Machine Learning, 43, 121–162, 2001

Unsupervised Learning of Word Segmentation Rules with Genetic Algorithms and Inductive Logic Programming

DIMITAR KAZAKOV kazakov@cs.york.ac.uk
SURESH MANANDHAR suresh@cs.york.ac.uk
University of York, Heslington, York YO10 5DD, UK

Editors: Luc De Raedt, C. David Page and Stefan Wrobel

Abstract. This article presents a combination of unsupervised and supervised learning techniques for the generation of word segmentation rules from a raw list of words. First, a language bias for word segmentation is introduced and a simple genetic algorithm is used in the search for a segmentation that corresponds to the best bias value. In the second phase, the words segmented by the genetic algorithm are used as an input for the first order decision list learner CLOG. The result is a set of first order rules which can be used for segmentation of unseen words. When applied on either the training data or unseen data, these rules produce segmentations which are linguistically meaningful, and to a large degree conforming to the annotation provided.

Keywords: unsupervised machine learning, inductive logic programming, natural language, word segmentation

1. Introduction

Word segmentation is the task of splitting words into a number of constituents or *morphemes*, e.g. *sleep-ing*, *dis-member-ed*, in order to compare words and study their differences. Here word segmentation is used as related to word morphology, and not to *tokenization*, where lexical constituents are identified in the text. Word segmentation is an important subtask of morphological analysis with a range of applications from hyphenation to identification of morphosyntactic categories and text-to-speech conversion. For instance, the computational lexicons used in Natural Language Processing (NLP) are often more complex than just an exhaustive list of all words and their grammatical properties. If the lexical lookup in such a system operates by morphological analysis, "it will almost certainly have to segment the sequence of characters in the word into morphemes" (Ritchie et al., 1992).

1.1. Methods for word segmentation

Methods for word segmentation can be based on a theory of morphology laid down by an expert (Ritchie et al., 1992; Skoumalová, 1997). One can also attempt to use data (*text corpora*) in which words are annotated with their grammatical properties to learn how to segment words. For this purpose, some corpus-based approaches employ *eager learning*

in that they derive a single theory or tool that can subsequently be applied to the training or unseen data (Rumelhart & McClelland, 1986; Mooney & Califf, 1995). Other, *lazy learning*, methods use the training data directly when segmenting a word (Deligne, 1996; Yvon, 1997).

A comparison between hand-crafted tools for word segmentation and those learnt from data shows that the former, unlike the latter, can be based on rather sophisticated approaches, such as generative phonology (Chomsky & Halle, 1968) or two-level morphology (Koskenniemi, 1983), in which each word constituent is the "surface level" projection of a morpheme, which is now an abstract concept, whose actual spelling or pronunciation is obtained after the application of a number of rules. This approach allows for several word constituents, as they appear in the data, to be mapped on to a single morpheme, which is then related to a set of grammatical properties. For instance, both *-s* and *-es*, in *kicks* and *passes*, would be the surface level realisations of a single PLURAL morpheme indicating the grammatical property with the same name.

In corpus-based word segmentation, there is either no explicit theory learnt, as when neural networks (Rumelhart McClelland, 1986) or lazy learning (Daelamans et al., 1997) are used, or the derived theories are less sophisticated and do not use any abstractions of the word constituents found in data (Brill, 1994; Mikheev, 1997).

The work described here adds to the relatively small but growing number of methods in natural language processing that use little or no annotation (Collins & Singer, 1999; Riloff & Jones, 1999; Thompson et al., 1999).

1.2. Supervised ILP learning of morphology

Inductive Logic Programming (ILP) (Lavrač & Džeroski, 1994) has proved to be a feasible way to learn linguistic knowledge in domains, such as morphological analysis (Blockee, 1994), part-of-speech tagging (Cussens, 1997) and parsing (Zelle & Mooney, 1993; Kazakov, 1996). Unlike statistical or connectionist approaches, the learnt theory can be easily understood and modified by a human expert. ILP has been applied to learning word morphology on several occasions. In most cases, the main task is to learn the relationship between a word and its morphosyntactic features, which effectively involves segmenting the words and making use of the constituents produced (Mooney & Califf, 1995; Manandhar et al., 1998; Muggleton & Bain, 1999). ILP has proved very efficient in the area of morphology, as the learning of English past tense shows (Mooney & Califf, 1995; Muggleton & Bain, 1999). When trying to predict the past tense form of an English verb from its present, the current ILP approaches outperforms both the best application of connectionist approach (Rumelhart & Mc-Clelland, 1986) and propositional decision trees (Ling, 1994) known so far. The theories learnt come in two flavours, as first order decision trees (Mooney & Califf, 1995; Manandhar et al., 1998) or pure first order logic clauses (Muggleton & Bain, 1999). Of these two approaches, the former is a product of eager learning, whereas the latter is a half-way house between lazy and eager learning. As in lazy learning, the approach selected by Muggleton and Bain does not extract a single theory from the corpus. Instead, for each analysed word, it derives a theory relevant to that word, and therefore provides an explanation of its decisions.

The use of ILP for morphology learning extends the concept language used with first order logic. At present, first order logic is mainly used as a convenient way of representing operations over strings and variable-length lists of possibly structured features. Since ILP, unlike statistical or propositional learning, allows for background knowledge to be easily integrated, it is expected that the gap between the theories acquired with ILP, and hand-crafted theories, should be gradually closed by augmenting the set of background concepts, from the currently used simple string operations to increasingly more complex linguistic concepts, rule templates and theories (Kazakov, 2000a).

First-order decision lists (Mooney & Califf, 1995; Manandhar et al., 1998) are an excellent representation language for stating morphological rules. These could be rules that describe regular behaviour as well as exceptions to the rules, exceptions to exceptions, etc. Decision-lists can easily represent such ordered (exceptions-to-regular) hierarchies. Earlier work (Mooney & Califf, 1995) has shown how morphological rules in decision list notation can be learnt from suitable annotated input. In particular, the CLOG decision list learner used in this paper has been shown to perform efficiently for learning of morphological rules (Manandhar et al., 1998).

A good example of the ILP potential in learning word morphology is the case of agreement among vowels in successive syllables in Turkish known as vowel harmony (Matthews, 1997). Supplying the concepts of 'vowel', 'front vowel' and 'back vowel' would allow the ILP learner to describe that agreement by specifying that each pair of vowels should be of the same type, rather than by explicitly enumerating all correct pairs of vowels, as a propositional decision tree learner would do (Quinlan, 1986).

1.3. Unsupervised learning of word segmentation

Using annotated corpora greatly facilitates learning. However, there are situations in which one is interested in Unsupervised Learning (UL), that is, from unannotated corpora. Motivation for UL can vary from purely pragmatic, such as the high cost or unavailability of annotated corpora, to theoretical, when language is modelled as yet another communication code within the framework of Information Theory (Shannon & Weaver, 1969). Unsupervised learning is set on three pillars: a data representation framework or *language bias*, a bias introducing order among all possible representations (*representational bias*), and a search technique looking for the best representation of data with respect to the representational bias. One can also distinguish between lazy UL, which merely segments the words provided, and eager UL in which a theory potentially applicable to other data is extracted.

A look at ILP shows that first order logic allows for the easy implementation of almost any data representation. The best theory to be learnt by an ILP learner is usually selected accordig to a built-in criterion imposing as a trade-off between its generality and complexity. The learnt theory can be seen as a new representation of the data, i.e., the criterion used for its selection can be seen as a built-in representational bias. Representational biases other than the default can also be implemented. It is the issue of search that finally decides whether ILP alone is suitable for unsupervised learning. Exhaustive search usually employed in ILP is inefficient for tasks for which an exact algorithm for finding the optimal solution exists.

The complexity of the word segmentation task means that in either case one may be forced further to reduce search by imposing constraints, such as a ceiling on the length or number of word constituents.

For instance, the multigram approach text as a concatenation of a limited set of variable-length strings. The representational bias aims to maximise the likelihood of both the set of strings underlying the data, and the data itself with respect to that set of strings (Deligne, 1996; Deligne & Bimbot, 1997). The search for the best segmentation is implemented as a search for a minimal cost path in a directed acyclic graph. The method requires the *a priori* selection of the maximal length of constituents. It is based on the corpus statistics, and therefore requires a sufficient amount of data. A shortcoming of the approach is that the results cannot be reused for another data set without repeating the learning step.

Other algorithms for unsupervised learning of word segmentation using corpus statistics are also known (Harris, 1955). There is also work based on the notion of *analogy* (Pirelli, 1993). The segmentations so derived have also been used for text-to-speech conversion rules (Yvon, 1997). An eager learning approach aiming at the segmentation of words into two constituents that uses Minimal Description Length (MDL) representational bias is described by Brent et al. (1995). The approach is of theoretical interest, as it treats word morphology from the general point of view of Information Theory. However, according to the authors themselves the search method described in the paper has proved inadequate.

1.4. Unsupervised GA & ILP learning

The work described here combines some of the advantages of existing methods for unsupervised learning of word segmentation with the expressiveness of the ILP framework. Unlike several practical tools aiming at word morphology, which limit the length of word constituents (Brill, 1994; Mikheev, 1997; Deligne & Bimbot, 1997), our goal was to allow for unlimited lengths. The frequently made assumption of a maximum of two constituents per word (Pirelli, 1993; Yvon, 1997; Brent et al., 1995), although having its clear limitations, was adopted to keep the data representation simple as the main interest was focussed on the search technique and subsequent learning of first order logic segmentation rules. The Naïve Theory of Morphology (NTM) representational bias is borrowed from our previous work (Kazakov, 1997), and could be seen as a simplified version of the Minimal Description Length-based one used by Brent et al. (1995). This simplification allows for the implementation of an efficient search algorithm (Kazakov, 1997). Firstly, the NTM bias is used to define the fitness function of a genetic algorithm (GA), which is then applied to the search for the best segmentation of a list of words. Then, the list of segmented words obtained with the genetic algorithm is used as input for the decision list learning algorithm CLOG (Manandhar et al., 1998). The result is a logic program in a decision list representation that can be used to segment unseen words (figure 1).

The generality of the underlying principle, based on the notion of Minimal Description Length (MDL), is of theoretical interest, as it can be used, for instance, to compare the quality of different representations of the same word paradigm. From a more practical point of view, its application can be considered in areas where a single split per word is desirable. From a machine learning point of view, this article introduces a combination of unsupervised

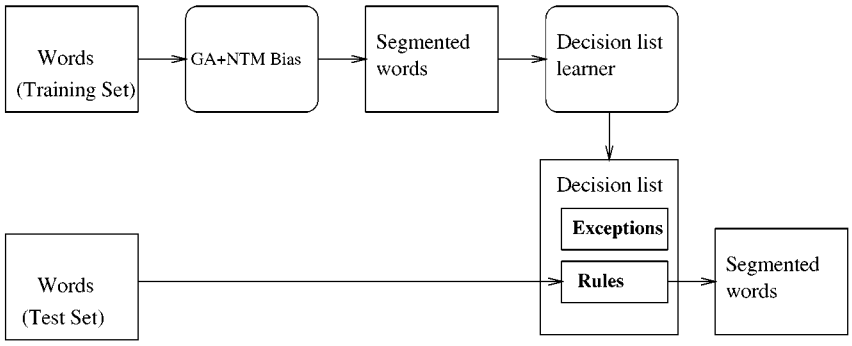

Figure 1. GA & ILP learning of word segmentation rules.

and supervised learning methods that is novel, and can be of interest for domains other than natural language processing.

The selection of data was motivated by the need for a clear and generally accepted gold standard for the segmentation used in the evaluation phase. Also, the language of the corpus had to contain word constituents of sufficient length to test the potential advantages of a framework using unlimited length constituents. Our corpus of conjugated French verb forms, unlike the English past tense data, satisfied both requirements, and hence was used in the experiments.

2. Word segmentation and word morphology

As word segmentation is usually the initial stage of a more detailed morphological analysis, the type of segmentation required depends on the application and model of word morphology used. This section starts with a formal definition of the concept of morpheme. It then presents some of the models used to represent word morphology, and discusses the relationship between the number of constituents per word allowed in the model, the complexity of search required to learn that model, and the areas of possible application.

2.1. Morpheme

It is usually assumed that word segmentation results in a number of constituents that cover the entire word, do not overlap, and follow each other. In fact, the last condition is generally valid for the Indo-European languages, but does not hold for Arabic (Matthews, 1974, p. 131).

The rôle of the word constituents has varied throughout the history of linguistics. In what is now considered as the classical approach in morphology, three basic assumptions are often made (Fradin, 1994):

1. There are minimal word constituents, *morphemes*, with which morphology operates.
2. The word is made up of morphemes which follow each other.

3. There is a set of rules that produce the actual pronunciation/spelling of each morpheme according to its context.

For instance, it can be said that the word *truthful* is made of two morphemes, *truth* and *ful*. Also, one might want to say that the word *studied* consists of the morphemes *study* and *ed* where a special rule handles the change of the final 'y' in *study* into 'i' in the context of *-ed*.

So, there are two different meanings of 'morpheme':

1. Any sequence of characters which has either grammatical or lexical meaning.
2. An invariant lexical or grammatical unit realised by one or more sequences of characters.

In the above definitions based on Matthews (1997), 'sequence of characters' was substituted for 'configuration of phonological units', since a lot of NLP research aims to provide segmentation and morphological analysis of text rather than speech, and in many languages the spelling does not directly reflect the pronunciation. That change in terminology may raise the objection that it would make possible to split a single phonological unit represented by two or more characters in the text. However, the requirement that the sequence of characters have grammatical or lexical meaning eliminates those cases.

If 'morpheme' is used in its second sense, it represents a certain kind of abstraction over a number of variants, called *morphs*. One can imagine a word represented at two levels. If the *lexical level* contains the morphemes included in the word, the *surface level* shows the actual word as formed by the concatenation of the corresponding morphs. For instance, one can speak of the pairs of morphemes DOG + 'PLURAL MORPHEME', BOSS + 'PLURAL MORPHEME', LEAF + 'PLURAL MORPHEME', 'NEGATIVE MORPHEME' + DO which at the surface level produce *dogs, bosses, leaves, undo*, respectively.

A distinction is usually made between *inflectional* and *derivational* morphology. Inflectional morphology is concerned with productive paradigms such as different forms of a verb, e.g. *walk*, *walks*, *walked* or different forms of a noun, e.g. *horse*, *horses* in which the *Part of Speech* (PoS). i.e., *noun*, *verb*, *adjective*, etc., does not change. Only the morphosyntactic features, such as Tense or Person change. On the other hand, *derivational* morphology is concerned with transformations changing the word meaning and PoS. For instance, adding *-able* to a verb can produce an adjective with a related, yet different, meaning. Combining words to form a new one is known as *compounding*, e.g. *blackboard*.

Different types of morphemes can be used to model the above types of morphology. The most detailed representation would usually include a *root* modified by *affixes*, which in turn are divided into *prefixes*, *suffixes* and *infixes*, the latter being of little or no use for the Indo-European family of languages. For instance, in *un-**assum**-ing* and *non-**deriv**-at-ion-al* the highlighted roots are preceded by a prefix, and followed by one or more suffixes.

2.2. Paradigm representation

From a practical point of view, it is important to represent all the word-forms of a word, and the corresponding set of morphosyntactic features, in a way that is more compact than exhaustive enumeration. Such a compact theory of word formation should also have generative power, enabling prediction of new word-forms from the known ones and analysis of unseen

words. A certain level of complexity is required if the theory is to reflect the subtleties of the described language. However, a detailed description of language morphology can be costly and, in practice, the complexity of the model used is often chosen as a trade-off between its cost and performance.

For many European languages, such as English, the Romance and Slavonic languages, simple and efficient representations of the inflections of the same lexical entry can be obtained as a combination of an inflectionally invariable stem (*walk*-) and a set of endings (-*s*, -*ing*). The use of several stems (*beg*-*s* but *begg*-*ed*) corresponding to different parts of the paradigm can simplify the representation by decreasing the number of endings used. The paradigms of several words (lexemes) can be further merged into larger clusters called *inflectional classes*. "They are classes [...] which determine the form which an inflection will take from one paradigm to another" (Matthews, 1974). For instance, the verbs *walk* and *talk* belong to the same inflectional class forming its various forms by adding -*ed*, -*s* or -*ing*.

The choice of representation also varies according to the purpose. For instance, when a second language is studied, the practical reasons are put forward, or, in Matthews's words (1974):

> Any language is learnt by a mixture of rote-learning, rules and practice and it is the job of the language teacher to work out what combination is the most efficient.

Indeed, it is often simpler for the human learner and the NLP developer alike to cover some of the cases as exceptions, by simply memorising them, rather than to use a complex set of rules needed otherwise.

2.3. Generative phonology and two-level morphology

Several approaches to morphology use the notion of abstract, lexical-level, morphemes. In these formalisms, words are formed by firstly combining morphemes at lexical level, and then applying rules to derive the word surface-level representation. In the framework of generative phonology (Chomsky & Halle, 1968), the initial, lexical-level sequence of morphemes is modified by sequential application of rewriting [generative] rules, and a word can go through several intermediate stages before it takes its final, surface-level form.

Some of the criticism of this approach is related to the procedural character of the rules, as they have to be applied in a certain order, and operate only in an underlying to surface direction. These objections have been addressed in Koskenniemi's work (1983) on Two-Level Morphology. The approach is based on two lexicons containing morphemes (roots and affixes), and a set of morphological rules. The rules establish whether a given sequence of characters at the surface level (as it appears in the text) can correspond to another given sequence of symbols used to represent the morphemes in the lexicon. The morphological rules are implemented as finite state transducers (FST). FST are automata with two bands, i.e. they are based on alphabets composed by pairs of symbols.

For instance, the Past Tense of *to copy* can be represented at lexical level as a combination of the morphemes COPY+ED. The corresponding string at surface level is *copi*+*ed* (figure 2). Pairs of corresponding symbols at the two levels have to be aligned, possibly making use

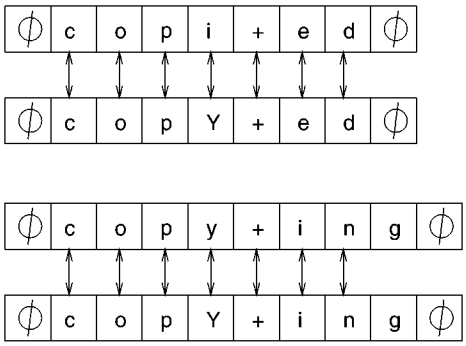

Figure 2. Lexical vs surface level in two-level morphology.

of nil ('$\emptyset$') symbols. In the case of the Present Participle, the mapping is from COPY+ING to *copy+ing*. One possible rule may postulate that the symbol Y corresponds to the letter *i* if the right-hand side (RHS) context of Y is E. There could be another rule matching Y to *y* if the RHS context of *Y* is I, etc.

From a theoretical point of view, there are several aspects that make the two-level approach interesting. It makes use of morphemes close to the surface level, and it does not need intermediary descriptions that cannot be directly related to empirical observations. Furthermore, this approach is defended by its supporters as declarative, allowing to separate the conditions governing a linguistic phenomenon from their application (Fradin, 1994). Also, two-level rules are bi-directional, i.e. they can operate in an underlying-to-surface direction (generation mode) or in a surface-to-underlying direction (recognition mode) (Antworth, 1991). One related approach worth mentioning is Kaplan and Kay's multilevel morphology (1994).

2.4. Two vs. many constituents

If no constraints are imposed on the number and length of constituents, and the only requirement is that they follow each other and do not overlap, then the number of possible segmentations of a word of length n is 2^{n-1}. It follows immediately from the fact that any segmentation can be represented as a binary vector of length $n-1$ in which each 1 marks a segment boundary.

The number of word constituents vary between models, and depends on the particular application. In the context of corpus-based learning, the higher that number, the more complex the task. There is a trade-off between the number of constituents and their maximal length—an increase in the former can be compensated with the decrease of the latter, and vice versa.

One may wish to segment a word into several constituents if a different morphosyntactic feature can be assigned to each of them. This is often unachievable, as Table 1 shows. Here the French verb ending *-ons* indicates the third person of plural. However, any attempt to find two separate morphemes indicating the verb tense and mood would fail.

Table 1. Words and morphosyntactic features.

Word-form	Tense	Mood	Person	Number
aim-ons	Present	Indicative	third	plural
aim-i-ons	Past	Indicative	third	plural
aim-er-ons	Future	Indicative	third	plural
aim-er-i-ons	Present	Conditional	third	plural
chant-er-i-ons	Present	Conditional	third	plural

Increasing the number of constituents per word brings in issues related to the generative power of the model. Such a model should clearly be designed to limit the number of spurious words it can generate. That could be achieved by introducing different types of morphemes to restrict the way in which they are combined. However, it would be very hard to learn such a model within the unsupervised learning framework, as it would mean learning morphemes and their classes, i.e., in ILP lingo, learning several new predicates in parallel.

Word segmentation can be limited to a single split per word where the two constituents belong to different classes. To avoid awkward descriptions, the two classes will be called 'prefix' (the LHS one) and 'suffix' (the one on the right). Here the terms are not used with their usual linguistic meaning. Depending on the application, they can correspond to a stem and ending, if the application uses that type of morphological lexicon (Kazakov, 1993). Alternatively, both prefix and suffix can correspond to a concatenation of morphemes in applications aiming at text-to-speech conversion of unknown words (Yvon, 1997). In that case, one is trying to copy the pronunciation of the word constituents as they appear in known words. Looking for a segmentation with the minimal number of constituents makes the approach more reliable, e.g. *dis-honest* would be a better guess than *di-shone-st*.

The up-to-two-constituents-per-word model is also the standard one used to teach the inflectional morphology (conjugations and declinations) of several languages, such as French (Bescherelle, 1980) and Czech (Havránek & Jedlička, 1981). This model is not very well suited for the combined description of inflectional and derivational morphology, where one would expect *deriv-at-ion-al*, *fire-work-s* and other similar segmentations with more than two constituents. The usefulness of such detailed segmentation depends on the application. If the task is to predict the part of speech of an unknown word, the concatenation of derivational and inflectional morphemes (*nation-alised*, v.) may be more informative then the inflectional morpheme alone (*nationalis-ed* v. but *red*, adj.).

3. Related work

This section will offer an insight into some of the existing approaches used for unsupervised word segmentation.

3.1. Segmentation techniques based on analogy

In his *Course in General Linguistics* (1916), de Saussure describes a principle of analogy according to which, in a long term, the word-forms in a given language change and tend to

form the following patterns:

$$Pref_1 + Suf_1 : Pref_1 + Suf_2 = Pref_2 + Suf_1 : Pref_2 + Suf_2 \quad (1)$$

For example, words that obey this pattern are *work*+*s* : *work*+*ing* = *sleep*+*s* : *sleep*+*ing*. In Pirelli (1993) and Yvon (1997), a word is only segmented into a prefix and suffix if one can find in the corpus another *three words* satisfying the pattern in (1). The analogy principle helps to filter out spurious segmentations generated by simply matching pairs of words. For instance, although the prefix *on*- appears in the words *onyx* and *ontology*, one would never segment *on-yx* if the corpus does not also contain another pair of words *Pref*+'*yx*', *Pref*+'*tology*' where $Pref \neq on$.

There are situations when the principle of analogy will fail even if there is a very obvious segmentation at hand. The following example in (2) shows that, for instance, W_1 cannot be segmented, as the word $Pref_2 + Suf_1$ is missing in the data.

$$\begin{aligned} W_1 &= Pref_1 + Suf_1 \quad & W_4 &= Pref_1 + Suf_2 \\ W_2 &= Pref_2 + Suf_2 \quad & W_5 &= Pref_2 + Suf_3 \\ W_3 &= Pref_3 + Suf_3 \quad & W_6 &= Pref_3 + Suf_1 \end{aligned} \quad (2)$$

Both Brill (1994) and Mikheev (1997) describe methods for finding the set of PoS tags[1] of an unknown word. Their segmentation method uses just one row or column of the analogy principle, i.e., they look for n words sharing a prefix $Pref$ satisfying $\{Pref + Suf_i\}_{i=1}^n$ or a suffix Suf satisfying $\{Pref_i + Suf\}_{i=1}^n$. The affixes to be used in the rules are selected as the ones that generate the correct PoS tag for the largest number of words. Brill's algorithm attempts to derive rules for PoS tag prediction from each affix present in the data. As this is computationally expensive, the search is pruned by limiting the maximal length of an affix to 4. This threshold can be increased at the cost of considerable reprogramming, but any decision has to be made *a priori*, and would mean a blind shot or require knowledge about the language used in the corpus.

3.2. Harris's approach

Harris (1955) describes an unsupervised approach to the segmentation of utterances spelt phonetically. The approach counts the number of different phonemes $br(Prefix)$ that can appear in an utterance of the language after a given initial sequence $Prefix$ of phonemes. The notation $br(n)$, where n is the prefix length, will also be used in the context of a given utterance. For each utterance available, its left substrings $Prefix_i$ are produced and the counts $br(Prefix_i)$ computed. Then the utterance is segmented where the function br reaches its local maxima. It is possible to adapt the method for the segmentation of words instead of whole utterances, while also replacing phonemes with letters.

Although not mentioned in the paper, Harris's method can be efficiently implemented by creating a *trie* for all utterances. A *trie* is a labelled tree where the labels correspond to a single character (Knuth, 1973). They are usually employed to provide fast and efficient method for storing dictionaries. The tree has a unique root marking the beginning of all words, and all leaves are labelled with the character marking the end of a word; only

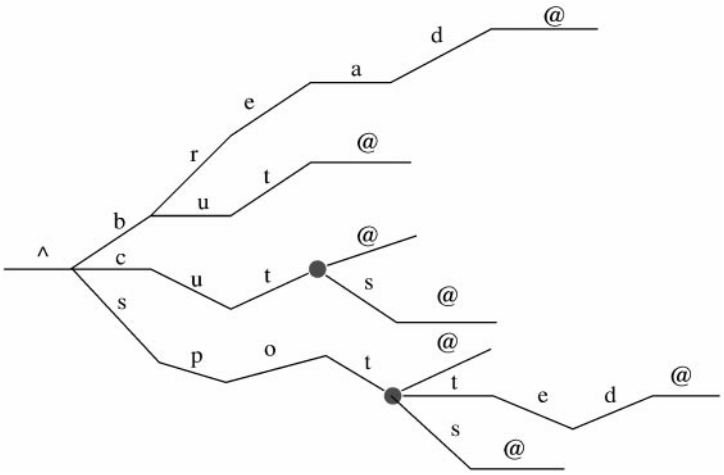

Figure 3. Trie structure for a list of words.

complete paths from the root to a leaf correspond to valid words. The function *br* (*Prefix*) gives the number of branches following the path *Prefix* (hence its name). Figure 3 gives an example trie for the words `but, cut, cuts, bread, spot, spots, spotted`.

To segment a word, the algorithm looks at the number of branches *br* (*Prefix*) for any path *Prefix* from the root downwards. Figure 4 gives the value of *br* (*Prefix*) for the above list of words. Then a word w is segmented after the initial substring of length n, if $br(n)$ reaches a local maximum that is greater than both $br(n-1)$ and $br(n+1)$. Figure 5 gives some examples of such maxima indicated with the character •. If a plateau is found, as in the case shown in figure 5(c), then all nodes on the plateau are considered as segmentation points provided a downhill slope follows. Using this criterion, one obtains `cut-`, `cut-s`, `spot-`, `spot-s`, `spot-ted`. However, `but` and `bread` are not segmented—their case is shown in figure 5(d).

An alternative approach suggested by Harris is to focus on the word (utterance) suffixes instead. Starting from the end of the word, the trie can be constructed for the words in reverse. Thus the word `mangeasses` would be entered into the trie as `sessaegnam`. Then the same

```
but       : ^-3, b-2, u-1, t-1, @-0
cut       : ^-3, c-1, u-1, t-2, @-0
cut-s     : ^-3, c-1, u-1, t-2, s-1, @-0
bread     : ^-3, b-2, r-1, e-1, a-1, d-1, @-0
spot      : ^-3, s-1, p-1, o-1, t-3, @-0
spot-s    : ^-3, s-1, p-1, o-1, t-3, s-1, @-0
spot-ted  : ^-3, s-1, p-1, o-1, t-3, t-1, e-1, d-1, @-0
```

Figure 4. Values of *br* (*Prefix*) following the last letter of each prefix.

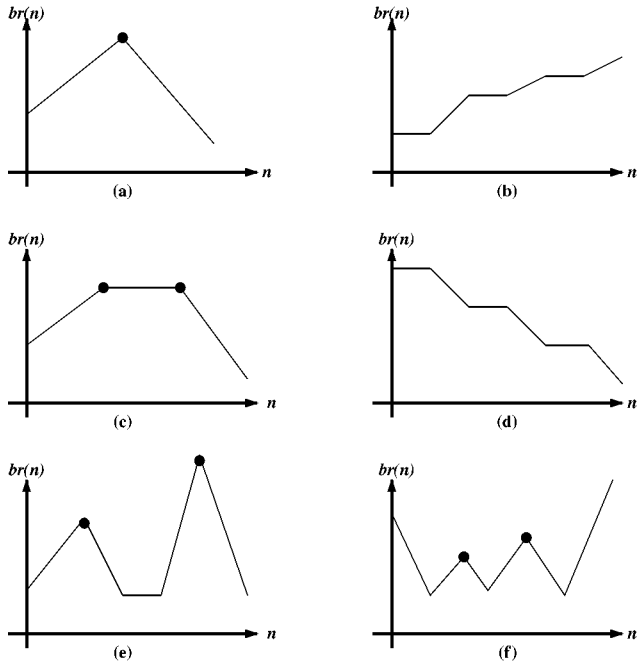

Figure 5. Segmentation points for various shapes of $br(n)$.

segmentation criterion is used. Harris claims in his paper (1955) that this modification of his technique performs better for English. In another modification, the criterion for segmentation after the n-th letter is based on both $br(n)$ and $br(n+1)$. However, it is difficult to formalise the procedure, as it makes use of a number of vaguely defined constraints of the kind $br(n+1)$ *is around Value*.

3.3. Approaches based on information theory

Deligne's thesis (1996) describes the segmentation of a long single string, possibly containing several sentences with no delimiters between words. One could adapt her method to the task of word segmentation by introducing firm boundaries in the text corresponding to the word borders. The method is looking to maximise the likelihood of both data and model at the same time. All possible segmentations are represented as a directed graph with one start and one end node. Each edge corresponds to a sequence of letters forming a segment. Each path from the start to the end node represents a particular segmentation of the whole text. Each edge of the graph is assigned a weight representing the probability of this segment. This probability is estimated simply as the relative frequency of the string in the text, and the search for the best segmentation—the one which maximises the product of all segments' probabilities—is reformulated as a search for a minimal-cost path in the graph. Once an initial segmentation is obtained, it is used to re-estimate the probability of each segment, and to segment the text again. The process stops when the segmentation becomes stable.

An obvious virtue of the method is that no previous knowledge of the language is required. The complexity of the method has led Deligne to limit the maximal length of segments in her experiments. Even in the context of segmenting lists of words, rather than continuous text, removing this restriction may cause problems, as the number of constituents become exponential with respect to the word length. This, additionally multiplied by the complexity $O(n^2)$ of the search for a minimal-cost path (n is the number of nodes in the graph) can easily become an issue, especially for aglutinative languages, such as Turkish and Magyar, in which words can be of considerable length.

Another word segmentation method based on information theory is employed by Brent et al. (1995). The article describes a binary encoding of a list of words based on lexicons of word constituents, and a table describing how these constituents are combined to form each of the words. The encoding requiring the minimal number of bits is assumed to describe the optimal segmentation of the list of words.

As the search space for possible encodings is very large, the approach limits the number of constituents per word to two. Also, suffixes (right-hand-side constituents) that share a suffix between themselves are not allowed. The search technique is iterative, trying to reduce the binary encoding size by alternately adding new suffixes to their lexicon, or by removing suffixes from that lexicon if the addition of new suffixes does not result in shorter representations. The results are summed up in the authors' words:

> [Our method] promises to be a useful tool for refining and making explicit the linguistic intuitions themselves. [...] The bad news is search. Non-trivial linguistic theories have so many degrees of freedom that exhaustive search is impossible. [...] this promises to keep the field of natural language interesting for some time to come.

In a more recent work, Brent (1999) describes an unsupervised method for recovering word boundaries from text in which all word delimiters (spaces etc.) have been removed. The method estimates probabilities of potential word sequences from which the original text could be reconstructed. Brent's method has been tested successfully on child-directed speech corpora. Further work is needed to test whether Brent's approach can be adapted successfully for the task studied here.

4. French verb: Morphology and data set used

To measure the performance of our method we produced a specially created data set consisting of entire paradigms of several French verbs. This section provides a short overview of French Verb morphology, the data set of conjugated French verbs used in the experiments, and their representation within the two-segment-per-word model.

4.1. French verb morphology

Verbs in French have a complex conjugation (see Table 2) which combines synthetic, i.e. single-word forms with analytical ones making use of auxiliary verbs. For instance (*j'*) *aimerai* is the first person, singular of Simple Future, (*j'*) *ai aimé* is the first person, singular

Table 2. First and second French conjugations: Short stem in bold face.

First conjugation					Second conjugation			
Indicative			Subjunctive		Indicative		Subjunctive	
Present			Present		Present		Present	
1. pers. sing.	**aim**	e	**aim**	e	**fin**	is	**finiss**	e
2. pers. sing.	**aim**	es	**aim**	es	**fin**	is	**finiss**	es
3. pers. sing.	**aim**	e	**aim**	e	**fin**	it	**finiss**	e
1. pers. pl.	**aim**	ons	**aim**	ions	**finiss**	ons	**finiss**	ions
2. pers. pl.	**aim**	ez	**aim**	iez	**finiss**	ez	**finiss**	iez
3. pers. pl.	**aim**	ent	**aim**	ent	**finiss**	ent	**finiss**	ent
Imperfect			Imperfect		Imperfect		Imperfect	
1. pers. sing.	**aim**	ais	**aimass**	e	**finiss**	ais	**finiss**	e
2. pers. sing.	**aim**	ais	**aimass**	es	**finiss**	ais	**finiss**	es
3. pers. sing.	**aim**	ait	**aim**	ât	**finiss**	ait	**fin**	ît
1. pers. pl.	**aim**	ions	**aimass**	ions	**finiss**	ions	**finiss**	ions
2. pers. pl.	**aim**	iez	**aimass**	iez	**finiss**	iez	**finiss**	iez
3. pers. pl.	**aim**	aient	**aimass**	ent	**finiss**	aient	**finiss**	ent
			Imperative				Imperative	
Simple Past			Present		Simple Past		Present	
1. pers. sing.	**aim**	ai			**fin**	is		
2. pers. sing.	**aim**	as	**aim**	e	**fin**	is	**fin**	is
3. pers. sing.	**aim**	a			**fin**	it		
1. pers. pl.	**aim**	âmes	**aim**	ons	**fin**	îmes	**finiss**	ons
2. pers. pl.	**aim**	âtes	**aim**	ez	**fin**	îtes	**finiss**	ez
3. pers. pl.	**aim**	èrent			**fin**	irent		
			Conditional				Conditional	
Simple Future			Present		Simple Future		Present	
1. pers. sing.	**aimer**	ai	**aimer**	ais	**finir**	ai	**finir**	ais
2. pers. sing.	**aimer**	as	**aimer**	ais	**finir**	as	**finir**	ais
3. pers. sing.	**aimer**	a	**aimer**	ait	**finir**	a	**finir**	ait
1. pers. pl.	**aimer**	ons	**aimer**	ions	**finir**	ons	**finir**	ions
2. pers. pl.	**aimer**	ez	**aimer**	iez	**finir**	ez	**finir**	iez
3. pers. pl.	**aimer**	ont	**aimer**	aient	**finir**	ont	**finir**	aient
Participle			Infinitive		Participle		Infinitive	
Present			Present		Present		Present	
	aim	ant	**aimer**	–	**finiss**	ant	**finir**	–
Past					Past			
masc. sing.	**aimé**	–			**fini**	–		
masc. pl.	**aimé**	s			**fini**	s		
fem. sing.	**aimé**	e			**fini**	e		
fem. pl.	**aimé**	es			**fini**	es		

of Past Tense formed with the auxiliary *avoir* (to have). For the purposes of our research, only synthetic forms of the verb have been considered.

There are three main conjugations of the French Verb. Further subdivision can bring the number of classes to as many as 82 (Bescherelle, 1980). The first conjugation consists of verbs, the infinitives of which end in *-er*, and with a first person singular present indicative form in *-e*, such as *aimer*, (*j'*) *aime*. The second conjugation includes verbs with infinitives ending in *-ir*, and a present participle ending in *-issant*. These two conjugations cover the vast majority of French verbs. A small number of verbs (about 350) form the third conjugation, e.g. *croire* and *venir*. Characteristic for this conjugation is that its verbs have many irregular forms that cannot be predicted from the conjugation of other verbs. As a result, the conjugation of these irregular verbs has to be described separately for each of them.

The paradigm of the verbs of the first two conjugations, considered 'regular', can be represented as produced by the concatenation of an invariable stem, and a set of endings. In Table 2, the invariable stem is printed in bold face. The remaining part of the word-forms are the verb endings as described in the summary of the three French conjugations (Bescherelle, 1980, p. 20).

In the irregular verb paradigms belonging to the third conjugation, the stem may change completely, so that there are word-forms the stems of which do not share even a single letter/phoneme. The annotation of the word-forms in these paradigms is problematic. Although one might use as standard the segmentation based on the endings from the above mentioned summary, a detailed, per-paradigm, description based on a recognised authority is missing, and we opted against the introduction of a single gold standard for any of the verbs of the third conjugation. Instead, for these verbs, non-quantitative evaluation of the segmentation produced by the learning algorithm has been carried out on individual basis.

Even for the 'regular' verbs, the spelling of the stem, i.e., the word constituent on the left of what was defined as endings in Table 2, can undergo some changes. For instance, *mang-e*, *mange-ons*, *plac-e*, *plaç-ons*. There are also cases when both the spelling and pronunciation of the stem change, as in *cèd-e*, *céd-ons*, *adhèr-e*, *adhér-ons*, *pès-e*, *pes-ons*. Because of these peculiarities, grammarians (Bescherelle, 1980) split the conjugations into sub-cases (13 for the first conjugation, one regular case plus one exception for the second), in which the stem is either appropriately shortened, so that the remaining part is invariable (*man-, pla-*), or more complex templates are introduced, e.g. combining the invariable part of the verb paradigm and a 'fill-in' consonant (*c-*,*-d-*), (*adh-*,*-r-*), (*p-*,*-s-*) with a set of endings which have a slot for the missing 'fill-in' consonant. Such segmentation schemes cannot be used for the evaluation of our approach because of the two-constituents-per-word assumption. In formalisms which distinguish between lexical and surface level, such as two-level morphology (Koskenniemi, 1983), there would be a single invariable stem MANG at lexical level, which is realised in the written forms at surface level as either *mang-* or *mange-*. Here we operate directly with all those alterations of the stem. For a given word-form, there is only one such stem alteration which is considered correct, and will be called *short stem* (SS).

In all forms of Simple Future, Present Conditional, and Infinitive, the short stem of regular verbs is immediately followed by the same suffix, *-er-* or *-ir-*. Similarly, all but one form in

Imperfect Subjunctive combine the short stem, namely its alternation in the second person, singular of Simple Past, with an *-ass-* for the first conjugation; the corresponding suffix for those and several other second-conjugation forms is *-iss-*. Past Participle in French is also used as adjective. The base form, masculine singular, of that adjective is formed by adding *-é*, resp. *-i* to the short stem. Then, all other word-forms, masculine plural, feminine singular, feminine plural, are formed by adding the endings *-s*, *-e*, *-es*, respectively.

Based on the observations made in this section, the following two representations of the French Verb paradigm were chosen.

Short-stem representation: All forms of the verb are split into *short stem*, as defined above, and the corresponding ending. The short stem is shown in Table 2 in bold face.

Long-stem representation: The longest of the following constituents *SS-*, *SS-ass-*, *SS-er-*, *SS-é-*, resp. *SS-*, *SS-iss-*, *SS-ir-*, *SS-i-*, is used as a stem. In Table 2, these *long stems* are separated with a blank space from the endings. An exception of this general principle is made for the singular of Present Indicative, and the third person plural of Simple Past of the *II* conjugation, where the short stem is used rather than *SS-i* or *SS-ir*. The last amendment makes the representation conform to the one commonly used when French is taught as a foreign language.

4.2. French verb data set

The data set used in the experiments described in this article consists of two parts. The first part contains the whole paradigms of 221 regular (first and second conjugation) verbs, i.e., for each of these verbs all its different synthetic forms are enumerated. These regular verbs can be grouped in classes according to the changes their short stem undergoes. These classes are shown in Table 3, where each class is represented by the infinitive of one of the verbs belonging to the class. The table also shows the number of verbs in each of the conjugational classes. The 'irregularity' of the verbs in the second part of the data set also shows in the higher number of different word-forms per paradigm, as compared to the

Table 3. Regular verb conjugation classes represented in the data set.

Class	Conjugation	Short stem alterations	Number of verbs
aimer	*I*	*invariable*	128
placer	*I*	plac-/plaç-	13
manger	*I*	mang-/mange-	12
peser	*I*	pes-/pèse-	6
céder	*I*	céd-/cèd-	6
jeter	*I*	jet-/jett-	3
appeler	*I*	appel-/appell-	4
finir	*II*	*invariable*	49
Total			221

Table 4. Data set statistics.

Data set	Word-forms	Different word-forms	Different word-forms per paradigm
Regular verbs	11322	8364	37.8
Irregular verbs	1572	1223	39.5
Regular + irregular	12894	9587	38.0

regular case (see Table 4). The words in this first part of the data set have been annotated with their short-stem and long-stem segmentations as defined in the previous section.

The data set has been generated by combining the endings of each verb paradigm with the corresponding stems. The result is stored in the predicate `fv/4`. The first three arguments of each `fv/4` clause contain a word, followed by its suffix and stem. The information in the last argument is not used at present, and contains the general conjugational class, *I*, *II* or *III*, and the representative of the finer-grain template used to generate the segmentation. The data in `fv/4` comes in two flavours, according to whether the segmentation is based on the short or the long stem. A sample of the latter data set is shown in figure 6.

The second part of the data set comprises the paradigms of the irregular (*III* conjugation) verbs *aller* (to go), *être* (to be), and examples of the following conjugational templates *boire* (2 verbs), *cuire* (5 verbs), *écrire* (4 verbs), *mettre* (3 verbs), *rendre* (4 verbs), *recevoir* (3 verbs), *courir* (3 verbs), and *tenir* (5 verbs). In total, there are 31 irregular verb paradigms in the data set. The predicate `fv/4` is used again, but only the first argument is actually used in the experiments.

Speaking of templates in the context of what is claimed to be irregular verbs could sound confusing, as in such case one should be able to define the stem and suffix by comparing the conjugation of two verbs of the same template. This is not the case here. In fact, many of the verbs within the same template are derived from each other by prefixation, for example, *courir* and *parcourir*. The other paradigms show very little difference within the same template, which is sometimes limited to a single letter, e.g. *venir* and *tenir*. A good example of the same situation in English is given by the verbs *to gun* and *to pun*. To claim that the verb stem consists of that single letter would be misleading. Finally, one could compare regular and irregular verb forms sharing the same morphosyntactic features to find the common ending. However, some of the irregular verbs have forms for which factoring out

```
%fv(Word,LongStem,Suffix,[gr=ConjugationGroup,tmpl=Template).
fv([m,a,n,g,e],[m,a,n,g],[e],[gr=1,tmpl=manger]).
fv([m,a,n,g,e,s],[m,a,n,g],[e,s],[gr=1,tmpl=manger]).
fv([m,a,n,g,e,a,n,t],[m,a,n,g,e],[a,n,t],[gr=1,tmpl=manger]).
fv([m,a,n,g,e,r,a,i],[m,a,n,g,e,r],[a,i],[gr=1,tmpl=manger]).
```

Figure 6. Data set of segmented words.

the common ending would leave them with an empty stem, e.g. (*tu*) *es* (Present Indicative, second person singular of *to be*), which is clearly infeasible.

So, although for many irregular verb forms one could identify a constituent playing the rôle of an inflectional morpheme, the exact form of this constituent would be disputable. For instance, any of the suffixes *-iens*, *-ens*, *-ns*, *-s* could play the inflectional morpheme rôle in the conjugated forms *viens* and *tiens* of the verbs *venir* and *tenir*. Here one is tempted to provide unambiguous annotation for all forms of all irregular verbs for the purposes of quantitative evaluation, but this is clearly not possible.

Personal communication with a number of French linguists[2] has also shown that consensus on each and every individual paradigm is unlikely to be reached. We could have chosen a single segmentation as correct, but in our case it would have meant taking a position in a discussion on French Verb morphology, which is far from our home ground. Instead, we have opted for providing the segmentations of irregular verbs produced with our method as food for that discussion. These segmentations can be obtained on request.

5. Unsupervised segmentation using a genetic algorithm

This section describes the bias employed here to rank putative segmentations of a set of words. The section also shows how a Genetic Algorithm (GA) using the bias as a fitness measure can be employed to search for the best segmentation of a set of words with respect to the bias.

5.1. NTM bias for word segmentation

Let us consider the segmentation of a list of words in the two-segments-per-word framework. The segment boundary position for each word can be represented by the length of the first constituent, that is, an integer between zero and the word length. Also, the segmentation of all words in the list can be represented by a vector of integers. The word list along with an arbitrary vector of that kind introduces a *naïve theory of morphology* (NTM) (Kazakov, 1997).

Each such theory defines two lexicons where prefixes, resp. suffixes are enumerated without repetition (see figure 7). The quality of the theory will be estimated by the number of characters $N = P + S$ in the prefix lexicon (P) and the suffix one (S)—the smaller that number, the better the theory. The upper bound N_{max} of that measure is given by the number of characters W in the word list. This case is reached when no prefix nor suffix has been generated twice by the theory. Now, the word segmentation bias can be formulated.

NTM Word Segmentation Bias: *among a set of naïve theories of word morphology, select the one with the lowest number of characters in the corresponding pair of lexicons.*

Note that the NTM bias, similarly to the Harris approach (see Section 3.2), does not take into account the frequencies of words in actual texts. This we believe is rightly so since the manner in which a word is segmented is independent of how often that word is used. The NTM bias is based on the hypothesis that substrings composed out of real morphemes

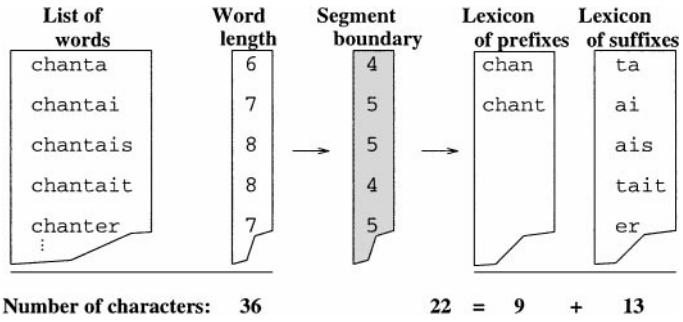

Figure 7. A naïve theory of word morphology.

occur in the words with a frequency higher than any other left or right substrings.[3] In that way, a segmentation with a low N would produce lexicons where 'prefixes' and 'suffixes' correspond very often to single morphemes or their combinations. Since the word list can be stored as a list of pairs of indices `<prefix,suffix>` along with the two lexicons of affixes, the bias described can also be seen as using the shortest representation to encode the lexicon.

The NTM bias provides an ordering on hypothesised segmentations of a given list of words. Next we need a search algorithm to search the space of plausible segmentations and find a segmentation that is optimal with respect to this bias. Since the NTM bias is a global measure that can only be computed once the segmentation for the entire word list is known, it does not lend easily to greedy search techniques that use gradient information (such as hill-climbing search). However, one can employ a genetic algorithm to search the space of possible segmentations using the NTM bias as a fitness measure (Kazakov, 1997).

Although not explored here, a possible way to divide the word constituents found into multiple constituents is the repeated use of the GA with each of the derived lexicons as input (Kazakov, 2000b).

In the next section, we describe our GA implementation of the word segmentation task.

5.2. GA implementation

Genetic algorithms (GA) (Goldberg, 1989) are an often used search technique for tasks with a large search space where greedy search methods might get stuck in local maxima. Genetic algorithms are inspired by Darwinian evolution. A GA maintains a set of candidate solutions called *chromosomes* or *individuals*, and applies the natural selection operators of *crossover* and *mutation* successively to generate new candidate solutions from existing ones. A *fitness function* is employed to rank the individuals to determine their goodness. The chromosomes are represented as a sequence of letters from a given alphabet. The *crossover* operation constructs two new child chromosomes by splicing two parent chromosomes at n points. The *mutation* operator creates a new chromosome from a single parent by randomly changing a single letter of the parent chromosome. Chromosomes are mutated according to some mutation probability known as *mutation rate*.

Procedure: Simple genetic algorithm

1. Initialisation
   a) Create a random population of candidate solutions (*individuals*) of *popsize* size.
   b) Evaluate all individuals using the fitness function.
   c) Store the best evaluated individual as *best-ever* individual.
2. Generation and Selection
   a) Sample the individuals according to their fitness, so that in the resulting *mating pool* those with higher fitness appear repeatedly with a higher probability.
   b) Apply crossover with probability *crossover_rate*.
   c) Apply mutation with probability *mutation_rate*.
   d) Evaluate all individuals using the fitness function.
   e) Update the best-ever individual.
3. If the stopping condition is satisfied, provide the *best-ever* individual as a solution. Otherwise go to step 2.

Figure 8. Simple genetic algorithm.

The version of GA employed in our experiments, known as a *simple genetic algorithm* (SGA), is shown in figure 8. Each chromosome in the SGA codes the segmentation of the entire word list. A chromosome is a vector of integers. Each integer at index i represents the segment point for the word w_i in the word list. For example, in figure 7, the sequence $<45545\ldots>$ would be a chromosome encoding the segmentation shown. The fitness of a chromosome is calculated as $N_{max} - N$. The fitness function defined in that way does not take negative values and the SGA searches for individuals with maximal fitness.

Mutation in our SGA is defined either as a shift of the morpheme boundary with one position to the left or right, or, as a random choice of a new boundary position from the allowed interval. The mutation rate is set by the user and is increased by 20% after 3 generations with the same best-ever individual. The higher mutation rate is kept until a better individual is produced and then changed back to its original value. A 2-point crossover is applied. The fittest individual is kept through the generations and is finally given as a result of the learning after a certain number of generations. There are two characteristics that are typical for all GA: 1. they are nondeterministic and 2. they provide a solution, which may be not optimal.

As running the SGA on the whole training data proved to be time consuming we split the training data into chunks of certain length. We then separately ran the SGA on each of them, and then put back together the segmented words of all chunks. This technique proved

to be a feasible trade-off between the input data size and the time needed to find a NTM of high quality. The described decomposition also allows the individual SGA runs to be run in parallel.

Another technique related to the splitting of training data into separate lists is to sort in alphabetical order the list of words before splitting it. This technique is based on the assumption that a list containing many word-forms of the same paradigm is suitable for learning since the difference between the maximal value N_{max} of the NTM bias value, and the number of characters N in the lexicons of the best NTM is very great. That difference would be increased for each of the chunks of the list of words, provided that for the language represented in the data set most of the differences between word-forms of the same paradigm are concentrated in their suffixes.

The data set used here has been artificially produced. The corpus only contains conjugated verbs to allow for a clear evaluation framework. Except for the restriction imposed on the part of speech, it is also possible to obtain such an exhaustive list of word-forms from an unannotated corpus, if the words of the corpus are listed and all duplicates removed. With the increasing corpus size, the word list would converge on an exhaustive lexicon of word-forms.

6. Learning segmentation rules with GA & ILP

The value of the segmentations obtained with the GA is limited by two factors. Firstly, in most cases, these segmentations are not optimal, that is, some of the segmentations can be seen as incorrect, i.e., suboptimal with respect to the NTM bias (Table 5 shows a sample of the GA output). Secondly, these segmentations do not represent a theory that is applicable to examples not present in the training data.

The first problem can be addressed to a certain extent by the application of some simple iterative statistical procedure similar to the one used by Deligne (1996). For instance, for a given word and prefix/suffix lexicons containing the relative frequency of the items, one can select the segmentation that maximises the product of the relative frequencies of the prefix and the suffix:

$$P_{seg} = \frac{Freq_{pref} * Freq_{suf}}{W^2}$$

Table 5. Sample of GA output.

ébarb+e	ébatt+irent	ébouillant+erez	ébourgeonn+ée
ébarb+ent	ébatt+issent	ébouillant+eriez	ébourgeonn+és
ébarb+erai	ébatt+ra	ébouillant+erions	éb+ouriffais
ébarb+erez	éborgn+es	ébouillant+ez	ébouriff+ait
ébarb+ez	éborgn+ons	ébouillant+ions	ébouriff+asse
ébarb+iez	éborgn+ées	ébouillant+ât	ébouriff+ent
ébarb+ons	ébouillant+assent	éboul+ait	ébouriff+er

Once this is done for all words, the lexicons are computed again and the step is repeated until the segmentation becomes stable (Kazakov, 2000b). However, this criterion needs the prefix *and* the suffix of the best segmentation to be present in the lexicons.

In the GA & ILP approach adopted here, the output of the GA is used in the form of clauses of the predicate `seg(Word,Suffix)`. An example of this representation is `seg([a,i,m,e,s],[e,s])`. These clauses are used as a training set of positive examples for the ILP learner CLOG. The theory for word segmentation obtained as a result is a first order decision list consisting of two parts, exceptions and rules (in that order). Each exception is in fact the training example unchanged, i.e., it can be used for the segmentation of just one word in the training set. Exceptions do not have any impact on the segmentation of unseen words, so they are removed from the decision list. In most cases, exceptions correspond to segmentations that are incorrect with respect to the NTM bias. When the segmentation rules, with the exceptions removed, are applied to the training set of words, in general the result is a better segmentation than the one generated with the GA. Figure 1 summarises the GA & ILP approach.

The rest of this section introduces CLOG, and then describes how it has been used for the learning of word segmentation rules.

6.1. Decision list learning in CLOG

CLOG, described in figure 11, is a system for learning first-order decision lists. CLOG shares a fair amount of similarity with FOIDL (Mooney & Califf, 1995). Like FOIDL, CLOG can learn first-order decision lists from positive examples only, which is a very desirable property for language learning.

The input to CLOG is a list of examples. For instance, the examples in figure 9 describe the plural of some common English nouns. From the training examples we are interested in learning an ordered set of rules. For example, given appropriate background knowledge,

```
plural([l,i,p], [l,i,p,s]).
plural([m,e,m,b,e,r], [m,e,m,b,e,r,s]).
plural([d,a,y], [d,a,y,s]).
plural([s,e,c,o,n,d], [s,e,c,o,n,d,s]).
plural([o,t,h,e,r], [o,t,h,e,r,s]).
plural([l,i,e], [l,i,e,s]).
plural([m,a,s,s], [m,a,s,s,e,s]).
plural([c,l,a,s,s], [c,l,a,s,s,e,s]).
plural([s,p,y], [s,p,i,e,s]).
plural([m,a,n], [m,e,n]).
plural([w,o,m,a,n], [w,o,m,e,n]).
plural([f,a,c,e], [f,a,c,e,s]).
```

Figure 9. Sample training data set for CLOG.

```
plural([s,p,y],[s,p,i,e,s]):- !.
plural(X,Y) :- mate(X,Y,[],[],[a,n],[e,n]),!.
plural(X,Y) :- mate(X,Y,[],[],[a,s,s],[a,s,s,e,s]),!.
plural(X,Y) :- mate(X,Y,[],[],[],[s]),!.
```

Figure 10. Rules induced with CLOG from the data in figure 9.

from the examples in figure 9, CLOG will produce the set of rules and exceptions given in figure 10. The first clause in figure 10 is an exception. Although the general rule in English is to change the ending *-y* to *-ies*, CLOG was not able to learn this rule because there is only one example in the above training set showing this pattern. The second rule states that if a word ends in an *-an* then the plural form replaces it with *-en* (see below for the definition of the `mate/6` predicate). The last rule is the default rule that will apply if the previous rules fail. This default rule will add an *-s* ending to any word.

The CLOG algorithm is a clause-at-a-time sequential cover algorithm (Mitchell, 1997), which learns a decision-list starting with the last clause. CLOG employs bottom-up guidance only to consider generalisations that are relevant to an example. This is similar to the strategy used in Progol (Muggleton, 1995).

Currently, these generalisations are generated by the user-defined predicate `generate_clauses (+Example,-Clauses)`. As an example, consider the following scheme used by Manandhar et al. (1998). Note that the definition of `generate_clauses` here is different from the one used in the experiments described in the next section. The following example is provided as an example for its ease of understanding.

```
generate_clauses(plural(U,V),Clauses):-
    bagof( ( plural(X,Y) :- (mate(X,Y,P1,P2,S1,S2), !)),
           mate(U,V,P1,P2,S1,S2),
           Clauses).
```

Here `mate(U,V,P1,P2,S1,S2)` is true if there exists `Stm` such that `U = P1+Stm+S1` and `V = P2+Stm+S2` ('+' here denotes concatenation). Intuitively, the plural of `U` is `V` if `P1` is a prefix of `U`; `P2` is a prefix of `V`; `S1` is a suffix of `U`; `S2` is a suffix of `V` and both `U` and `V` share a common stem `Stm`.

```
mate(W1,W2,[],[],S1,[]):- split(W1,W2,S1).
mate(W1,W2,[],[],[],S2):- split(W2,W1,S2).
mate(W1,W2,[],[],S1,S2):- split(W1,X,S1), split(W2,X,S2).
mate(W1,W2,P1,P2,S1,S2):- split(W1,P1,W11), split(W2,P2,W22),
                          split(W11,X,S1), split(W22,X,S2).
split([X,Y|Z],[X],[Y|Z]).
split([X|Y],[X|Z],W)   :- split(Y,Z,W).
```

Given the query `mate([l,i,p],[l,i,p,s],P1,S1,P2,S2)`, the following solutions are obtained:

```
P1 = [],   S1 = [],   P2 = [],     S2 = [s]
P1 = [],   S1 = [],   P2 = [i,p],  S2 = [i,p,s]
P1 = [],   S1 = [],   P2 = [p],    S2 = [p,s]
P1 = [l],  S1 = [p],  P2 = [p],    S2 = [p,s]
```

Thus `generate_clauses(plural([l,i,p],[l,i,p,s]),Clauses)` will give:

```
Clauses = [ ( plural(X,Y) :- (mate(X,Y,[],[],[],[s]), !) ),
            ( plural(X,Y) :- (mate(X,Y,[],[],[i,p],[i,p,s]), !) ),
            ( plural(X,Y) :- (mate(X,Y,[],[],[p],[p,s]), !) ),
            ( plural(X,Y) :- (mate(X,Y,[l],[l],[p],[p,s]), !) )
          ]
```

Each of the above clauses are generalisations of the example `plural([l,i,p], [l,i, p,s])`.

The efficiency of CLOG comes from the fact that it does not use the generate-clause-then-evaluate strategy, on which existing learners such as FOIDL are based. Instead, CLOG employs a generate-all-relevant-clauses-then-evaluate-all strategy which computes in a single iteration the heuristic goodness of all clauses. This results in a very significant gain in efficiency. For the previous example, the training set will be used for the simultaneous evaluation of all four clauses to determine the best one with respect to the *gain* function (see below).

CLOG has lower memory requirements and can train from large data sets since it keeps only a single training example in memory at any one time. All remaining training examples are stored in files. This contrasts with most other ILP learners which keep all training examples in memory.

The set of generalisations of an example produced by the predicate `generate_clauses/2` is called a *generalisation set*. This is denoted by GC in figure 11. CLOG cycles every input example through the generalisation set in a single iteration checking whether a candidate generalisation covers the example positively or negatively. Once this process is completed, the best candidate generalisation is chosen. The example set is pruned using this candidate and the cycle repeats. The following notation is used in figure 11:

- QP denotes the number of examples covered correctly;
- QN denotes the number of examples incorrectly covered;
- SP denotes the number of previously covered examples covered correctly;
- QN denotes the number of previously covered examples incorrectly covered.

In contrast to FOIDL, CLOG learns rules sequentially from the training examples, therefore, changing the ordering in the examples may result in a different ordering of the learnt rules.

The *gain* function currently used in CLOG is user-defined. For the segmentation problem we chose the following simple gain function:

$$gain = \mathrm{QP} - \mathrm{SN} - \mathit{ClauseLength}$$

where *ClauseLength* is the number of literals in the clause body and QP, SN are as defined above.

```
Let PTC be the (positive) examples to be covered
Let CPE be the set of covered (positive) examples initially empty
Let DL be the decision list being learnt initially empty
While PTC not empty
      DO
          Let x be an (arbitrary) example in PTC
          Let GC = { (G,0,0,0,0) | G is a clause that covers x }
          For each example e in PTC
              For each (G,SP,SN,QP,QN) in GC
                  if G covers e positively
                    then (G,SP,SN,QP,QN) := (G,SP,SN,QP+1,QN)
                    else if G covers e negatively
                        then (G,SP,SN,QP,QN) := (G,SP,SN,QP,QN+1) endif
                  endif
          For each example e in CPE
              For each (G,SP,SN,QP,QN) in GC
                  if G covers e positively
                    then (G,SP,SN,QP,QN) := (G,SP+1,SN,QP,QN)
                    else if G covers e negatively
                        then (G,SP,SN,QP,QN) := (G,SP,SN+1,QP,QN) endif
                  endif
          Let Best ∈ GC be such that gain(Best) is maximum
          For each example e in PTC
              if Best covers e positively
                  then
                    CPE := CPE ∪ {e}
                    PTC := PTC − {e}
              endif
          For each example e in CPE
              if Best covers e negatively
                  then
                    CPE := CPE − {e}
                    PTC := PTC ∪ {e}
              endif
          Add Best to top of DL
      ENDDO
```

Figure 11. CLOG algorithm.

The experiments in Manandhar et al. (1998) show that CLOG is significantly more efficient than FOIDL in the task of learning morphology rules. On the task of analysis of the plural of English nouns, the running times for FOIDL and CLOG are given in Table 6 (all times on a SUN SPARC 10).

In our earlier work on the segmentation of French verbs (Kazakov & Manandhar, 1998), the running times of CLOG on a data set of 200 and 500 training examples were 96 and 811 seconds respectively. For FOIDL, they were 900 and 14428 seconds respectively on the same data sets.

Table 6. Running times for CLOG and FOIDL on the task of learning rules for the analysis of English nouns.

Training set size	100	200	300	400	500	600	700	800	900	1063
CLOG [s]	21	61	112	204	277	477	646	941	1198	1771
FOIDL [s]	254	1290	2191	5190	9521	18353	33915	–	–	–

Our experience with CLOG indicates that it is ideally suited for applications where the user has a clear idea of the type of clauses that can be generalised from an example, and where the total size of the generated clauses is not too large (very roughly no more than 100–150 clauses per example on current hardware).

6.2. Setting up CLOG *for the learning of segmentation rules*

This section describes the coding of the background predicates needed by CLOG to induce decision-list rules for word segmentation.

From the SGA output, the list of all known prefixes and suffixes were generated and stored as `prefix(Prefix)`, resp. `suffix(Suffix)`. Thus, `seg([a,i,m,e,s],[e,s])` would generate `prefix([a,i,m])` and `suffix([e,s])`. These were used to restrict the segmentation points that CLOG considered in the hypothesised rules.

For the task of learning segmentation rules, we coded our definition of `generate_clauses/2` that took as input an example and generated as output all clauses the body of which contained at most two `append/3` literals and covered the example. The generated clauses included all possible variable binding patterns that were consistent with the mode declarations and covered the example. The predicate `append/3` was used in the following deterministic modes — `append(+Pref,-Suf,+W)`, `append(-Pref,+Suf,+W)`. It was defined in the standard way as:

```
append([],L,L).
append([H|T1],L,[H|T2]):-
  append(T1,L,T2).
```

Furthermore, the values for `Pref` and `Suf` had to be present in `prefix/1`, resp. `suffix/1`. For instance, provided `prefix([a,r,m])`, `prefix([a,r,m,e,r])`, `suffix([e,r])`, and `suffix([e,r,e,z])` are present in the segmentations generated with the SGA, the query `generate_clauses(seg([a,r,m,e,r,e,z],C)` produces the output in Table 7.

7. Experiments and results

The French Verb data set described in Section 4.2 has been used here to learn first order word segmentation rules with the help of a GA using the naïve theory of morphology bias and CLOG as described previously. The setting for the experiments is specified by the values of following parameters:

Table 7. Output of the query `generate_clauses([a,r,m,e,r,e,z],C).`

```
seg(W,[e,r,e,z]):- append(P,[e,r,e,z],W),!.
seg(W,[e,z])    :- append(P,[e,z],W),!.
seg(W,S)        :- append([a,r,m],S,W),!.
seg(W,S)        :- append([a,r,m,e,r],S,W),!.
seg(W,[e,r,e,z]):- append([a,r,m],S,W),
                   append(P,[e,r,e,z],W),
seg(W,[e,z])    :- append([a,r,m],S,W),
                   append(P,[e,z],W),!.
seg(W,[e,r,e,z]):- append([a,r,m,e,r],S,W),
                   append(P,[e,r,e,z],W),!.
seg(W,[e,z])    :- append([a,r,m,e,r],S,W),
                   append(P,[e,z],W),!.
seg(W,S)        :- append([a,r,m],S,W),
                   append(P,[e,r,e,z],W),!.
seg(W,S)        :- append([a,r,m],S,W),
                   append(P,[e,z],W),!.
seg(W,S)        :- append([a,r,m,e,r],S,W),
                   append(P,[e,r,e,z],W),!.
seg(W,S)        :- append([a,r,m,e,r],S,W),
                   append(P,[e,z],W),!.
seg(W,[e,r,e,z]):- append(P1,[e,r,e,z],W),
                   append(P2,[e,r,e,z],W),!.
seg(W,[e,r,e,z]):- append(P1,[e,r,e,z],W),
                   append(P2,[e,z],W),!.
seg(W,[e,z])    :- append(P1,[e,z],W),
                   append(P2,[e,r,e,z],W),!.
seg(W,[e,z])    :- append(P1,[e,z],W),
                   append(P2,[e,z],W),!.
seg(W,S)        :- append([a,r,m],S,W),
                   append([a,r,m],S1,W),!.
seg(W,S)        :- append([a,r,m],S,W),
                   append([a,r,m,e,r],S1,W),!.
seg(W,S)        :- append([a,r,m,e,r],S,W),
                   append([a,r,m],S1,W),!.
seg(W,S)        :- append([a,r,m,e,r],S,W),
                   append([a,r,m,e,r],S1,W),!.
```

Data set used: (1) both *regular* and *irregular* verbs or (2) regular verbs only.

GA input chunk size: This represents the number of words in the chunks into which the input list of words is split. The GA is then applied separately on each of them, as described above. Values used: 100, 120 words.

Training data size: The entire data set used is split in a random way into two halves. One of the halves is used as "unseen data". In different experiments, 1/4, 1/2, or all of the

remaining half has been used as training data. More precisely, the training half of the data has been split into lists of the given length (100 or 120) and after the application of the GA, either all, or each second, or one out of four GA-segmented lists have been used to learn segmentation rules with CLOG.

Data used for evaluation: Since no annotation has been provided for the irregular verbs, the quantitative evaluation of the performance of the rules learnt could only be carried out on the regular verbs in the data. The regular verbs in either the training or the unseen half of the data set, or all regular verbs (training+unseen) have been used.

Order of CLOG *training examples*: Learning from examples (`seg/2` clauses, segmented words) in alphabetical order—marked as (α) in the tables of results—which has been introduced in the GA input has been compared to setting in which the CLOG input has been sorted according to the length of the endings—marked ($\triangle$)—,so that the words with the shortest endings have been processed first, and those with the longest one last.

The ordering of CLOG training examples avoids problems in the following situation. If one of two legal endings is a right substring of the other, e.g. *-ons*, *-ions*, and the correct decision is always to produce the longer of the two, then, in the decision list learnt with CLOG, the rule deriving the shorter ending has to be *below* the rule deriving the longer one, i.e., the former rule has to be learnt first. If the opposite happens, and the rule looking for the longer ending (e.g. -ions) is added to the decision list first, then to add the segmentation rule fired by the presence of the shorter ending only (e.g. -ons) would mean to misclassify the longer stem examples. In practice, that conflict leads to over-specific rules, which are based on both stem and suffix. However, if the suggested ordering is used, no example of the longer ending will be seen, and, subsequently, no rule based on that longer ending derived, before all examples of the shorter ending have been processed by the learner.

Only word lists with the repetitions removed have been used for learning. However, evaluation has been done on the full list of verb paradigms where word-forms appeared more than once. The reason for that is related to the case when the long-stem paradigms are used as gold standard for evaluation. In that case, it may happen that the same word-form, e.g. *finis* is annotated with two different segmentations, depending on the morphosyntactic category to which it corresponds: ***fin****-is*, first and second person, singular of Present Indicative, or, ***fini****-s*, masculine plural of Past Participle. In fact, very few words in the data set allow for such multiple segmentation.

7.1. Varying the GA input length

In the first tests, a sample of 1200 different word-forms from the data set including both regular and irregular verbs was used as training data in two experiments, with GA input chunk size 100, resp. 120 words. The results are shown in Table 8. Columns 2–4 shows the ratio N/N_{max}, where N_{max} is the number of characters in all 1200 words, and N is the number of characters in the stem and ending lexicons corresponding to the list of segmented words generated by (1) GA, or (2) CLOG when its rules (without the exceptions) have been applied on the training data (cf. figure 1). As a certain number of words were not covered by any of the CLOG rules, that ratio has been computed twice for CLOG. Firstly, the ratio $(P + S)/W_{covered}$ was found for the covered, i.e., successfully segmented words only (see

Table 8. Performance as a function of the GA input chunk size (1200 w. in total).

GA input chunk	N/N_{max}			Evaluation on all annotated (regular) verbs							
	GA	CLOG		L		L or S		L, S, S-r		L,S,S-r/ss	
	all w.	cov. w.	all w.	Acc	Prec	Acc	Prec	Acc	Prec	Acc	Prec
100 words	0.145	0.135	0.170	69.93	78.73	85.06	95.76	86.12	96.96	86.56	97.45
120 words	0.136	0.128	0.153	63.89	70.16	85.14	93.48	87.66	96.26	88.40	97.07

column 'cov. w.'). Secondly, the ratio $(P+S+U)/W_{all}$ was computed for all words, where the number of characters U in the uncovered words has been added to the size of the stem and suffix lexicons, as if these words had been segmented at their very end or beginning (see column 'all w.').

The results show that the subset of words covered by the CLOG rules really corresponds to segmentations with lower overall segmentation bias value than the average for the segmentation of all words produced by the GA.

The remaining part of the table shows the results of the application of the segmentation rules on the set of all regular verbs, i.e., those for which annotation is available. Two parameters are used for evaluation. *Accuracy* has its standard meaning, the percentage of words for which the segmentation produced by our rules was identical to the one provided in the annotation. *Precision* computes the same score as accuracy, but does not take into account words for which no segmentation could be generated, i.e. no rule covers them. Accuracy and precision are shown for four different cases. In the first case (L column), only the generation of the long stem is considered correct. In the second case (L or S) we count as correct segmentations those deriving either long or short stem.

Two more cases of common errors are also counted, as they could be of interest to linguists. In some of the segmented words, the suffix *-er-/-ir-* belonging to the long stem is split, and the part common to all three conjugations, *-r-*, is attached to the ending. As this in fact could be the correct segmentation if the verbs were of the *III* conjugation, a separate case has been made. The last case in the table accounts for segmentations in which the common part of the *I* and *II* conjugation suffixes *-ass-*, *-iss-*, usually considered as a part of the long stem, is attached to the ending instead.

In the experiment where the GA was applied on shorter (100-word) chunks of the word list, the subsequently learnt first-order rules were able to produce for almost 70% of the words the same segmentation as the one used in the long-stem representation ('L' column in the table). For over 85% of the words, the application of the rules resulted in segmentation that contained either the long or the short stem ('L or S' column in the table). If only the words that were actually segmented by the rules are considered, then the score for the 'L or S' case raises to 95.76%.

Comparison between accuracy and precision achieved in each of the two experiments shows that accuracy in the 'L or S' case remains almost unchanged (∼85%). However, applying the GA on longer lists of words (120 vs 100) results in a more balanced distribution of long and short stem segmentations as the increasing number of words with the same ending partially compensates the bias in data introduced by the alphabetical sort.

Table 9. Impact of the order of examples on the rules learnt by CLOG.

Order of (the 4768) training examples	L		L or S		L, S, S-r		L, S, S-r/ss	
	Acc	Prec	Acc	Prec	Acc	Prec	Acc	Prec
Alphabetical (α)	67.61	70.62	87.93	91.86	92.05	96.16	93.00	97.15
Shortest endings first ($\triangle$)	73.64	73.64	90.58	90.58	93.21	93.21	93.97	93.97

7.2. Impact of the training examples ordering

The results of learning from a 4768-word sample of both regular and irregular verbs have been compared for two cases: (1) when CLOG was provided with the GA output in alphabetical order (α), and (2) when the GA output was sorted so that the words with the shortest endings came first ($\triangle$). All annotated (regular) verbs have been used for evaluation. The results are reported in Table 9. The overall impact of the latter ordering is positive: the segmentation rules become applicable to all words, i.e. $Acc = Prec$; more words are segmented according to the same paradigm representation (73.64%, long stem); the overall accuracy increases, in the 'L or S' case with more then 2.5%, and goes above 90%. However, the larger coverage of the $\triangle$-rules is paid for with a decrease in precision in the 'L or S' case.

7.3. Performance on unseen data

Here the performance of the segmentation rules learnt from samples of both regular and irregular verbs is evaluated on two test data sets: the unseen regular verbs, and all regular verbs (see Table 10). In the former case, none of the words used for evaluation has been used in the learning phase. The latter test data set includes all unseen words along with the inflected forms of regular verbs used for learning. Comparison shows that, when applied on unseen data only, segmentation rules achieve only slightly lower accuracy and precision for the long stem case. However, in the 'L or S' case both accuracy and precision are better on unseen data than on data also containing training examples. A possible explanation of the overall increase of 'L or S' accuracy and precision, is that there are a certain number of over-fitting rules that produce incorrect segmentation for some of the training words, but are less often or never fired in the case of unseen data.

Table 10. Performance on unseen vs all data.

Training examples	Test data set	L		L or S		L, S, S-r		L, S, S-r/ss	
		Acc	Prec	Acc	Prec	Acc	Prec	Acc	Prec
2400 (α)	Unseen reg.	68.82	72.48	89.11	93.84	92.14	97.03	92.98	97.91
2400 (α)	All reg.	70.69	76.43	86.29	93.30	88.55	95.74	89.43	96.69
4768 ($\triangle$)	Unseen reg.	70.97	70.97	92.55	92.55	95.96	95.96	96.61	96.61
4768 ($\triangle$)	All reg.	73.64	73.64	90.58	90.58	93.21	93.21	93.97	93.97

Table 11. Learning from regular verbs only vs learning from all verbs.

Tr. examples type/size/order		L		L or S		L, S, S-r		L, S, S-r/ss	
		Acc	Prec	Acc	Prec	Acc	Prec	Acc	Prec
Reg.+irreg.	4768 (α)	69.45	74.27	85.21	91.12	88.39	89.42	89.42	95.63
Regular only	4157 (α)	64.37	69.98	83.22	90.46	86.27	93.78	88.55	96.27

7.4. Impact of irregular verbs in the training data

Since annotation could only be provided in the case of regular verbs, one may question the reasons for including conjugated forms of irregular verbs to the training data. In fact, the authors' original expectation was that these examples would act as noise, which would decrease the overall performance of the rules when evaluated on the task of segmentation of regular verbs. However, the results in Table 11 did not confirm that hypothesis. Adding more examples, although corresponding to a different conjugation, increases the scores in both 'L' and 'L or S' cases. There are two reasons which could account for that. Firstly, many of the additional irregular verb examples have the same endings as the corresponding examples of either *I* or *II* conjugation. As a result, even if the verb paradigm as a whole is irregular, the individual word-forms support segmentation that conforms to either of the regular conjugations. Secondly, the fact that irregular verbs take more, sometimes completely different stems throughout their paradigm decreases the coverage of rules making use of stems, favouring in such way the learning of ending-based rules, which for French and most of the Indo-European languages are the more general ones.

7.5. Learning from training data sets of different size

After having explored the influence of several parameters in the learning or evaluation phase on the learning algorithm performance, the usual comparison of the results of learning from training samples of different size is presented here. Table 12 shows the results of learning from 1200, 2400 and 4768 word-forms of both regular and irregular verbs, where the input of both GA and CLOG was in alphabetical order, and the GA input was split into lists of 100 words each. The table shows the relatively low impact of the training data set size for

Table 12. Accuracy/precision as a function of the training data size.

Tr. examples number/order	L		L or S		L, S, S-r		L, S, S-r/ss	
	Acc	Prec	Acc	Prec	Acc	Prec	Acc	Prec
1200 (α)	69.93	78.73	85.06	95.76	86.12	96.96	86.56	97.45
2400 (α)	70.69	76.43	86.29	93.29	88.55	95.74	89.42	96.69
4768 (α)	67.61	70.62	87.93	91.86	92.05	96.16	93.00	97.15

the samples chosen. With the increasing number of training examples, precision decreases, but the overall accuracy in the 'L or S' case increases.

In all experiments described in Section 7, the GA was run for 300 generations, with a number of individuals equal to 800 or 960, i.e., eight times higher than the number of words in the input GA lists (100 or 120). The crossover rate was set to 75%, and the mutation rate to 0.5%. These parameters were selected after a number of undocumented experiments.

For populations of 800 individuals, the average time for each run on a Pentium 200 MHz platform was about 12 min. A single run of CLOG on the largest training sample (4768 examples) took about 10 hours, when running SICStus Prolog as a single thread on SGI Origin 2000 with 180 Mhz clockspeed and 256 MB RAM per processor.

7.6. Samples of Segmented Words and rules

Figure 12 shows a sample of the decision lists learnt by CLOG. In all experiments, the rules learnt corresponded to five basic patterns. Figure 13 displays representatives of these five patterns obtained in the experiment in the second row of Table 9, that is, learning rules from 4768 examples.

Rules of the first type look for a particular left substring of the word, which for the given data set is usually the stem of the word. Such rules have a limited scope given by the number of word-forms using that stem in the verb paradigm. The rules of the second type are looking for a particular ending. In French, and many other languages, word morphology operates with a closed set of inflectional suffixes. As a result, for any representative data set the rules based on such suffixes should cover a stable percentage of examples, i.e. the absolute number of words covered by them should increase approximately linearly with the size of the list of words available.

The last three rule patterns can be a result of the insufficient number of examples or noise, i.e. imperfect segmentation, in the GA output where learning from a larger or less noisy data would result in rules of the first two types. However, these rules are of real interest when they identify substrings (morphemes) that have not been or cannot be derived by the GA within the given setting. For instance, the example shown for the third pattern covers two words: *comptez* and *comptiez* where the alteration *-Ø-/-i-* in the ending allows to distinguish between Present Indicative on one hand, and Imperfect Indicative and Present Subjunctive, on the other. Similarly, the example of the fifth pattern looks for the right substring *-ira* before producing the ending *-a*. The difference *-ir-* between these two strings is actually the suffix used to derive the *I I* conjugation long stem from the short one in Simple Future and Present Conditional.

For the same case of learning rules from 4768 regular and irregular verb word-forms, the first-order decision list learnt contained 638 exceptions and 321 rules. Thirty-six of those rules corresponded to the second rule pattern, unconditionally deriving an ending whenever present in the word.

Examples of segmentation of the entire paradigm of verbs from all three conjugations are shown in Table 13. The stem and ending produced are separated with a hyphen, and the long stem as defined in Section 4 is set in bold face. The paradigms of the first two verbs, *aimer* and *manger*, belong to the first conjugation. Segmentation has produced the long stem for most of their word-forms, and the short one in the remaining cases. The short stem

```
seg([t,r,a,h,i,s,s,e],[s,s,e]):- !.
seg([g,r,o,s,s,i,s,s,o,n,s],[s,s,o,n,s]):- !.
seg([a,r,r,o,n,d,i,s,s,e,s],[s,s,e,s]):- !.
...
seg(A, B) :-      append([i,n,d,i,q,u,e,r],B,A),
                  append(_, [e,z], A), !.
seg(A, B) :-      append([l,o,u,a,s,s], B, A), !.
seg(A, B) :-      append([e,m,b,a,r,r,a,s,s],B,A),
                  append(_,[e,n,t],A), !.
seg(A, B) :-      append([a,c,c,u,s,a,s,s], B, A), !.
...
seg(A, [a,i,s]):- append(_, [a,i,s], A),
                  append(_,[i,s,s,a,i,s],A), !.
...
seg(A, B) :-      append([b,l,e,s,s], B, A), !.
seg(A, B) :-      append([b,a,r,r,a], B, A), !.
seg(A, B) :-      append([b,a,r,r], B, A), !.
seg(A, [a,i]) :-  append(_, [a,i], A), !.
seg(A, B) :-      append([a,r,r,o,s,e], B, A), !.
seg(A, B) :-      append([a,r,r,o,s], B, A), !.
seg(A, B) :-      append([a,f,f,l,i,g,e], B, A), !.
seg(A, B) :-      append([a,f,f,a,i,b,l,i], B, A), !.
seg(A, [a]) :-    append(_, [a], A), !.
seg(A, [t]) :-    append(_, [t], A), !.
seg(A, [s]) :-    append(_, [s], A), !.
...
seg(A, [e]) :-    append(_, [e], A), !.
seg(A, []) :-     append(_, [], A),  !.
```

Figure 12. A sample of the decision list rules learnt with CLOG.

has been invariably preferred for the past participle thus producing the endings *-é, -ée, -és, -ées*. In the case of *manger*, the segmentation rules have produced both alterations *mang-* and *mange-* of the short stem. The third paradigm shown is the second-conjugation verb *finir*. The only word-forms for which segmentation has produced neither the long nor the short stem are in fact the cases handled by the special amendment to the definition of the long-stem paradigm representation. Should that amendment be dropped, the result would be conform to the long-stem representation in all cases. As each representation scheme is a convention adopted out of convenience, the fact that learning from unannotated data results in a certain paradigm representation may be considered as a serious support for its wider use.

Table 13. Segmentation of *aimer*, *manger* (*I* conj.), *finir* (*I I* conj.), *aller* (*I I I* conj.)

aim-e	**mang**-e	**fin**i-s	**v**-ais
aim-es	**mang**-es	**fin**i-s	**v**-as
aim-e	**mang**-e	**fin**i-t	**v**-a
aim-ons	**mange**-ons	**finiss**-ons	**all**-ons
aim-ez	**mang**-ez	**finiss**-ez	**all**-ez
aim-ent	**mang**-ent	**finiss**-ent	**v**-ont
aim-ais	**mange**-ais	**finiss**-ais	**all**-ais
aim-ais	**mange**-ais	**finiss**-ais	**all**-ais
aim-ait	**mange**-ait	**finiss**-ait	**all**-ait
aim-ions	**mang**-ions	**finiss**-ions	**all**-ions
aim-iez	**mang**-iez	**finiss**-iez	**all**-iez
aim-aient	**mange**-aient	**finiss**-aient	**all**-aient
aim-ai	**mange**-ai	**fini**-s	**all**-ai
aim-as	**mange**-as	**fini**-s	**all**-as
aim-a	**mange**-a	**fini**-t	**all**-a
aim-âmes	**mange**-âmes	**fin**-îmes	**all**-âmes
aim-âtes	**mange**-âtes	**fin**-îtes	**all**-âtes
aim-èrent	**mang**-èrent	**fin**ir-ent	**all**-èrent
aim-erai	**manger**-ai	**finir**-ai	**ir**-ai
aim-eras	**manger**-as	**finir**-as	**ir**-as
aim-era	**manger**-a	**finir**-a	**ir**-a
aimer-ons	**manger**-ons	**finir**-ons	**ir**-ons
aim-erez	**manger**-ez	**finir**-ez	**ir**-ez
aimer-ont	**manger**-ont	**finir**-ont	**ir**-ont
aim-e	**mang**-e	**finiss**-e	**aill**-e
aim-es	**mang**-es	**finiss**-es	**aill**-es
aim-e	**mang**-e	**finiss**-e	**aill**-e
aim-ions	**mang**-ions	**finiss**-ions	**aill**-ions
aim-iez	**mang**-iez	**finiss**-iez	**aill**-iez
aim-ent	**mang**-ent	**finiss**-ent	**aill**-ent
aimass-e	**mangeass**-e	**finiss**-e	**all-ass**e
aimass-es	**mangeass**-es	**finiss**-es	**all-ass**es
aim-ât	**mange**-ât	**fin**-ît	**all-ass**e
aimass-ions	**mangeass**-ions	**finiss**-ions	**all-ass**ions
aimass-iez	**mangeass**-iez	**finiss**-iez	**all-ass**iez
aimass-ent	**mangeass**-ent	**finiss**-ent	**all-ass**ent

(*Continued on next page.*)

Table 13. (*Continued*).

aimer-ais	**manger**-ais	**finir**-ais	**ir**-ais
aimer-ais	**manger**-ais	**finir**-ais	**ir**-ais
aim-erait	**manger**-ait	**finir**-ait	**ir**-ait
aimer-ions	**manger**-ions	**finir**-ions	**ir**-ions
aim-eriez	**manger**-iez	**finir**-iez	**ir**-iez
aimer-aient	**manger**-aient	**finir**-aient	**ir**-aient
aim-é	**mang-é**	**fini**-	**all-é**
aim-és	**mang-é**s	**fini**-s	**all-é**s
aim-ée	**mang-é**e	**fini**-e	**all-é**e
aim-ées	**mang-é**es	**fini**-es	**all-é**es
aim-ant	**mange**-ant	**finiss**-ant	**all**-ant
aim-er	**manger**-	**finir**-	**all-er**

```
1. seg(A,B) :-      append([b,l,e,s,s], B, A), !.
2. seg(A,[a,i]) :- append(_,[a,i],A), !.
3. seg(A,B) :-      append([c,o,m,t,e], B, A),
                    append(C,[e,z],A), !.
4. seg(A,B) :-      append([o,r,g,a,n,i,s], B, A),
                    append([o,r,g,a,n,i,s,a],C,A), !.
5. seg(A, [a]) :-   append(_, [a], A),
                    append(_, [i,r,a], A), !.
```

Figure 13. Types of rules learnt.

Finally, Table 13 shows the paradigm of the irregular verb *aller*. Here all segmentations correspond to either the short or the long stem representation. Some of the segmentations contain the stems *v-*, *all-*, and *aill-*, all of which can be considered as alterations of the short stem. According to the grammar book used as a reference here, the *III* conjugation endings for Simple Future are *-rai*, *-ras*, etc., which for the verb in question leaves merely *i-* as the short stem. The long stem producing exactly the same endings as those in the regular conjunctions, would be *ir-*, i.e., the one derived by the rules.

7.7. Comparison between the GA and Harris

Harris's segmentation technique has been used to compare the relative merits of the GA as a preprocessing step for the ILP learner with another algorithm.

7.7.1. Experimental Comparison. In the first experiment, the two algorithms have been applied on the training data set containing 4678 regular and irregular verb forms. Then

Table 14. GA vs Harris.

	Long Stem			Short Stem		
Algorithm	Rec.	Prec.	F-score	Rec.	Prec.	F-score
GA	68.50	68.50	68.50	16.21	16.21	16.21
Harris-RL, N-split	92.22	61.48	73.76	58.66	38.91	46.78
Harris-RL, 1-split	65.15	59.94	62.42	30.62	28.17	29.34

the number of short and long stems produced has been counted for the subset of 4157 regular verb forms for which annotation was available (cf. first row of Table 9). For Harris's method, the trie based on reversed words (RL-trie) was used, as it yielded consistently better results. To avoid comparing methods that produce a different number of segments, one or two for the GA or several for Harris's method, a modification of the latter was suggested. Instead of all local maxima of the function $br(n)$, only the global maximum defined a segment boundary.If, for instance, $br(\texttt{mange}) > br(\texttt{mangeass})$ for the multiple segmentation `mange-ass-es`, this would result in the single segmentation `mange-asses`. This modification reduced the number of segments to two per word in most cases. For the words with k global maxima, $k > 1$, k different segmentations of two segments each were produced as output. The results were evaluated separately for each class of stem (long or short). Recall, precision and F-score (Eqs. 3–5) were used for evaluation, where F-score is defined as the geometric mean of recall and precision (see Table 14). The measure of accuracy used in the experiments with GA and CLOG is a specific case of recall in which at most one stem per word is produced.

$$Recall = \frac{Correct\ stems}{All\ words} \tag{3}$$

$$Precision = \frac{Correct\ stems}{All\ segmentations\ produced} \tag{4}$$

$$F\text{-}score = 2 \times \frac{Recall \times Precision}{Recall + Precision} \tag{5}$$

The results in Table 14 show that the standard Harris method is the best if several segments per word are sought after, recall is more important than precision, and there is no strong preference for the consistent use of the same, long- or short-stem, verb paradigm. However, the fact that the method generates several segments per word makes it unsuitable as a preprocessing step for our setup of the decision list learner, as it is programmed to split each word at one point, and expects training data of the same type.

The comparison between the GA and the modified version of Harris's algorithm shows the superiority of the GA in two respects. Firstly, the GA performs better in the generation of a single-type stem (the long one). Intuitively, the much higher ratio of long versus short stems generated by the GA should mean that CLOG would be much less confused by the two concurrent patterns, and the rules it learns would produce the same type of stem more often. This expectation has been confirmed by the results in Table 15. The table compares

Table 15. Harris vs GA as a preprocessing step for CLOG.

	L		L or S		L, S, S-r		L, S, S-r/ss	
Algorithum	Acc	Prec	Acc	Prec	Acc	Prec	Acc	Prec
GA+CLOG	69.45	74.27	85.21	91.12	88.39	89.42	89.42	95.63
Harris-LR+CLOG	39.85	41.00	77.38	79.61	81.01	83.35	84.61	87.01
Harris-RL+CLOG	55.01	60.93	77.50	85.75	77.50	85.75	74.50	85.85

the performance of CLOG rules learnt from the training data set (4768 words) on the whole set of training and unseen regular verbs. Two versions of Harris's method, with the trie built starting from the first (LR-trie) or last (RL-trie) word letter, are compared. The latter is better in terms of both accuracy and precision, and it more consistently generates the same type of stem.The combination of GA and ILP (CLOG) is clearly the best of the three, as it easily outperforms Harris+CLOG irrespective of whether the standard LR or the reversed (RL) procedure is employed.

7.7.2. Stem Alteration Handling. Harris's method is particularly prone to errors when stem alteration occurs. Such alterations can reflect phonetic changes or are required by spelling conventions. A number of French Verb stem alterations are shown in Table 3. It can be seen that stem changes are concentrated at its end, and in most cases consist of adding an extra letter or replacing the final one with another. For instance, the stem of the verb *manger* is *mang-*, however, the letter 'e' is added at its end when a consonant or a hard vowel (a, o, u) follows. The forms of *manger* present in the training corpus, and their segmentation by Harris and the GA, are compared in Table 16. A number of suffixes in the table start with the letter 'e', which, in these cases, does not belong to the stem *mang-*. For these words, marked with an asterisk, Harris erroneously produces the stem *mange-*. Unlike Harris's method, the GA segmentation of this verb always contains a correct short (S) or long (L) stem. It is possible to generalise this observation to a number of patterns when the standard, multi-segment, Harris approach will fail.

Consider the part of trie shown in figure 14 in which the successors of *Stem* are in $m+1$ subtrees, the one starting with the letter L, and m others $\{Tr_i\}_{i=1}^{m}$. None of these subtrees starts with the same letter, as required by definition. Let also $\{LTr_i\}_{i=1}^{n}$ be the subtrees following the path $Stem + L$. Then if $m + 1 > n$, Harris's method will always produce *Stem*, including the cases when $Stem + L$ is the correct stem. The following is an equivalent statement:

> If the word $W = Stem + L + Suf$ is segmented immediately after *Stem*, L is a single character, and W is not segmented after $Stem + L$ then any word $W' = Stem + L + Suf_{any}$ will also be segmented after *Stem*.
>
> In the opposite case, when $m + 1 < n$, the stem *Stem* will never be produced if the letter L follows (the case of *manger* shown in Table 16).

Another serious shortcoming of Harris's approach is related to the distance from the beginning (or end when applied from right to left) of the word segmented. The farther from

Table 16. Comparison between Harris's and GA segmentations.

Harris's segmentation	Short/long GA stem	GA segmentation
mange-ai	S	mange-ai
mange-ais	S	mange-ais
mange-ant	S	mange-ant
mange-ass-es	L	amangeass-es
mange-ass-i-ez	L	mangeass-iez
mange-ass-i-ons	L	mangeass-ions
mange-ons	L	mange-ons
*mange-r	S	mang-er
*mange-ra	L	manger-a
*mange-rai	L	manger-ai
*mange-raient	S	mang-eraient
*mange-rait	L	manger-ait
*mange-ras	L	manger-as
*mange-ron-s	L	manger-ons
*mange-ron-t	L	manger-ont
*mange-z	S	mang-ez
mange-âmes	S	mange-âmes
mange-âtes	S	mange-âtes
mang-iez	S	mang-iez
mang-ions	S	mang-ions
mang-èrent	S	mang-èrent
mang-é	S	mang-é
mang-ée	S	mang-ée
mang-ées	S	mang-ées

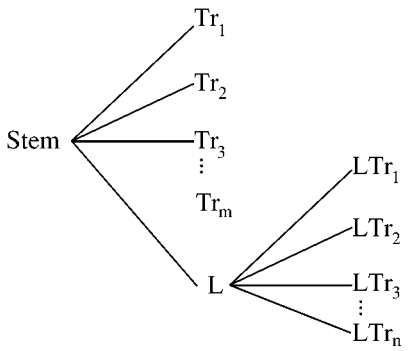

Figure 14. Stem alteration as handled by Harris.

the beginning, the less robust the method, as the number of words sharing the same first n letters decreases. At the extreme, for a word W of length k and a certain $n < k$, there will be no word in the corpus sharing the first n letters, and the word W will never be segmented after letter $n - 1$ even if there is a morpheme boundary. For instance, the correct segmentation of the noun `whatchamacallits` (pl.) is only possible (in left-to-right mode) if the singular form `whatchamacallit` is present in the corpus. In Harris's original paper (1955) this problem is avoided by prompting a human operator to provide an appropriate example, something clearly infeasible for large data sets.

Finally, a look at the data shows a small number of words, the segmentation of which cannot be generated by Harris's method, even if the words clearly contain several morphemes, as the number of branches in the trie never rises to a peak, for instance *placions, placiez, placèrent*, etc. The GA is usually able to handle the completely regular morphology of these words in a proper way.

8. Conclusions

The hybrid approach to unsupervised learning of word segmentation rules described in this paper is an efficient combination of GA and ILP. The former provides a first approximation of the concept learnt and reduces dramatically the search space. The latter learns rules that can be employed to segment unseen words. The results of the tests show that a set of rules for word segmentation can be learnt from a *limited* number of *unannotated* words, as opposed to other approaches where very large and/or tagged corpora are used. Indeed, the size of the corpora here is in the order of 10^3 words, which is well below the average used in statistical NLP.

Theoretical comparison with other learning strategies shows that some of them (Yvon, 1997) use small subsets of data to derive "microtheories" (cf. the analogy principle on p. 13) that are independent from the rest of the data. On the other hand, the best segmentation choice can be based on some integral criterion, such as MDL (Brent et al., 1995) or the constituent relative frequency in data (deligne,1996). The naïve theory of morphology (NTM) bias chooses a single segmentation that is related to the segmentation of most of the other words. In the limit, when only words with the same prefix (or suffix) are provided, the NTM bias, unlike the analogy principle (Pirelli, 1993; Yvon, 1997) is still able to produce the segmentation. Also, NTM does not need to impose a limitation on the length of affixes as other approaches (Brill, 1994; Deligne, 1996) do.

One should clearly distinguish the NTM bias for word segmentation from the GA. The former is a convenient simplification of the MDL principle, which only takes into account the size of the model (theory). The GA is an easy-to-implement algorithm for the search of the best NTM. Here the GA can be seen as a search technique in the MDL framework for word segmentation, which so far does not appear to have been exploited.

The NTM bias shows its feasibility in a number of cases when other approaches fail. One such case has been demonstrated for Harris's method. Also, the case shown in Eq. 2 in Section 3.1, which cannot be handled by de Saussure's analogy as used by Pirelli (1993) and Yvon (1997), is usually easily managed by the NTM bias.

Limitations of the approach are the computational complexity of the GA search for the best NTM and the "at most two constituents per word" model. One possibility would be to modify the NTM bias so that it would, as in Brent et al. (Brent et al.; 1995), take into account the amount of information needed to reconstruct the data set from the NTM. A shift in the encoding paradigm, and a modification of the MDL bias would be required to make possible the use of GA for the search of segmentations with more than two elements. The time efficiency that CLOG provided is a good premise for experimentation with larger data sets; bootstrapping techniques should also be considered for the generation of the segmentated list of words used by CLOG to learn rules. One could also explore the possibility of using our approach as a preprocessing phase of other morphology learning tasks, for instance, for the prediction of PoS of unknown words. However, the ability to obtain a good approximation of the best NTM representation of a lexicon seems to be the most important contribution to NLP so far as this is of theoretical interest.

From a machine learning point of view, probably the most important contribution of this work is the described combination of unsupervised and supervised learning that is novel, and with a potential range of applications out of the NLP area.

Acknowledgments

The authors are grateful to the anonymous referees for their detailed and highly useful comments and for pointing out Harris's work (1995). They also wish to thank Stephen Muggleton and James Cussens for the thoughtful remarks. The first author is indebted to Olga Štěpánková for a number of improvements suggested, and to François Yvon for the discussions. He has been supported by ESPRIT Long Term Research Action *No* 20237 on Inductive Logic Programming II and part of this research was carried out by him at CTU Prague.

Notes

1. For instance, NNP (proper noun), JJ (adjective), DET (determiner), etc.
2. at ESSLLI Summer School '96.
3. This presumption is limited to the languages in which the main operator used in derivational and inflectional morphological rules is `concatenation`.

References

Antworth, E. (1991). Introduction to two-level phonology. *Notes on Linguistics*, *53*, 4–18.

Bescherelle. (1990). *Laconjugaison dictionnaire de douze mille verbes*. Paris: Hatier.

Blockeel, H. (1994). *Application of inductive logic programming to natural language processing*. Master's Thesis, Katholieke Universiteit Leuven, Belgium.

Brent, M. (1999). An efficient, probabilistically sound algorithm for segmentation and word discovery. *Machine Learning*, *34*, 71–106.

Brent, M., Lundberg, A., & Murthy, S. (1995). Discovering morphemic suffixes: A case study in minimum description length induction. In *Proc. of the Fifth International Workshop on Artificial Intelligence and Statistics*. Ft. Lauderdale, FL.: Vanderbilt University.

Brill, E. (1994). Some advances in transformation-based part of speech tagging. In: *Proc. of the Twelfth National Conference on Artificial Intelligence* (pp. 748–753), AAI Press/MIT Press.

Chomsky, N. & Halle, M. (1968). *The Sound patterns in English*. New York: Harper and Row.

Collins, M. & Singer, Y. (1999). Unsupervised models for named entity classification. In *EMNLP-WVLC99*.

Cussens, J. (1997). Part-of-speech tagging using Progol. In *Proc. of the Seventh International Workshop on Inductive Logic Programming*. (pp. 93–108).

Daelamans, W., van den Bosch, A., & Weijters, A. (1997). IGTree: Using trees for compression and classification in lazy learning algorithms. *Artificial Intelligence Review*, *11*, 407–423.

de Saussure, F. (1916). *Course in general linguistics*. New York: Philosophical Library. 1959 edition.

Deligne, S. (1996). *Modèles de séquences de longueurs variables: Application au traitement du langage écrit et de la parole*. Ph.D. Thesis, ENST Paris, France.

Deligne, S. & Bimbot, F. (1997). Inference of variable-length linguistic and acoustic units by multigrams. *Free Speech Journal*, *1*(4). http://cslu.cse.ogi.edu/fsj/.

Fradin, B. (1994). L'approche à deux niveaux en morphologie computationnelle et les dévelopments récents de la morphologie. *Traitement automatique des langues*, *35*(2), 9–48.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley.

Harris, Z. S. (1955). From phoneme to morpheme. *Language*, *31*(2).

Havránek, B. & Jedlička, A. (1981). *Česká mluvnice*. Prague, Czechoslovakia: SPN.

Kaplan, R. M. & Kay, M. (1994). Regular models of phonological rule systems. *Computational Linguistics*, *20*(3), 331–379.

Kazakov, D. (1993). *Modul pro komunikaci v přirozeném jazyce*. Master's Thesis, Czech Technical University, Prague.

Kazakov, D. (1996). An inductive approach to natural language parser design. In K. Oflazer & H. Somers (Eds.), *Proceedings of NeMLaP-2*, Ankara (pp. 209–217). Bilkent University.

Kazakov, D. (1997). Unsupervised learning of naïve morphology with genetic algorithms. In W. Daelemans, A. Bosch, & A. Weijters (Eds.), *Workshop Notes of the ECML/MLnet Workshop on Empirical Learning of Natural Language Processing Tasks*. Prague, Czech Republic (pp. 105–112).

Kazakov, D. (2000a). Achievements and prospects of learning word morphology with inductive logic programming. In J. Cussens & S. Džeroski (Eds.), *Learning Language in Logic*. Berlin: Springer.

Kazakov, D. (2000b). *Natural language applications of machine learning*. Ph.D. thesis, Czech Technical University, Prague, Czech Republic.

Kazakov, D. & Manandhar, S. (1998). A hybrid approach to word segmentation. In D. Page (Ed.), *Proc. of the Eighth International Conference on Inductive Logic Programming*, Madison, Wisconsin, (pp. 125–134). Springer-Verlag.

Knuth, D. (1973). *The art of computer programming: Sorting and searching*, (Vol. 3). Reading, Massachusetts: Addison-Wesley.

Koskenniemi, K. (1983). *Two-level morphology: A general computational model for word-form recognition and production*. Finland: University of Helsinki, Dept. of General Linguistics.

Lavrač, N. & Džeroski, S. (1994). *Inductive logic programming techniques and applications*. Chichester: Ellis Horwood.

Ling, C. X. (1994). Learning the past tense of English verbs: The symbolic pattern associatior vs. connectionist models. *Journal of Artificial Intelligence Research*, *1*, 209–229.

Manandhar, S., Džeroski, S., & Erjavec, T. (1998). Learning multilingual morphology with CLOG. In D. Page (Ed.), *Proc. of The Eighth International Conference on Inductive Logic Programming*, Madison, Wisconsin, (pp. 135–144).

Matthews, P. H. (1974). *Morphology: An introduction to the theory of word-structure*. Cambridge University Press. First edition.

Matthews, P. H. (1997). *The concise oxford dictionary of linguistics*. Oxford University Press.

Mikheev, A. (1997). Automatic rule induction for unknown word guessing. *Computational Linguistics*, *23*(3), 405–423.

Mitchell, T. M. (1997). *Machine learning*. New York: McGraw-Hill.

Mooney, R. J. & Califf, M. E. (1995). Induction of first-order decision lists: Results on learning the past tense of English verbs. *Journal of Artificial Intelligence Research*, *3*, 1–24.

Muggleton, S. (1995). Inverse entailment and Progol. *New Generation Computing, 13*, 245–286.
Muggleton, S. & Bain, M. (1999). Analogical prediction. In *Proc. of the Ninth International Workshop on Inductive Logic Programming*, Bled, Slovenia. Springer-Verlag.
Pirelli, V. (1993). *Morphology, analogy and machine translation*. Ph.D. Thesis, Salford University, UK.
Quinlan, J. (1986). Induction of decision trees. *Machine Learning, 1*(1), 81–106.
Riloff, E. & Jones, R. (1999). Learning dictionaries for information extraction by multi-level bootstrapping. In *AAAI-99*, Orlando, FL (pp. 474–479).
Ritchie, G. D., Russel, G. J., Black, A. W., & Pulman, S. G. (1992). *Computational morphology: practical mechanisms for the English lexicon*. London: MIT.
Rumelhart, D. E. & McClelland, J. (1986). On learning the past Tense of English verbs. In *Parallel distributed processing* (Vol. II, pp. 216–271). Cambridge, MA: MIT Press.
Shannon, C. E. & Weaver, W. (1963). *The mathematical theory of communication*. Urbana: University of Illinois Press.
Skoumalová, H. (1997). Czech lexicon by two-level morphology. In R. Marcinkevičiene and N. Volz (Eds.), *Proceedings of the Second European Seminar of TELRI — "Language Applications for a Multilingual Europe,"* IDS/VDU, Mannheim/Kaunas (pp. 123–145).
Thompson, C. A., Califf, M. A., & Mooney, R. J. (1999). Active learning for natural language parsing and information extraction. In *ICML99* (pp. 406–414).
Yvon, F. (1997). Paradigmatic cascades: A linguistically sound model of pronunciation by analogy. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistic (ACL)*, Madrid.
Zelle, J. M. & Mooney, R. J. (1993). Learning semantic grammars with constructive inductive logic programming. In *Proceedings of AAAI-93* (pp. 817–822). Washington, DC: AAI Press/MIT Press.

Received April 2, 1999
Revised December 13, 1999 & April 12, 2000
Accepted May 10, 2000
Final manuscript June 15, 2000