive* and *recept* (as in *recept-ion*) are alternate realizations (that is, allomorphs) of the same morpheme, paralleled by *deceive/decept(ion)*, *perceive/percept(ion)* and *conceive/concept(ion)*, but it is less clear whether the correct form is *recep* or *recept*.

And other conceivably related forms are not in fact related at all: for example, the stems *resolut-* and *revolut-* (from the words *resolution/revolution*) are not related by any rule relating *s* and *v* in English.

In this section, we present an algorithm that takes certain steps towards dealing with this challenge. At present, the algorithm to be presented is capable only of detecting rules of allomorphy that delete stem-final material, like the deletion of word-final *–e* in English, and rules that cause alternations of a stem-final letter (e.g., *y* becomes *i*) before certain suffixes (e.g., *–es*). This capability is useful, however, for a range of languages, including English. Considerable work remains before the range of actual alternations can be automatically detected; some further examples are discussed in the next section.

Let us step back and think about this problem more generally. The task of finding the principles that relate the forms (allomorphs) of a stem is generally conceived of as the task of discovering the phonology of a language, a problem that has been attacked by a number of researchers, especially in the past ten years (Ellison (1991), Albright (2002), Albright and Hayes (2003), Neuvel and Fulop (2002), and others.) Most, but not all, of this work has assumed that some "oracle" – some outside source of information – provides the phonology learner with the information that two words are morphologically related: the two words may be explicitly marked as being part of the same morphological paradigm, for example. But the present algorithm does not have access to that information, by the ground-rules that we have set for it.

The most reliable information the framework has is the set of robust signatures in the language, and it is this information that it uses to determine if there is stem-final deletion at play in the language it is considering. Suppose there is a suffix F which deletes stem-final L, and suffix F appears with stems that appear with a null suffix. (For example, F might be the suffix *–ing* in English, and L the letter *–e*.) Then there will appear to be two distinct signatures in the language: *NULL.F* (from "regular" stems that do not end in L) and *L.F* (from stems that end in L). In addition, under these phonological conditions, the morphology may have wrongly analyzed some cases of stem-final L's as having been part of a larger suffix.

Since signatures are the most reliable distributional information we have about the language at this point, we use them to detect this situation in the following way. We consider all 1-letter suffixes, and for each one (which we will mnemonically enough call '*e*', in honor of one of the suffixes that actually passes the tests we are about to describe), we wish to establish three classes of suffixes:

1. suffixes that *might* delete *e* (e.g., *–ing* deletes stem-final *e*);
2. suffixes that were erroneously given an initial *e* (e.g., if a suffix *ement* were established for English, which should actually be *ment* with a stem-final *e*), and
3. suffixes (e.g., *–d*) which were erroneously analyzed, because they are actually of the form *eX* (e.g., *–ed*) and delete stem-final *e*.

In cases (1) and (3), we indicate that a suffix f deletes a preceding *e* with this notation: *<e>f*. For example, if *ing* deletes a preceding *e*, we henceforth represent the suffix as *<e>ing* rather than as simply *ing*.

We identify a *potentially deleting* suffix f_{del}, a member of our class 1, by finding pairs of signatures, σ_1 and σ_2, where σ_1 is of the form $NULL.f_{del}$, (for example, $NULL.ing$) and σ_2 is of the form $e.f_{del}$ (for example, *e.ing*), with the further condition that the number of stems in the signature $NULL.f_{del}$ exceed those in the signature $e.f_{del}.$: that is, if a suffix (such as *–ing*) is a deleter, it is still the case that its non-deleting occurrences are more frequent than its deleting occurrences. We identify those in class 2 (like the spurious suffix *ement* in English; call such a suffix f_L) if f_L is of the form eX (i.e., it begins with *e*), and X is also an existing suffix, and X occurs with more stems than eX ($= f_L$) does: that is, the correctly analyzed suffix is more frequent than the misanalyzed one. We identify those in class 3 (call one d; e.g., *–d* in English) if the concatenation *ed* also exists (that is, the concatenation of the '*e*' that we are exploring), and the number of stems that occurs with *ed* is greater than the number of stems that occur with d – again, the non-deleting case must be more frequent that the deleting case.

In each of these three cases, we associated with each suffix f a modified form of the suffix, which we will refer to as *T(f)*. T can be thought of as a mapping to a more abstract morphophonological representation. In the case of a suffix in Class 1, such as *ing*, $T(ing) =$ *<e>ing*; for one in Class 2, such as *ement*, $T(ement) = ment$; for one in Class 3, such as *d*, $T(d) =$*<e>ed*. *T(e)* is defined as the f suffix. For any suffix x that has not been assigned a modified form by the three methods defined in this paragraph, we define *T(x)* as *x*. The map T can then be taken to apply to concatenations of suffixes by applying to each of the suffixes individually. For example, $T(d.ing) =$*<e>ed.<e>ing*. We will also need an operation that maps these more complex signatures to a simpler representation, in which the deleting elements *<x>* are suppressed; we refer to this as [ ]∗: e.g., [*<e>ed.<e>ing*]∗ = *ed.ing*.

The preceding identifications are preliminary to the next step, which is crucially linked to description length. The computational motivation for discovering these more complex relationships between various pairs of stems is that by recognizing these relationships, we can decrease the total number of signatures, which in turn significantly reduces the description length of the morphology and of the corpus. In fact, we can say that the push to discovery allomorphy is motivated by the desire to extend the reach of a small number of signatures.

We now consider all signatures σ that contain suffix L such that $[T(\sigma)]*$ also exists. For example, *e.d.ing* is such a signature, because of (6a); hence (6b) follows.

(6a) $$T(e.d.ing) = NULL.{<}e{>}ed.{<}e{>}ing$$

(6b) $$[T(e.d.ing)]* = NULL.ed.ing$$

In addition, the signature *NULL.ed.ing* independently exists in English. We count the number of distinct signatures that satisfy this property, and count the total number of stems that are associated with these signatures, setting thresholds of 5 and 50, respectively. A suffix L that passes this test is interpreted as being erroneously analyzed as a suffix, and is reintegrated into preceding stems; suffixes are reassigned according to the function T, as defined above.

A similar method is used to identify a stem-final segment that mutates under the influence of a following suffix (for example, stem-final y mutates to i before suffixes such as *–al*: $bury + al > burial, beauty + ful > beautiful, dry + ed > dried$). For each 1-letter suffix Y, we do the following. For each suffix Z that occurs with Y in a signature $Y.Z$ (e.g., Z might be the suffix *ies* in the signature *y.ies*), we look to see if Z can be decomposed into IZ^*, where I is a single letter, and Z^* is an existing suffix (in the cases just mentioned, the Z^* would be *–al*, *–ful*, or *–ed*). If that condition is met, we define $T(Z)$ as the ordered pair (I, Z^*), written a bit more perspicuously as $\{Y|I\}Z*$. Intuitively, $\{Y|I\}Z*$ means a fixed suffix Z^* which has the property of mutating an immediately preceding Y to I, much as *<e>ing* refers to a fixed suffix *ing* which mutates a preceding e to the null string. Note that the letter identified as I can be distinct in the case of each suffix; it is merely the first letter of the suffix. If, however, there exists a common letter (call it I) that is shared by a majority of the suffixes, then we conclude that all suffixes Z such that $T(Z) = (I, Z*)$ are indeed of the form Z^* (e.g., *–ial* is reanalyzed as *–al*) with the property that they mutate a stem-final Y to I. The corresponding stems are then modified, so that they take on a final $-Y$.

These procedures illustrate how the drive for simplification of signatures can lead to the discovery of simple patterns of allomorphy.

11 Evaluation

As is the case with most natural language efforts, a quantitative evaluation of the accuracy of morphological analysis of English is fraught with issues that were initially unexpected. We built by hand a gold standard of some 15,000 words and the target morphological analysis we expected. This turned out to be a much greater challenge than we expected, and we will explain why in what follows.[8]

We decided to evaluate with an accuracy measure, rather than with precision and recall as in Goldsmith (2001). This was based on a practical consideration: in a certain sense, all that we care about is getting the "right" answer, and on producing a system that gets the "right" answer as often as possible, so we decided to assign a positive value only to the analyses of those words which matched our gold standard analysis. The gold standard contains an indication of where the final suffix is in each (non-compound, non-proper noun) word, if there is one.

We ran into the following sorts of issues in developing the gold standard:

1. Words in which we did not know whether there was a morphological analysis. Is there a morphological analysis in such words as *boisterous*, *ambassador*, *annual*, *poem* (cf. *poet*), *agrarian*, *armor*, *benediction*, *crucial*, or *worn*?

[8] The initial preparation of this gold standard was done by Nikki Adams.

2. Words in which we were certain that there was a morphological analysis, but we were not sure which of two different analyses was the "right" one: is *allergic* based on a stem *allerg*, or is it from *allergy* plus the suffix *ic*? Is *alphabetical* based on *alphabet* or on *alphabetic*? Is *Algerian* from *Algeria* plus *–n*, or plus *–an*, or plus *–ian*? We know there is a suffix *–ian* in *Corinth-ian* (and maybe in *Belg-ian*), and *Palestin-ian*, and probably in *Canad-ian*. But what about *Cuban*? Is that a suffix *–an* or *–n*? In a different area, is *dogmatically* to be analyzed as *dogmatic* plus *ally*? Most words ending in *–ally* are arguably made up of two suffixes, -*al*- plus *–ly*, as in *abnormally* (from *abnormal* plus *ly*); but *dogmatical* is not a word: shouldn't this play a role in our analysis?
3. Words in which simple *segmentation* of the words into stem plus suffix was not sufficient; the true stem of the word was different from the result of segmenting the word into two pieces. The clearest example of this involved final *–e*'s: *loving* is composed of *love* plus *–ing*. In other cases, the modification is greater: *decision* is *decide* + *ion, cutting* is *cut* plus *ing*, *decency* is *decent* plus *y*. *Curiosity* is *curious* + *ity*. Is *application* built from *apply* plus *ation*? Not so clear.
4. In some words, segmentation is the wrong thing to worry about: *crises* is *crisis* in the plural form. How do deal with that: treat it as *crisis* + *s*?
5. In some cases, it is not clear what the "right" form for the suffix is. Is the analysis of *churches* to be *church* plus *s* or plus *es*?
6. We know there is morphology involved, but is it English morphology? Is *corpus* based on a stem *corp* plus a suffix *us*? I am not sure, though I am reasonably confident that *alumnus* is *alumn* + *us* (related to *alumn* + *i*). Similarly: *debutante*.

We decided that our research goals would be best satisfied by the following set of decisions:

1. Make the standard of the gold standard extremely high; a low score is an acceptable consequence. It should not be the case that the algorithm is penalized if it comes up with an analysis that is in some sense correct, and yet "better" than the one placed on the gold standard. For example, if the algorithm discovers the analysis of *alumnus* as *alumn* plus *–us*, it should not be penalized for this, even if we are surprised that it does so.
   When it is really not clear what the analysis is, do not score the algorithm one way or the other on the word. The word will still be part of the input, but it will not be scored. We also made the assumption that the analyses of proper nouns were not to be tested.
2. When there is clear allomorphy, make a decision ahead of time as to which aspects of the morphology the algorithm is responsible for. At this point in time, we decided that we wanted our algorithm to be tested on learning the stem-final *–e* deletion and stem-final *–y* allomorphy, and so we set the gold standard correct analysis of words such as *loving* and *cries* as *love+ing* and *cry+es* (but not *cry+s*), respectively. In future work, we will add to our

gold standard, and make it possible to select which other aspects of English allomorphy one wishes one's algorithm to be tested against.

3. The gold standard must be made publicly available.

On the first 200,000 and the first 300,000 words of the Brown corpus, Linguistica achieved accuracy of 72%. Of the errors (that is, of the 28% of the words that were not correctly analyzed), approximately 30% were due to inaccurately reconstructing "missing" stem-final –*e*'s. For example, when the words *abused* and *abusive* were found (but no other related words, notably *abuse*), the algorithm was unable to reconstruct abuse as the stem, and it reconstructed instead *abus*, and these analyses were scored as errors. (That is, if we did not demand the reconstruction of these –*e*'s, accuracy would rise to approximately 80%).

12 Conclusions and additional word

We have summarized in this paper the critical elements of an algorithm for the unsupervised learning of the morphology of a language like English. It has been implemented and tested, and is available at http://linguistica.uchicago.edu. Executable for Windows and Linux and source code can be downloaded from that site.

One of the goals of the present work is to determine what the limits of language learning are when the input data contains no semantic information and little or no use of syntactic context. It seems to me that determining such limits is a very important scientific question – even if it is difficult to make absolutely precise how to determine in any given case whether semantic or semantic-like information is being used. I think it is fair to say that *we do not know* what the limits are of how much morphological structure can be inferred without access to such resources as dictionaries, bilingual corpora, full semantic representations, and so on. There is only one way to find out, though, and that is to try as hard as we can to develop a system that infers the morphology from the very limited resources used in Linguistica and then see what can, and what cannot, be learned. I believe that it is fair to say that we have not hit any evident roadblocks so far, even if it is too soon to say where and when the first one may appear.

References

Albright, A. (2002) Islands of reliability for regular morphology: Evidence from Italian. *Language* **78**, 684–709.

Albright, A. and Hayes, B. (2003) Rules vs. analogy in English past tenses: a computational/experimental study. *Cognition* **90**, 119–61.

Bell, T. C., Cleary, J. G. and Witten, I. H. (1990) *Text Compression*. Prentice Hall.

Brent, M. (1999) An Efficient, Probabilistically Sound Algorithm for Segmentation and Word Discovery. *Machine Learning* **34**(1–3), 71–105.

Ellison, T. M. (1991) The iterative learning of phonological constraints. Dissertation. http://citeseer.nj.nec.com/ellison91iterative.html

Goldsmith, J. (2001) The unsupervised learning of natural language morphology. *Computational Linguistics* **27**, 153–198.

Goldsmith, J., and Hu, Y. (2004) From signatures to finite state automata. *Midwest Computational Linguistics Colloquium*, Bloomington IN.

Hafer, M. A. and Weiss, S. F. (1974) Word segmentation by letter successor varieties. *Information Storage and Retrieval* **10**, 371–385.

Harris, Z. (1955) From phoneme to morpheme. *Language* **31**, 190–222.

Harris, Z. (1967) Morpheme boundaries within words: report on a computer test. *Transformations and Discourse Analysis Papers 73.*

Harris, Z. (1970) *Papers in Structural and Transformational Linguistics.* D. Reidel.

Hu, Y., Matveeva, I., Goldsmith, J. and Sprague, C. (2005) The SED heuristic for morpheme discovery: a look at Swahili. In: W. Sakas, A. Clark, J. Cussens and A. Xanthos. *Psychocomputational Models of Human Language Acquisition Workshop at ACL 2005.*

Neuvel, S. and Fulop, S. (2002) Unsupervised learning of morphology without morphemes. *Proceedings of the ACL Workshop on Morphological and Phonological Learning*, pp. 31-40. Philadelphia.

Rissanen, J. (1989) *Stochastic Complexity in Statistical Inquiry*. World Scientific Series in Computer Science. World Scientific.

Wallace, C. S. and Georgeff, M. P. (1983) A general objective for inductive inference. Technical Report 32, Department of Computer Science, Monash University.

Wallace, C. S. and Dowe, D. L. (1999) Minimum Message Length and Kolmogorov Complexity. *The Computer Journal* **42**(4), 270–283.

Xanthos, A. (2003) Du k-gramme au mot: variation sur un thme distributionnaliste. *Bulletin de linguistique et des sciences du langage (BIL)* 21.