Morphological and Phonological Learning: Proceedings of the 6th Workshop of the ACL Special Interest Group in Computational Phonology (SIGPHON), Philadelphia, July 2002, pp. 31-40. Association for Computational Linguistics.

Unsupervised Learning of Morphology Without Morphemes

Sylvain Neuvel
Dept. of Linguistics
sneuvel@uchicago.edu

Sean A. Fulop
Depts. of Linguistics
and Computer Science
sfulop@uchicago.edu

The University of Chicago

Abstract

The first morphological learner based upon the theory of Whole Word Morphology (Ford et al., 1997) is outlined, and preliminary evaluation results are presented. The program, Whole Word Morphologizer, takes a POS-tagged lexicon as input, induces morphological relationships without attempting to discover or identify morphemes, and is then able to generate new words beyond the learning sample. The accuracy (precision) of the generated new words is as high as 80% using the pure Whole Word theory, and 92% after a post-hoc adjustment is added to the routine.

The aim of this project is to develop a computational model employing the theory of *whole word morphology* (Ford et al., 1997) capable on the one hand of identifying morphological relations within a list of words from any one of a wide variety of languages and, on the other, of putting that knowledge to use in creating previously unseen word forms. A small application called Whole Word Morphologizer which does just this is outlined and discussed. In particular, this approach is set against the literature on computational morphology as an entirely different way of doing things which has the potential to be generalized to all known varieties of morphology in the world's languages, a feature not shared by previous methods. As it is based on a model of the mental lexicon in which all entries are entire, fully fledged words, this project also serves as an empirical demonstration that a word-based morphological theory that rejects the notion of morpheme as minimal unit of form and meaning (and/or grammatical properties) is viable from the point of view of acquisition as well as generation.

1 Morphological learning

Since its inception in the mid 1950s, the field of computational morphology has been characterized by a paucity of procedures for generation. Notwithstanding the impressive body of literature on the shortcomings of traditional Paninian morphology, most computational research projects also rely on a traditional notion of the morpheme and ignore all non-compositional aspects of morphology. These observations are obviously not unrelated and are in part inherited from the field of computational syntax where applications traditionally were designed to assign a syntactic structure to a given string of words, though this is less true today.

1.1 Segmentation and morpheme identification

Word formation and the population of the lexicon, while central to morphological theory, are noticeably absent from the field of computational morphology. Most computational work in the field of morphology has focused on the identification of morphemes or morphological parsing while paying little or no attention to generation. While these applications find a common goal in the automatic acquisition of morphology, it is helpful to distinguish between two types of analysis in light of the often very different results sought by various morphological learners.

On the one hand, some applications focus exclusively on the **segmentation** of words or longer strings into smaller units. In other words, their

function is to identify morpheme boundaries within words and, as such, they only indirectly identify morphemes as linguistic units. Zellig Harris's (Harris, 1955; Harris, 1967) pioneering work suggests that morpheme boundaries can be determined by counting the number of letters that follow a given substring within a corpus (v. (Hafer and Weiss, 1974) for a further development of Harris's ideas). Janssen (1992) and Flenner (1994; 1995) also work towards segmenting words but use training corpora in which morpheme boundaries have been manually inserted. Recent work by Kazakov and Manandhar (1998) combines unsupervised and supervised learning techniques to generate a set of segmentation rules that can further be applied to previously unseen words.

On the other hand, some computational morphological applications are designed solely to **identify morphemes** based on a training corpus and not to provide a morphological analysis for each word of that corpus. Brent (1993), for example, aims at finding the right set of suffixes from a corpus, but the algorithm cannot double as a morphological parser.

More recently, efforts have been developing which identify morphemes *and* perform some sort of analysis. Schone and Jurafsky (2001) employ a great many sophisticated post-hoc adjustments to obtain the right conflation sets for words by pure corpus analysis without annotations. Their procedure uses a morpheme-based model, provides an analysis of the words, and does in a sense discover morphological relations. Goldsmith (2001b; 2001a), inspired by de Marcken's (1995) thesis on minimum description length, attempts to provide both a list of morphemes and an analysis of each word in a corpus. Also, Baroni (2000) aims at finding a set of prefixes from a corpus, together with an affix-stem parse of each of the words.

While they might differ in their methods or objectives, all of the above morphological applications share a common characteristic in that they are learners designed exclusively for the acquisition of morphological *facts* from corpora and do not generate new words based on the information they acquire.

1.2 Parsing and generation

Only a handful of programs can both parse and generate words. Once again, these programs fall into two very distinct categories. In view of the disparity between these programs, it is useful to distinguish between genuine morphological learners able to generate from acquired knowledge and generators/parsers that implement a man-made analysis. The latter group is perhaps the most well known, so let us begin with them.

Kimmo-type applications of two-level morphology (Koskenniemi, 1983; Antworth, 1990; Karttunen et al., 1992; Karttunen, 1993; Karttunen, 1994) can provide a morphological analysis of the words in a corpus and generate new words based on a set of rules; but these programs must first be provided with that set of rules and a lexicon containing morphemes by the user. Similar work in one- and two-level morphology has been done using the Attribute-Logic Engine (Carpenter, 1992). Some of these systems (e.g. (Karttunen et al., 1987)) have a front-end that compiles more traditional linearly ordered morphological rules into the finite-state automata of two-level morphology. Once again, these applications require a set of man-made lexical rules to function. While the practical uses of such applications as PC-Kimmo are incontestable, it is clear that they are part of a different endeavour, and should not be confused with genuine morphological learners.

The other relevant group of computational applications can, as mentioned, both acquire morphological knowledge from corpora and generate new words based on that knowledge. Albright and Hayes (2001a; 2001b) tackle the wider task of acquiring morphology and (morpho)phonology based on a small paradigm list and their learner is able to generate particular inflected forms given a related word. Džeroski and Erjavec (1997) work towards learning morphological rules for forming particular inflectional forms given a lemma (a set of related words). Their learner produces a set of rules relating all the members of a paradigm to a base form. The program can then produce a member of that paradigm on command given the base form. While the methods used by Albright and Hayes and Džeroski and Erjavec radically differ, both use a form of supervised learning which significantly reduces the amount of information their learner has to acquire. Albright and Hayes train their program using a paradigm list in which each entry contains, for example, both the present and past tense forms of an English verb.

Similarly, the training data used by Džeroski and Erjavec similarly has a base form, or lexeme, associated to each and every word so that all the words of a given paradigm share a common label. The distinctions between the two methods are immaterial, what matters is that both learners are being told which words are related to which and are left with the task of describing that relation in the form a rule. In other words, the algorithms they use cannot discover that words are morphologically related.

1.3 What's morphology?

In the above algorithms, the task of determining whether one word is related to another in a morphological sense is most frequently left to the linguist, as this information has to be encoded in the training data for these algorithms. (Some of the most recent work such as (Schone and Jurafsky, 2001) and (Goldsmith, 2001b) are notable exceptions to this paradigm.) This is perhaps not surprising, since no serious attempt at defining a morphological relation has been made in the last few decades. American structuralists of the forties and fifties proposed what have been referred to as discovery procedures (v. (Nida, 1949), for example) for the identification of morphemes but since the mid fifties (Chomsky, 1955), it has been customary for morphological theory to ignore this aspect of morphology and relegate it to studies on language acquisition. But, since a morphological learner like that presented here is designed to model the acquisition of morphology, it seems that it should above all be able to determine *for itself* whether two words are morphologically related or not, whether there is anything morphological to acquire at all.

Another important thing to note about the vast majority of computational morphology learners is their reliance on a traditional notion of the morpheme as a lexical unit and their exclusive focus on concatenative morphology. There is a panoply of recent publications devoted to the empirical shortcomings of traditional so-called "Item-and-Arrangement" morphology (Hockett, 1954; Bochner, 1993; Ford and Singh, 1991; Anderson, 1992; Ford et al., 1997), and the list of phenomena that fall out of reach of a compositional approach is rather impressive: zero-morphs, ablaut-like processes, templatic morphology, class markers, partial suppletion, etc. Still, seemingly every documented morphological learner relies on a Bloomfieldian notion of the morpheme and produces an Item-and-Arrangement analysis; this description applies to all of the computational papers cited above.

2 An alternative theory

Whole Word Morphologizer (henceforth WWM) is the first implementation of the theory of Whole Word Morphology. The theory, developed by Alan Ford and Rajendra Singh at Université de Montréal, seeks to account for morphological relations in a minimalist fashion. Ford and Singh published a series of papers dealing with various aspects of the theory between 1983 and 1990. Drawing on these papers, they published a full outline of it in 1991 (Ford and Singh, 1991) and an even fuller defense of it in 1997 (Ford et al., 1997). Since then, aspects of it have been taken up in a series of publications by Agnihotri, Dasgupta, Ford, Neuvel, Singh, and various combinations of these authors. The central mechanism of the theory, the Word Formation Strategy (WFS), is a sort of non-decomposable morphological transformation that relates full words with full words (or helps one fashion a full word from another full word) and parses any complex word into a variable and a non-variable component. Neuvel and Singh (In press) offer a strict definition of morphological relatedness and, based on this definition, suggest guidelines for the acquisition of Word Formation Strategies.

In Whole-Word Morphology, any morphological relation can be represented by a rule of the following form:

(1) $|X|_{\alpha} \leftrightarrow |X'|_{\beta}$

in which the following conditions and notations are employed:

1. $|X|_{\alpha}$ and $|X'|_{\beta}$ are statements that words of the form X and X' are possible in the language, and X and X' are abbreviations of the forms of classes of words belonging to categories α and β (with which specific words belonging to the right category can be unified in form);

2. $'$ represents all the form-related differences between X and X';

3. α and β are categories that may be represented as feature-bundles;

4. $\leftrightarrow$ represents a bi-directional implication;

5. X' and X are semantically related.

There are several ramifications of (1). First, there is only one morphology; no distinction, other than a functional one, is made between inflection and derivation. Second, morphology is relational and not compositional. The program thus makes no reference to theoretical constructs such as 'root', 'stem', and 'morpheme', or devices such as 'levels' and 'strata' and relies exclusively on the notion of morphological relatedness. And since its objective is not to assign a probability to a given word or string, it must rely on a strict formal definition of a morphological relation. Ultimately, the theory takes the Saussurean view that words are defined by the differences amongst them and argues that some of these differences, namely those that are found between two or more pairs of words, constitute the domain of morphology. In other words, two words of a lexicon are morphologically related if and only if all the differences between them are found in at least one other pair of words of the same lexicon.

3 Overview of the method

Under the assumption that the morphology of a language resides exclusively in differences that are exploited in more than one pair of words within its lexicon, WWM (Algorithm 1 in the next section) compares every word of a small lexicon and determines the segmental differences found between them. The input to the current version of the program is a small text file that contains anywhere from 1000 to 5000 words. Each word appears in orthographic form and is followed by its syntactic and morphological categories, as in the example below:

(2)

cat,	Ns	(Noun, singular)
catch,	V	
catches,	V3s	(Verb, (pres.) 3rd pers. sing.)
decided,	Vp	(Verb, past)

The algorithm simply compares each letter from word A to the corresponding one from word B to produce a comparison record, which can be viewed as a data structure. Currently, it works on orthographic representations. This means it would as easily work on phonemic transcriptions, but it will require empirical evaluation to see whether the results from these can improve upon those obtained using spellings, and we have not yet gone through such an exercise. It starts on either the left or right edge of the words if the two words share their first (few) segments or their last (few) segments, respectively (the forward version is presented in Algorithm 2 in the next section). This is just a simple-minded way of aligning the similar parts of the words for the comparison; a more sophisticated implementation in the future could use a more general sequence alignment procedure. The segments are placed in one of two lists in the comparison structure (differences or similarities) based on whether or not they are identical. Each comparison structure also contains the categories of both words, and is kept in a large list of all comparison structures found from analyzing the entire corpus. The example below shows the information in the comparison structure produced from the English words *receive* and *reception.* It includes the differences and similarities between the two words, from the perspective of each word in turn, as well as the lexical categories of the words.

(3)

Differences	
First word	Second word
####ive$_{V}$	####ption$_{Ns}$
Similarities	
First	Second
rece###	rece#####

Matching character sequences in the difference section are replaced with a variable. The result is then set against comparisons generated by other pairs of words and duplicate differences are recognized. In the example below, the comparisons produced by the pairs *receive/reception, conceive/conception* and *deceive/deception* are shown.

(4)

Differences	
First word	Second word
X ive$_{V}$	X ption$_{Ns}$
X ive$_{V}$	X ption$_{Ns}$
X ive$_{V}$	X ption$_{Ns}$

Similarities	
First	Second
rece###	rece#####
conce###	conce#####
dece###	dece#####

The three comparisons in (4) share the same *formal and grammatical* differences, and so the theory indicates they should be merged into one morphological strategy. Since the differences are the same, it is only the similarities that are actually merged. Each new morphological strategy is also restricted to apply in as narrow an environment as possible. Neuvel and Singh (Neuvel and Singh, In press) suggest that any morphological strategy must be maximally restricted at all times; this is accomplished by specifying as constant all the similarities found, not between words, but between the similarities found between words. In (4), all three sets of similarities end with the sequence of letters "ce." These similarities between similarities are specified as constant in each strategy and the length of each word is also factored in. The `merge` routine called in Algorithm 2 carries out this procedure; we don't show it because it is tedious but not especially interesting. The restricted morphological strategy relating the words in (4) is as follows:

(5)

Differences	
First word	Second word
X ive$_{V}$	X ption$_{Ns}$

Similarities	
First	Second
$*$##ce###	$*$##ce#####

For the sake of clarity, we can represent the information contained in (5) in a more familiar fashion using the formalism described in (1). The vertical brackets '$|\cdot|$' are used for orthographic forms so as not to confuse them with phonemic representations.

(6) $|*\#\#\text{ceive}|_{V} \leftrightarrow |*\#\#\text{ception}|_{Ns}$

The '#' signs in the above representations stand for letters that must be instantiated but are not specified; the '$*$' symbol stands for a letter that is not specified and that may or may not be instantiated. Strategy (6) can therefore be interpreted as follows:

(6′) If there is a verb that ends with the sequence "ceive" preceded by no less than two and no more than three characters, there should also be a singular noun that ends with the sequence "ception" preceded by the same two or three characters.

After performing the comparisons and merging, WWM extracts a list of morphological strategies, which are those comparison structures whose count is more than some fixed threshold. Table 1 contains a few strategies found from the first few chapters of *Moby Dick*. These strategies result from merging comparison structures which have the same differences—merging the similarities of several unifiable word pairs, and so many have no specified letters at all.

WWM then goes through the lexicon word by word and attempts to unify each word in form and category with the left or right side of this strategy. If it succeeds, WWM replaces all the segments fully specified on the side of the strategy the word is unified with, with the segments fully specified on the other side. For example, given the noun *perception* in the corpus and strategy (6), WWM will map the word onto the right hand side of (6), take out the sequence "ception" from the end and replace it with the sequence "ceive" to produce the new word *perceive.* The category of the word will also be changed from singular noun to verb. New words can thus be generated in a rather obvious fashion by taking each word in the original lexicon and applying any strategies that can be applied, i.e. whose orthographic form and part of speech can be unified with the word at hand. Algorithm 3 shows the basic generation procedure; once again the routines called `unify` and `create` which implement the nitty-gritty details of the above description are not given because they are more tedious than interesting, and will certainly need to be changed in more general future versions of WWM. Table 2 gives some of the new words WWM creates using text from *Le petit prince* as its base lexicon.

Table 1: Word-formation strategies discovered from *Moby Dick*

Differences		Similarities		
1st word	2nd word	1st word	2nd word	Examples
Xd_{PP}	X_V	****####e#	****####e	baked/bake, charged/charge
Xed_{PP}	X_V	*########	*######	directed/direct
Xs_{Np}	X_{Ns}	******#####	******####	helmets/helmet, rabbits/rabbit
$Xing_{GER}$	Xed_{PP}	******########	******######	walking/walked, talking/talked
$Xing_{GER}$	Xs_{V3s}	*****########	*****#####	walking/walks, talking/talks
$Xness_{Ns}$	X_{ADJ}	*****#########	********#####	short/shortness
Xly_{ADV}	X_{ADJ}	******######	******####	easy/easily, quick/quickly
$Xest_{ADJ}$	X_{ADJ}	*########	*####	hardest/hard, shortest/short
Xs_{V3s}	X_V	***#####	***####	jumps/jump, plays/play
Xer_{ADJ}	X_{ADJ}	*######	*####	harder/hard, louder/loud
$Xless_{ADJ}$	X_{Ns}	*########	*####	painless/pain, childless/child
$Xing_{GER}$	Xy_{ADJ}	*########	*#####	raining/rainy, running/runny
Xed_{PP}	Xs_{V3s}	**######	**#####	played/plays
$Xings_{Np}$	X_V	***#########	***#####	paintings/paint

Table 2: Words generated from *Le petit prince*

drames	Np	droitement	ADV
dressée	PF	drôles	AIP
dresser	INF	drôlement	ADV
dressa	Vp3	dunes	Np
dressais	Vi2	durerait	Vc3
dresse	V3	décidée	PF
dressent	V6	décider	INF
dressez	V5	décida	Vp3
dressait	Vi3	décide	V3
droits	AMP	décoiffé	AM
droites	AFP	déconcentrés	AMP

The output from the algorithm is a list of words,[1] much as in Table 2, which are generated from the input corpus using the morphological relations (strategies) discovered. The method described above will clearly force WWM to create words that were already part of its original lexicon; in fact, each and every word involved in licensing the discovery of a morphological strategy will be duplicated by the program. Generated words that were *not* part of WWM's original lexicon are then added to a separate word list containing only new words. If desired, this new word list can be merged with the original lexicon for another round of discovery to formulate new strategies based on a larger dataset. Additionally, each of the new words can simply be put through another cycle of word creation by applying the same strategies as before a second time.

[1]By *word* we mean an orthographic form together with the part of speech. Further work in this vein would add meanings as well.

4 Implementation

This section contains some pseudocode showing several basic components of the Whole Word Morphologizer. Algorithm 1 shows the main procedure, which takes a POS-tagged lexicon as input and outputs a list of all words that are possible given the morphological relations present in the lexicon.

The two procedures **compforward** and **compbackward** are symmetrical, so Algorithm 2 shows just the first of these. This algorithm provides the data structure which includes the differences and similarities between each pair of words in the lexicon, in similar fashion to the examples in the preceding section. In practice, only those pairs of words which are by some heuristic sufficiently similar in the first place are compared. Additionally, the two similarities sequences for each word pair are actually represented as one sequence which encodes the information found in the two sequences of the examples in the preceding; this is just for convenience of

Algorithm 1 WWM(`lexicon`)

Require: `lexicon` to be a list of POS-tagged words.
Ensure: a list `newwords` is generated
  for all tagged words w_i **do**
    for all tagged words w_j **do**
      if w_i and w_j share a beginning sequence **then**
        compforward(w_i, w_j)
      else if w_i and w_j share an ending sequence **then**
        compbackward(w_i, w_j)
      end if
    end for
  end for
  for all `comparison` structures in the list **do**
    if count(`comparison`) $>$ Threshold **then**
      append `comparison` to the list `strategies`
      generate(`lexicon`, `strategies`)
    end if
  end for

storage and computation.

Algorithm 3 shows the outline of the final stage, which generates an output list of words from the input lexicon and the morphological strategies. The strategy list is simply a list of all comparison structures that occurred more frequently than some arbitrary threshold number.

5 Accomplishments and prospects

5.1 Initial results

Whole Word Morphologizer has been tested on a limited basis using English and French lexicons of approximately 3000 entries, garnered from the POS-tagged versions of Le petit prince and Moby Dick. The program initially, without any post-hoc corrections, achieved between 70% and 82% accuracy in generation; these figures measure the percentage of the new words beyond the original lexicon that are possible words of the language. The figures thus measure a kind of *precision* value, in terms of the precision/recall tradeoff, and are fair values in that they do not include the generated words that are already in the lexicon.

Algorithm 2 compforward(w_1, w_2)

Require: w_1 and w_2 to be (word, category) pairs.
Ensure: a data structure `comparison` documenting the different and similar letters between w_1 and w_2 is merged into the global list of comparisons. `comparison` is a structure of 5 lists $w_1$dif, $w_1$cat, $w_2$dif, $w_2$cat, sim.
  for $x = 1$ to length(w_2) **do**
    if characters $w_1(x) = w_2(x)$ **then**
      append $w_1(x)$ to list sim
      if list $w_1$dif does not end with 'X' **then**
        append 'X' to both lists $w_1$dif and $w_2$dif
      else
        append $w_1(x)$ to $w_1$dif,
        append $w_2(x)$ to $w_2$dif, append '#' to sim
      end if
    end if
  end for
  for $x =$ length$(w_2) + 1$ to length(w_1) **do**
    append $w_1(x)$ to $w_1$dif
  end for
  if dif lists and categories match a comparison already in the list `comps` **then**
    merge comparisons and increment count(`comparison`)
  else
    append `comparison` to `comps`
    count(`comparison`) $\leftarrow 1$
  end if

A satisfactory *recall* metric seems impossible to think of in its usual sense here. First of all, there are generally an indefinite number of possible words in a language. One therefore cannot give a precise set of words that we wish the system could generate from a specific lexicon, so there seems to be no way to measure the percentage of "desired words" that are in fact generated. Even if we were to make such a list by hand from the current small corpora to use as a gold standard (which has been suggested by a referee), it must also be remembered that WWM discovers strategies (morphological relations) for creating new words from given ones. It cannot be expected to discover strategies that are not evident in a corpus. Indeed, WWM will *never* discover that, for example, 'am' and 'be' are related, because according to the theory of morphology being applied these

Algorithm 3 generate(`lexicon`, `strategies`)

Ensure: a list `newwords` is generated using `lexicon` and `strategies`
  for all words in `lexicon` **do**
    for all `strategies` **do**
      if unify(`lexicon`[x], `strategies`[x]) says the word and strategy match with either left or right alignment **then**
        newword ← **create**(`lexicon`[x], `strategies`[x])
        if newword is not in the lexicon or the list `newwords` **then**
          append newword to `newwords` list
        end if
      end if
    end for
  end for

words are only related by convention, not by morphology. "Nonproductive morphology" is not really morphology.

The real point is that we do not want to hold WWM's performance up against our own ideas about morphological relations among words, since it would be practically impossible to determine not merely a large set of possible words that *linguists* think are related to those in the corpus, but rather a set of possible words that WWM *ought to generate* according to its theory. This would amount to trying to beat WWM at its own game in pursuit of a gold standard, which could only be obtained using a better implementation of WWM's theory. A perfect implementation of Whole Word Morphology would have perfect recall, in view of our eventual goal of using this theory to inform us about the morphology of a language—about what ought to be recalled. We are not trying to learn something that we feel is already known.

5.2 What's learning?

It is worth considering the endeavor of learning morphology in terms of formal learning theory, as presented in Osherson et al. (1986) or Kanazawa (1998) for example. In the classical framework, the problem of learning a language from positive example data is approached by considering the successive guesses at the target language that a purported learner makes when presented with some sequentially increasing learning sample drawn from that language. Considering just morphology, it seems that the target language is the set of all possible words of the natural language at hand, a possibly infinite (or at least indefinite) set. WWM's output is a list of generated words subsuming the corpus, which are supposed to be all the words creatable by applying its idea of morphology to that corpus. It can thus be viewed as making a guess about the target language, given a certain learning sample. If the learning sample is increased, its guess increases in size also. The errors in precision of course mean that at the current corpus sizes its guesses are for the moment not even subsets of the target language.

According to one classic paradigm, a system would be held to be a successful learner if it could be proven to home in on the target language as the learning sample increased in size indefinitely. This is Gold's (1967) criterion of *identification in the limit.* In this framework, an empirical analysis cannot be used to decide the adequacy of a learner, and we would like to deemphasize the importance of the empirical results for this purpose. That said, the empirical results are for now all we have to show, but eventually we hope to produce a mathematical proof of just what WWM can learn, and just what kinds of lexicons are learnable in Gold's sense.

To our knowledge, it has never been proven whether the total lexicon of a natural language is identifiable in the limit from the sort of data we provide (i.e. POS-tagged words), using in particular the theory of Whole Word Morphology in a perfect fashion. Still, it is interesting that nothing about this language learning paradigm says anything about morphological analysis. The current crop of true morphological learners, e.g. (Goldsmith, 2001b), endeavor to learn to analyze the morphology of the language at hand in the manner of a linguist. Goldsmith has even called his Linguistica system a "linguist in a box." This is perhaps an interesting and worthwhile endeavor, but it is not one that is undertaken here. WWM is instead attempting to learn the target language in a more direct way from the data, without first constructing the intermediary of a traditional morphological analysis. We are thus not learning the linguist's notion of morphology but rather the *result* of morphology, i.e. the word forms

of the language together with the other information that goes into a word.[2]

5.3 Post-hoc fixes and future developments

A significant proportion of errors in generation result from the application of competing ambiguous morphological strategies. For example, when using the (French) text of *Le petit prince* as its base lexicon, WWM produces two strategies relating 2nd person verb forms to their infinitives. Given the verb *conjugues* 'conjugate,' pres. 2nd sing., one strategy produces the correct *-er* class infinitive *conjuguer* while the other creates the non-word **conjuguere*, based on the relation among *-re* verb forms like *fais/faire* 'do' and *vends/vendre* 'sell.' This is because of an inherent ambiguity among various word pairs which do not fully indicate the paradigms of which they are a part. WWM then adds to its lexicon, not only the correct form, but all the outputs warranted by its grammar.

To try to correct this problem, a form of lexical blocking has been implemented in the current version of the program. WWM creates every possible word, including different strategies giving the same one, and lets lexical lookup take precedence over productive morphology. The knowledge WWM possesses about its lexicon increases considerably during the creation of morphological strategies. The program learns not only which strategies are licensed by a given lexicon, but also which words of its lexicon are related to one another. WWM can assign a number to every lexical entry and give the same "paradigm" number to related words. Before adding a newly created word to its lexicon, the program looks for an existing word with the same paradigm number and category. For example, if WWM maps the word *decoction*, which was assigned to, say, paradigm 489 onto a strategy creating plural nouns, it will look for a plural noun belonging to paradigm 489 in its lexicon before it adds *decoctions* to the list of new words.

Preliminary results are encouraging, with WWM reaching up to 92% accuracy in generation after the blocking modification. Obviously the program needs to be systematically tested on multiple lexica from different languages, but these results strongly suggest that it is possible to model the acquisition of morphology as a component of learning to generate language directly, rather than to treat computational learning as the acquisition of linguistic theory as several current approaches do, e.g. (Goldsmith, 2001b).

Although the principles of whole word morphology allow one to contemplate versions of WWM that would work on templatic morphologies, polysynthetic languages, and a host of other recalcitrant phenomena, the current instantiation of the program is not so ambitious. The comparison algorithm detailed in the previous section compares words letter by letter, either from left to right or from right to left. No other possible alignments between words are considered and WWM is in its current state only capable of grasping prefixal and suffixal morphology. We are currently developing a more sophisticated sequence alignment routine which will allow the program to handle infixing, circumfixing, and templatic morphologies of the Semitic type, as well as word-internal changes typified by Germanic strong verb ablaut.

[2] In this theory, a word's form cannot be usefully divorced from the other information that allows its proper use, and in our implementation the POS tags (poor substitutes for what should be a richer database of information) are crucial to the discovery of the strategies.

References

Adam Albright and Bruce Hayes. 2001a. An automated learner for phonology and morphology. http://www.linguistics.ucla.edu/people/hayes/learning/index.htm.

Adam Albright and Bruce Hayes. 2001b. *Burnt* and *splang*: Some problems of generality in phonological learning. http://www.linguistics.ucla.edu/people/hayes/learning/index.htm.

Stephen R. Anderson. 1992. *A-Morphous Morphology*. Cambridge University Press.

Evan L. Antworth. 1990. PC-KIMMO: a two-level processor for morphological analysis. Occasional Publications in Academic Computing 16, Summer Institute of Linguistics, Dallas, TX.

Marco Baroni. 2000. An automated distribution-driven prefix learner. Presented at the 9th International Morphology Meeting, Vienna, Austria, February.

H. Bochner. 1993. *Simplicity in Generative Morphology*. Mouton de Gruyter, Berlin.

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

Bob Carpenter. 1992. *The Logic of Typed Feature Structures*, volume 32 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press.

Noam Chomsky. 1955. The logical structure of linguistic theory. Unpublished manuscript. Published as a book with a new introduction in 1975 by Plenum Press.

Carl de Marcken. 1995. *Unsupervised Language Acquisition*. Ph.D. thesis, MIT.

Sašo Džeroski and Tomaž Erjavec. 1997. Induction of Slovene nominal paradigms. In Nada Lavrac and Sašo Džeroski, editors, *Inductive Logic Programming, 7th International Workshop*, volume 1297 of *Lecture Notes in Computer Science*. Springer.

Gudrun Flenner. 1994. Ein quantitatives Morphsegmentierungssystem für spanische Wortformen. In Ursula Klenk, editor, *Computatio Linguae II*, pages 31–62. Steiner Verlag, Stuttgart.

Gudrun Flenner. 1995. Quantitative Morphsegmentierung im Spanischen auf phonologisher Basis. *Sprache und Datenverarbeitung*, 19(2):63–79.

A. Ford and R. Singh. 1991. Propédeutique morphologique. *Folia Linguistica*, 25(3–4):549–575.

A. Ford, R. Singh, and G. Martohardjono. 1997. *Pace Panini*. Peter Lang, New York.

E. M. Gold. 1967. Language identification in the limit. *Information and Control*, 10:447–474.

John A. Goldsmith. 2001a. Linguistica: An automatic morphological analyzer. In Arika Okrent and John Boyle, editors, *CLS 36: The Main Session*, volume 36-1. Chicago Linguistic Society, Chicago.

John A. Goldsmith. 2001b. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27(2):153–198.

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

Zellig Harris. 1955. From phoneme to morpheme. *Language*, 31:190–222.

Zellig Harris. 1967. Morpheme boundaries within words: Report on a computer test. In *Transformations and Discourse Analysis Papers*, volume 73.

Charles Hockett. 1954. Two models of grammatical description. *Word*, 10:210–231.

Axel Janssen. 1992. Segmentierung französischer Wortformen in Morphe ohne Verwendung eines Lexikons. In Ursula Klenk, editor, *Computatio Linguae*, pages 74–95. Steiner Verlag, Stuttgart.

Makoto Kanazawa. 1998. *Learnable Classes of Categorial Grammars*. Studies in Logic, Language and Information. CSLI Publications and the European Association for Logic, Language and Information.

Lauri Karttunen, Kimmo Koskenniemi, and Ronald M. Kaplan. 1987. A compiler for two-level phonological rules. Technical Report CSLI-87-108, Center for the Study of Language and Information, Palo Alto.

Lauri Karttunen, Ronald M. Kaplan, and Annie Zaenen. 1992. Two-level morphology with composition. In *Proceedings of the 15th International Conference on Computational Linguistics*, volume I, pages 141–148, Nantes, France.

Lauri Karttunen. 1993. Finite state constraints. In John A. Goldsmith, editor, *The Last Phonological Rule*, pages 173–194. University of Chicago Press.

Lauri Karttunen. 1994. Constructing lexical transducers. In *Proceedings of the 15th International Conference on Computational Linguistics*, volume I, pages 406–411.

Dimitar Kazakov and Suresh Manandhar. 1998. A hybrid approach to word segmentation. In David Page, editor, *Proceedings of Inductive Logic Programming-98*, volume 1446 of *Lecture Notes in Computer Science*. Springer.

Kimmo Koskenniemi. 1983. Two-level morphology: a general computational model for word-form recognition and production. Technical Report 11, Dept. of General Linguistics, University of Helsinki.

S. Neuvel and R. Singh. In press. Vive la différence! What morphology is about. *Folia Linguistica*.

Eugene Nida. 1949. *Morphology. The descriptive analysis of words*. University of Michigan Press, Ann Arbor, MI.

Daniel N. Osherson, Michael Stob, and Scott Weinstein. 1986. *Systems that Learn*. The MIT Press, Cambridge, MA.

Patrick Schone and Daniel Jurafsky. 2001. Knowledge-free induction of inflectional morphologies. In *2nd Meeting of the North American Chapter of the ACL*, pages 183–191. Association for Computational Linguistics, Morgan Kaufman.