Unsupervised Acquisition of Language Morphology

Jay Jech
University of Minnesota, Morris
600 E. 4th St.
Morris, MN 56267
jech0017@umn.edu

ABSTRACT
The morphology of a language is a knowledge of the ways in which the language’s words can change in form, while remaining similar in meaning. This paper seeks to present a basic survey of recent computational techniques for acquiring knowledge of the morphology of a natural language. The focus of this paper is on unsupervised learning; the computer takes raw text in an unknown language and from this input produces a representation of the text’s morphology (often, sets of word components i.e. morphemes).

Categories and Subject Descriptors
I.2.6 [Learning]: [Knowledge acquisition, Language acquisition]; I.2.7 [Natural Language Processing]: [Language parsing and understanding, Language models, Text analysis]

General Terms
Languages, Theory, Algorithms

Keywords
unsupervised learning, morphology, orthography, morpheme

1. INTRODUCTION
The morphology of a language refers to the study of its words; a word with a particular core meaning may surface in various forms. It is the task of morphological analysis to generate models which account for these different forms of the same word. The purpose of this paper is to survey and discuss modern techniques for the automatic generation of morphological analyses.

2. BACKGROUND
Native speakers of a language naturally acquire knowledge of the language morphology. Native English speakers, for example, know that the word “frog” refers to a single amphibious creature of a certain general shape and which exhibits a set of associated characteristics. Further, they know that the word “frogs”, while distinct from the word “frog”, is yet closely related in meaning. Similarly, the speakers’ knowledge relates the words {“cat”, “cats”} and yet recognizes that the pair {“goose”, “geese”} shares the same relation, even though the latter is quite different orthographically.

Human language learners are able to obtain all of this knowledge automatically through one or more subconscious processes. In order for computers to emulate this knowledge acquisition, it is necessary to develop a model of morphology learning. Various such models exist; any model of morphology learning can be associated with the task of generating a (correct) morphological analysis of the language.

2.0.1 A Simple Model
The simplest method for generating a morphological analysis of a newly encountered language is to examine some words in the language and look for patterns in the form of the word, or orthography. By examining the language {“frog”, “frogs”}, for instance, it is clear that the only difference in form between the two words is the “-s”. From this, a possible analysis is that “frog” is a root word with a specific meaning; “-s” is a morpheme which is applied to roots to form a new, but related, word.

If translated meanings of the words are available, then it is additionally possible to associate meanings with the morphemes. In the given example, for instance, if it is provided that “frog” means “1 (frog)” and “frogs” means “2 or more (frog)”, then it is plausible to assume that the root word “frog” means “(plural)” (in this case, a singular noun referring to an amphibious creature), and that the morpheme “-s” means “(plural)”. This approach is commonly referred to as morpheme-based morphology, or concatenative morphology. It assumes that a language consists of root words, or stems, to which affixes are added to create new word forms with slightly different meanings.

2.0.2 Selecting a Model
It is important to note that concatenative morphology is only one theoretical approach (albeit perhaps the most recognized and accepted.) There are other theories that are able to effectively produce morphological analyses without generating a list of morphemes. While concatenative morphology effectively represents English and various other languages with fair accuracy, some languages are better represented by a non-concatenative approach [8, 10]. In Mandarin Chinese, for instance, morphology is relatively unobserved; word order and sentence structure have a much stronger in-
fluence on the meaning of a sentence [3]. Other languages such as the nearly-extinct Ngalakurr of Southern Australia display such a rich morphology that syntactic structure is relatively free [1]. At present, unsupervised morphological analysers are typically designed with a particular language family in mind, although applicability to all languages is a consideration.

The goal of unsupervised morphology acquisition is for a computer to determine the relatedness of words, given only a selection of text from the language as input. Ideally, a computer could produce this information without being given any additional knowledge of the language being analysed — and it would be able to analyse any natural language. However, the differences between languages from distant families makes the construction of a “universal” morphological analyser seem unlikely, if not impossible.

### 2.0.3 Applications to Vocabulary Storage

A compact morphological analysis of a language can allow for the storage of a broad vocabulary in a smaller memory space. Using knowledge of morphology, a computer can generate vocabulary, rather than store each unique word individually. This is more efficient than storing words uniquely, since similar words will often share some redundant orthographic features. By taking advantage of this redundancy, multiple words can, for instance, be collapsed into a single root to which certain morphemes can be applied. As Creutz and Lagus illustrate,

“If one has 500,000 word forms... or essentially the same information using only 20,000 morphemes, considerable improvements in efficiency can be obtained.” [2]

A practical benefit of unsupervised morphology acquisition, then, is a reduction in the workload of human analysts. Even if a computer is only able to achieve an incomplete or partially correct morphology, having something to work with besides raw samples of the language can be useful for human morphologists who are working to produce an accurate morphology. As Goldsmith points out, “...to produce a morphology of a given language ‘by hand’ can take weeks or months” [4]. His program Linguistica is capable of producing a concatenative morphology of an input text within a trivial matter of minutes or seconds, depending on the size of the input file and the hardware of the host machine.

### 3. METHODS

#### 3.1 Linguistica

An early method, published by Goldsmith [4, 5], achieves fairly successful results for a morpheme-based unsupervised learning approach.

Goldsmith sets two primary objectives:

1. To determine morpheme boundaries.
2. To group the suffixes into categories.

The first task is a matter of dividing every word in the text into a stem and suffix(es). The second task involves analysing the frequency of the suffixes as they appear in the text, and then ranking the groups by order of frequency.

#### 3.1.1 Bootstrapping the Input

The task of dividing the words of the input text into stems and suffixes is achieved by applying one of two initial “bootstrapping” heuristics. The first heuristic applies a function, called \(H()\), to determine a split for each word. The best parse provided by the function for each word is then noted; the heuristic continues to iterate through the input text, parsing each word, “until no word changes its optimal parse, which is empirically... less than five iterations on the entire lexicon” [4].

The second heuristic makes use of \(n\)-grams. An \(n\)-gram is a probabilistic model which looks at the previous \(n\) occurrences of an entity (in this case, the previous \(n\) letters.) The heuristic looks at each word in the input text, with \(n\) from 2 to 6, examining the previous \(n\) letters starting at the end of each word (and not exceeding the start of the word). Each \(n\)-gram sequence of letters is then tallied as a possible suffix and weighted according to a probability measure. The set of \(n\)-gram suffixes accumulated in this respect is then narrowed by considering the probability of each suffix and restricting the total number of possible suffixes.

#### 3.1.2 Methodology

Implemented in C++ under the name Linguistica, Goldsmith’s technique relies on a framework known as Minimum Description Length (MDL) which was introduced by Jorma Rissanen in 1978. Minimum Description Length (MDL) could be described as a formalized version of Occam’s Razor. It seeks to find the optimal solution by selecting the hypothesis with the highest compression rate. Goldsmith uses this notion in Linguistica by considering the length of compressed data and the length of the model itself; the shortest overall is the optimal solution.

Data compression is achieved by referencing existing morphemes where they have already been used. As the input is analysed, a list of stems and a list of suffixes are generated. Representations of the actual words that appear in the text appear in a separate list which Goldsmith labels signatures. A signature is a structure comprised of two pointer lists: pointers to stems, and pointers to suffixes [4]. Each signature then represents a set of words; in the relation of each stem to a set of suffixes lies the representation of the morphology.

The signature principle, in addition to providing a compact representation of basic word morphology, allows the model to utilize recursion in the morphological representations of words. Goldsmith presents the example of English “workings” [4], which can be analyzed as having a stem of “work” and two subsequently applied suffixes: “-ings” and “-s”, respectively. While we could have the signature containing “work” as a stem also contain “-ings” in addition to “-ings” and “-s” in the list of suffix pointers, it is both more accurate and more efficient to instead apply the “-s” suffix to the word “working”. To do this, we need to be able to treat the compound word “working” as a stem, even though it exists in the model as a combination of stem and suffix. Goldsmith achieves this effect by revising the model so that for each pointer in a signature’s list of stem pointers, there is a flag which indicates if the object being pointed to is a simple stem (such as “work”) or instead a combination of stem and suffixes (as in “working”) [4].
3.1.3 Results

According to the data presented in [4], Linguistica correctly analyses on average 83% of an English or French text input of 350,000–500,000 words. 5–6% of the text are words that could normally be analysed as having a stem and suffix(es) but are here assigned incorrect stem or suffix(es); some percentage of the words (approximately 6–8%) is labeled “spurious” — these are words that would normally be treated as having a stem and suffix(es), but Linguistica did break the word into a stem and suffix(es); the remaining percentage (on average 3–4%) represents words that would have been assigned a stem or suffix(es) but which Linguistica failed to analyse accordingly.

3.1.4 Limitations

One issue with Linguistica is that it does not predict new word forms; it is restricted to only attach observed forms to words that actually appear in those forms within the text. As Creutz and Lagus point out,

“...if the following English verb forms have been observed: talk, talks, talking, walk, walked, the verbs talk and walk will go into separate paradigms...” [2]

In short, Linguistica is limited to producing a static morphology consisting of “morphological facts” of a specific input text [8]. In order to guarantee that the resulting morphology is in fact the morphology of a natural language, the input text would minimally have to contain every word (in all its forms) in the language at least once. An ideal solution would instead be capable of correctly predicting new word forms without encountering each word form directly. For instance, in the example noted by Creutz and Lagus above, a better morphological analyser would predict the forms “talked” and “walking” (allowing in this case for two signatures to be combined), regardless of the fact that instances of “talked” and “walking” did not appear directly in the input text.

We mentioned earlier that morpheme-based approaches are not always ideal for representing language morphology. Another limitation of Linguistica is a common limitation for a morpheme-based approach — an inability to appropriately capture transformational differences between morphologically related words — for instance, consider English “man” vs. “men”. Applying concatenative theory to this pair seems to require that we analyse “-m-” as the stem, with “-en” and “-an” as suffixes. However, a single-letter stem is both unlikely and undesirable; stems need to be distinguishable [2]. If “-m-” is analysed as a stem with a particular meaning, then subsequent words beginning with the letter “m” (of which there are many in English) may or may not be semantically related to the words “man” and “men” — but the presence of the letter “m” at the beginning of a word in this case seems to suggest that there is a semantic relation to the pair, even if there is not. A more common application of linguistic theory in the case of (man, men) is to postulate a morphological transformation which transforms “man” into “men” and vice versa, rather than representing the morphological relation between “man” and “men” with a stem and suffixes.

3.1.5 Related Work

Applying concatenative theory to the task of morphology acquisition is perhaps the most common approach.

Creutz and Lagus utilize a probabilistic model based on a theory known as maximum a posteriori (MAP) - a model which is equivalent to MDL [2].

Sharma et al also assume a morpheme-based and probabilistic approach; they build on the work of Goldsmith and others, then apply specific optimizations to increase effectiveness for analysing text from the Indic language family [10].

Schone and Jurafsky, too, apply a morpheme-based theory; they then incorporate knowledge of semantics and syntax in order to optimize the resulting analysis [9].

Johnson and Martin build on Goldsmith’s work and introduce a graph-based technique they call hub-searching; an attempt to adapt a model to handle prefixes and infixes [6].

3.2 Whole Word Morphologizer

A vastly different approach from that of the methods mentioned above is one which applies a more rigid theory of morphology: it does not assign any probabilities to words, nor does it rely on theoretical constructs such as morphemes, stems, or affixes [8]. Rather than determining morpheme boundaries and segmenting the words in the input text, this technique seeks to discover pure morphological relations between a given pair of words and then presents these relations in the form of general rules which can be tested against any pair of words in the input text.

Designed and implemented by Sylvain Neuvel and Sean Fulop, Whole Word Morphologizer represents morphology by generating such a set of morphological relations, here called “strategies.” Whereas an approach such as Linguistica attempts to analyse an input text in a traditional fashion comparable to that which is often employed by a human linguist [8], this approach instead seeks a more direct path to representing the morphology. This technique requires that the input text is tagged with part-of-speech categories (the implications of this will be addressed in Section 4).

3.2.1 Methodology

Whole Word Morphologizer examines every unique pair of words appearing in the text and then applies a pattern-matching technique. The two words in each pair are compared and checked for matching sequences of letters — the words must start or end with the same sequence of letters (or both start and end with the same sequence) in order to potentially be morphologically related. If the two words in the pair share a common start or end sequence, then there may be a morphological relation.

Representations of the two words are inserted into similarities and differences lists: the similarities list contains the common sequences, with the rest of the letter spaces filled in with pound (#) symbols; the differences list shows the contrasting characters, with matching sequences replaced by variables [8]. If the same morphological relation is described for two or more pairs (e.g., if an attempt is made to add a word representation as described above to either list, but that representation is already present in the list) then a strategy will be generated.

A strategy merges the similarities, differences, and part-of-speech tags into a bi-directional implication rule which can be applied against any word observed in the text. If the word matches either side of the implication, then the transformation described on the opposite side of the implication can be applied to create a new word.
Table 1, shown below, illustrates this process. The table format is modeled after the figures appearing in Neuvell et al [8]. Subscript indicates a part-of-speech tag (i.e. man
 indicates a singular noun “man.”)

<table>
<thead>
<tr>
<th>Differences</th>
<th>First Word</th>
<th>Second Word</th>
</tr>
</thead>
</table>
| X a Y N
 | X e Y N
 | X a Y N
 | X e Y N

<table>
<thead>
<tr>
<th>Similarities</th>
<th>First Word</th>
<th>Second Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>m#n</td>
<td>m#n</td>
<td>wom#n</td>
</tr>
</tbody>
</table>

Figure 1: Strategy \{Man, Men, Woman, Women\}

\[ | * * # a | N_{Np} \leftrightarrow | * * # e | N_{Np} \]

Figure 2: Strategy \{Noun_{mass} \leftrightarrow \text{Verb}_{past}\}

\[ | * # a | N_{Np} \leftrightarrow | * # e | \text{V}_{past} \]

Figure 3: Strategy \{\text{Verb}_{infin} \leftrightarrow \text{Verb}_{past}\}

\[ | * # e | \text{V}_{infin} \leftrightarrow | * # e | \text{V}_{past} \]

The strategy shown in Figure 2 would be associated with the paradigm linked to hail_{N_{Np}}, while the strategy shown in Figure 3 would be associated with the paradigm linked to hail_{V_{infin}}. As a result, the program will be capable of generating sailed_{V_{past}} (from the paradigm linked to hail_{V_{infin}}) which also contains sail_{V_{infin}}, but it will be incapable of generating sail_{N_{Np}} in this case because the strategy for transforming a \text{Verb}_{past} is associated with a separate paradigm than “sail” is linked to. This is desirable, since there is no mass noun “sail” in English.

4. COMPARISON OF METHODS

A major difference between the approaches employed by Whole Word Morphologizer and Linguistica is the size of the input text. Whole Word Morphologizer takes a text file ranging in size from 1,000 to 5,000 words [8]; Linguistica takes a text file ranging in size from 5,000 to 10,000,000 words [4]. The restriction on the input of Whole Word Morphologizer is perhaps due to the number of potential word pairs it must examine — for a 5,000 word input, if every word is unique, there is an immensely greater number of unique word pairs to analyse:

\[ \binom{5,000}{2} = 12,497,500 \]

Even if Linguistica acts on every word in a 1,000,000 word input text several times, the difference in terms of computational work between Linguistica and Whole Word Morphologizer has a potential to be very noticeable indeed; it becomes a matter of linear versus quadratic time.

Another noticeable difference between this method and those employed by one such as Linguistica is that this approach requires a part-of-speech tagged input text — every word in the input must be annotated with the word’s appropriate part-of-speech category (noun, verb, gerund, etc); Linguistica, on the other hand, requires only raw, unannotated text. While this is a marked difference between the two approaches, this additional knowledge is part of what enables Whole Word Morphologizer to generate new words. A morphology produced by Whole Word Morphologizer is more general than one produced by a concatenative approach; it does not strictly acquire morphological facts of the input, but instead produces a list of strategies which can generate observed word forms as well as words which appear indirectly in the input (e.g. “talked”, “walking” as discussed in 3.1). Whole Word Morphologizer can theoretically acquire a morphology from a tagged input, generate new (tagged) words, and then analyse the resulting text as a new input to potentially expand the vocabulary further. Linguistica, on the other hand, while capable of providing a compact representation of an input text, is incapable of expanding the input with predictions for additional words; it will only produce an analysis of what it is given.
The approach of Whole Word Morphologizer, like those similar to the method utilised by Linguistica, is still considered “unsupervised” because the program makes all of the decisions about what constitutes a morphological relation in the input text its own — it does not require any prior knowledge of the language’s morphology. Further, unsupervised part-of-speech-taggers are available and are apt to improve with time; combining Whole Word Morphologizer with an effective part-of-speech-tagger should minimize concerns about its stricter requirements when compared with an apparently more tractable approach such as Linguistica.

5. CONCLUSIONS

Current research on the unsupervised acquisition of language morphology shows great promise. There are at present several different approaches to the task — some of which are fundamentally different in terms of theory — but all of which have successfully identified morphological relations within the input text to some extent.

One type of approach allows for a text input to be analysed very quickly, in a manner similar to that often used by a human linguist; but this type of approach is presently restricted to generating a morphology that is limited to the direct scope of the input. A greatly different approach is capable of generating new possible words of the text without directly observing those words in the input; but this type of approach requires that each word in the input text is tagged with the corresponding part-of-speech category, and is more computationally complex.

Success rates of as much as 80–90% of the morphological relations for an input text being identified is by no means an insubstantial achievement. Further developments in this area may well increase this percentage to nearly 100%.

6. ACKNOWLEDGEMENTS

The author would like to extend his gratitude to Elena Machkasova — for providing valuable feedback and guidance over the course of this research project, and throughout the author’s studies at UMM.

7. REFERENCES


