Learning Word Segmentation Rules for Tag Prediction

Dimitar Kazakov¹, Suresh Manandhar², and Tomaz Erjavec³

University of York, Heslington, York YO10 5DD, UK,
{dimitar.kazakov,suresh}@cs.york.ac.uk.
WWW home page: ¹ http://www.cs.york.ac.uk/~dimitar/ and
² http://www.cs.york.ac.uk/~suresh/
³ Department for Intelligent Systems,
Jožef Stefan Institute, Ljubljana, Slovenia,
Tomaz.Erjavec@ijs.si.
WWW home page: http://nl.ijs.si/tomaz/

Abstract. In our previous work we introduced a hybrid, GA&ILP-based approach for learning of stem-suffix segmentation rules from an unmarked list of words. Evaluation of the method was made difficult by the lack of word corpora annotated with their morphological segmentation. Here the hybrid approach is evaluated indirectly, on the task of tag prediction. A pair of stem-tag and suffix-tag lexicons is obtained by the application of that approach to an annotated lexicon of word-tag pairs. The two lexicons are then used to predict the tags of unseen words in two ways, (1) by using only the stem and suffix generated by the segmentation rules, and (2) for all matching combinations of stem and suffix present in the lexicons. The results show high correlation between the constituents generated by the segmentation rules, and the tags of the words in which they appear, thereby demonstrating the linguistic relevance of the segmentations produced by the hybrid approach.

1 Introduction

Word segmentation is an important subtask of natural language processing with a range of applications from hyphenation to more detailed morphological analysis and text-to-speech conversion. In our previous work [5] we introduced a hybrid, GA&ILP-based approach for learning of stem-suffix segmentation rules from an unmarked list of words. Evaluation of the method was made difficult by the lack of word corpora annotated with their morphological segmentation. Here the quality of the segmentation rules learned with the hybrid approach is assessed, indirectly, through the task of morphosyntactic tag prediction.

Tag prediction of unknown words is an important preprocessing step performed by taggers. However, currently taggers either employ some simple heuristics for tag prediction based on the majority class tag, or word affixes [1]. In this paper we show that word segmentation information can be exploited to predict
the possible tags of unknown words with high accuracy. There are several methods which would not require the segmentation of training words to learn tag prediction rules from tagged lexicons. The tag prediction task is employed here to prove the close correlation between morphosyntactic tags and word segments produced by our rules. Success in this task would also imply the possibility of using those segments as a substitute for morphosyntactic tags when learning NLP tools from unannotated corpora.

An advantage of our approach is that it does not require a presegmented corpus for training. Instead, the system can be trained by supplying it with the same kind of lexicon of word-tag pairs as the one used in taggers.

In our previous work, we have described the hybrid approach combining unsupervised and supervised learning techniques for generation of word segmentation rules from a list of words. A bias for word segmentation [4] is reformulated as the fitness function of a simple genetic algorithm, which is used to search for the word list segmentation that corresponds to the best bias value. In the second phase, the list of segmented words obtained from the genetic algorithm is used as an input for CLOG [6], a first-order decision list learning algorithm. The result is a logic program in a decision list representation that can be used for segmentation of unseen words. Here an annotated lexicon of word-forms is used to assign morphosyntactic tags (or descriptions, MSDs) to each of the segments, and so build two annotated lexicons of stems and endings. The result is interpreted as a generative word grammar. The pertinence of this grammar is evaluated on the task of MSD prediction for unseen words, with and without the additional constraint of a single-word segmentation generated by the decision list learned in the previous step.

2 Overview of GA&ILP Learning of Segmentation Rules

This section provides a brief review of our hybrid GA&ILP approach, and for more details the reader should consult the paper in which the approach was first introduced [5].

2.1 Naïve Theory of Morphology as Word Segmentation Bias

Given a list of words segmented into stem-suffix pairs one can construct a pair of lexicons consisting of all stems, and suffixes, respectively.

The Naïve Theory of Morphology (NTM) bias [4] prefers segmentations which reduce the total number of characters \( N \) in the stem and suffix lexicons. The bias is based on the hypothesis that substrings composed out of real morphemes occur in the words with a frequency higher than any other left or right substrings.\(^1\) In that way, a theory with a low \( N \) would produce lexicons where 'stems' and 'suffixes' correspond very often to single morphemes or their concatenation. Since

\(^1\) This presumption is limited to the languages in which the main operator used to combine morphemes is concatenation.
the word list can be stored as a list of pairs of indices \(<\text{stem, suffix}>\) along with
the two lexicons, the bias described can be seen as using Occam’s razor to choose
the simplest theory corresponding to the given dataset.

## 2.2 Genetic Algorithms

Genetic algorithms (GA) [3] are often used as an alternative approach to tasks
with a large search space and multiple local maxima. A GA maintains a set of
candidate solutions called \textit{individuals} and applies the natural selection opera-
tors of \textit{crossover} and \textit{mutation} to generate, usually in several iterations, new
candidate solutions from existing ones. A \textit{fitness function} is employed to rank
the individuals to determine their goodness. The individuals are represented as
a sequence of characters of a given, often binary, alphabet. The \textit{crossover} oper-
ation constructs two new child individuals by splicing two parent individuals at
\(n\) points. The \textit{mutation} operator creates a new individual from a single parent
by randomly changing one of its characters. Individuals are mutated according
to some mutation probability known as \textit{mutation rate}.

The following algorithm, known as a \textit{simple genetic algorithm}, has been used for
the purposes of this research.

\textbf{Procedure simple genetic algorithm}

\begin{enumerate}
\item \textbf{Initialisation}
  \begin{enumerate}
    \item Create a random population of candidate solutions
      \textit{(individuals)} of size \textit{popsize}. \\
    \item Evaluate all individuals using the fitness function.
    \item Store the best evaluated individual as \textit{best-ever} individual.
    \item Set the number of generations to \textit{NG}.
  \end{enumerate}
\item \textbf{Generation and Selection}
  \begin{enumerate}
    \item \textbf{For} \textit{NG} generations \textbf{repeat}:
      \begin{enumerate}
        \item Sample the individuals according to their fitness, so that
          \textit{in the resulting mating pool those with higher fitness}
          \textit{appear repeatedly with a higher probability}.
        \item Apply \textit{crossover} with probability \textit{crossover rate}.
        \item Apply \textit{mutation} with probability \textit{mutation rate}.
        \item Evaluate all individuals using the fitness function.
        \item Update the \textit{best-ever} individual.
      \end{enumerate}
  \end{enumerate}
\item \textbf{Provide the best-ever individual as a solution}.
\end{enumerate}

## 2.3 GA Search for Best NTM

The so described genetic algorithm is used to search the space of possible seg-
mentations of given list of words and find a segmentation that is minimal with
respect to the NTM bias.

The representation of the list of segmented words in the GA framework is
straightforward. The position of the boundary between stem and suffix in a word
is represented by an integer, equal to the number of characters in the stem.
The segmentation of a list of words is represented as a vector of integers (see
Figure 1). A randomly generated population of such segmentations is used as a starting point of the GA search. The crossover operator constructs two new child chromosomes by splicing two parent chromosomes at one point. The mutation operator modifies the split position of a single word either by incrementing or decrementing by one the corresponding integer, or by changing it randomly within the word length.

2.4 Segmentation Rule Learning Using CLOG

The GA produces for a given list of words their segmentation, along with a pair of lexicons of stems and suffixes. Typically for GAs, the segmentation is only near-optimal w.r.t. the bias, i.e. the change of some segmentations would result in a better bias value. The reusability of the GA output for unseen words is limited. Indeed, one could use the lexicons to segment an unseen word into a stem and suffix present in the lexicons. However, if there is more than one such segmentation, there is no way to choose among them. In the hybrid GA&ILP approach, the ILP system CLOG is applied to learn segmentation rules which produce better segmentations than the GA alone, and can be used to find the best segmentation of unseen words.

CLOG [6] is a system for learning of first-order decision lists. CLOG can learn from positive examples only using the output completeness assumption [7], only considering generalisations that are relevant to an example. In the current implementation these generalisations are supplied by a user-defined predicate which takes as input an example and generates a hard-coded list of generalisations that cover that example. The gain function currently used in CLOG is user-defined. For the segmentation problem we chose the following simple gain function: $gain = QP - SN - C$ where $QP$ denotes the number of new examples covered positively, $SN$ denotes the number of previously covered examples that are covered negatively and $C$ is the number of literals in the clause body.

The words segmented with the GA are represented as clauses of the predicate $\text{seg}(W, P, S)$, for instance: $\text{seg}([a, n, t, o, n, i, m, i, h], [a, n, t, o, n, i, m], [i, h])$. 
Then \texttt{seg}/3 is used as a target predicate with mode \texttt{seg}(+,\,?,\,?) of the inductive learning algorithm. Also, the predicate \texttt{append}/3 is used as intentional background knowledge, and the range of the theory constants is limited to the set of stems and suffixes that appear in the predicate \texttt{seg}/3. In fact, either of the last two arguments of \texttt{seg}/3 is redundant and the second of them was omitted in the real input dataset, which resulted in the format \texttt{seg}([a,n,t,o,n,i,m,i,h],[i,h]). The result of ILP learning is an ordered list of rules (non-ground logic clauses) preceded by a list of exceptions represented as ground facts, such as the following example: \texttt{seg}([a,n,t,o,n,i,m],[i,m,a]). The exceptions do not have any impact on the segmentation of unseen words, and they are removed from the decision list. In most cases, exceptions correspond to imperfectly segmented words. When the segmentation rules, with the exceptions removed, are applied on the GA input list of words, the result is, in general, a segmentation with a better bias value. Figure 2 summarises schematically the GA&ILP approach.

3 Dataset

For our experiments we used part of the lexicon of the Slovene language created within the EU Copernicus project MULTEXT-East [2]. The project developed a multi-lingual corpus of text and speech data, covering six languages, including Slovene, and lexical resources covering the corpus data. The Slovene lexicon contains the full inflectional paradigms for over 15,000 lemmas; it has over half a million entries, where each entry gives the word-form, its lemma and morphosyntactic description. These descriptions are constructed according to the MULTEXT-East grammar, which follows international recommendations and is harmonised for seven languages. The MSDs contain all morphosyntactic features which are relevant to a given PoS. For the 7 parts of speech actually represented in the data used here, the number of features is as follows: Noun–8 features, Verb–9, Adjective–9, Pronoun–13, Adverb–3, Numeral–10, Abbreviation–1.
The MULTTEXT-East Slovene lexicon is freely available for research purposes. It includes a lexicon of neologisms from the Slovene translation of Orwell's 1984. From that lexicon, we used a list of 4383 different word-forms, i.e. with homonyms represented only once. The list was split at random into a training set of 3506 words and a test set containing 877 words. Next, the training set was divided into disjunctive lists of 100 words each and the genetic algorithm was separately run on each of them. Then those lists were merged again. This technique proves to be a feasible trade-off between the input data size and the time needed by the GA to find a segmentation of high quality. Task decomposition also makes the GA time complexity linear w.r.t. the input size. Indeed, if \( T \) is the time required to run the GA on a list of \( M \) words for a certain number of generations, then the time to apply the GA on a list of \( K \times M \) words will be approximately \( K \times T \) if the data set is divided into \( K \) separate chunks and the GA applied separately on each of them. The described decomposition also allows the individual GA runs to be run in parallel.

The list of segmented words so obtained was used as input of CLog. As a result, a first-order decision list was learned. The decision list contained 736 exceptions (Figure 3), and 242 rules (Figure 4). Only the rules were applied for the segmentation of the training list of words (cf. Figure 2), thus obtaining a segmentation with a better bias value.

%seg(Word, Suffix).
seg(["C", r, k, o, s, t, a, v, s, k, e], [k, o, s, t, a, v, s, k, e]) :- !.
seg(["C", i, t, a, m, o], [i, t, a, m, o]) :- !.
seg(["C", i, t, a, l], ["C", i, t, a, l]) :- !.

Fig. 3. Sample of decision list exceptions

Up to this point, no use whatsoever was made of the MSDs in the lexicon. In the final stage of data preparation, all MSDs which can be assigned to each word in the list were retrieved from the lexicon, and the 4-tuples word-sentence-suffix-possible MSD were stored as clauses of the predicate msd_train/4:

msd_train([a, d, o, p, t, i, v, n], [a, d, o, p, t, i, v, n], [i], ["A", f, p, m, s, a, y, \ldots , n]).

There are 10477 such clauses, i.e. each word in the training set corresponds in average to 3 morphosyntactic descriptions. The latter are very detailed, ranging from part of speech to up to 12 more features. The MSDs are represented as lists of one-letter constants, where the lists for a given PoS have the same length.

Similarly, the 877 words in the test set were annotated with all possible MSDs, producing as a result 2531 word-tag pairs tagged_lex(["C", e, s, t, e, m, u], ["A", f, p, m, s, d]).

Given the predicate msd_train/4, the following two lexicons can be generated:

---

2 For technical reasons, the characters č, š, and ž were replaced with 'C', 'S', and 'Z' respectively.
\[\text{seg}(A,B) :- \text{append}([s,u,p,e,r], B, A), !.\]

\[\text{seg}(A,B) :- \text{append}([s,u,g,e,r,i,r,a,j], B, A), !.\]

\[\text{seg}(A,B) :- \text{append}([s,u,g,e,r,i,r,a], B, A), !.\]

\[\text{seg}(A, [i]) :- \]
\[\quad \text{append}(_, [i], A), \]
\[\quad \text{append}(_, [i,t,e,i], A), !.\]

\[\text{seg}(A, [\_]) :- \text{append}(_, [\_], A), !.\]

\[\text{seg}(A, [\_]) :- \text{append}(_, [\_], A), !.\]

\[\text{seg}(A, [\_]) :- \text{append}(_, [\_], A), !.\]

\[\text{seg}(A, [\_]) :- \text{append}(_, [\_], A), !.\]

\[\text{seg}(A, [\_]) :- \text{append}(_, [\_], A), !.\]

\[\text{seg}(A, [\_]) :- \text{append}(_, [\_], A), !.\]

---

**Fig. 4.** Sample of segmentation rules

- *stem-tag* lexicon containing all stem-MSD pairs contained in \texttt{msd\_train/4}, e.g. \texttt{stem\_tag([a,d,o,p,t,i,v,n,i],[‘A’,f,p,m,s,a,y,-,n])}.
- *suffix-tag* lexicon containing all suffix-MSD pairs contained in \texttt{msd\_train/4}, e.g. \texttt{suffix\_tag([i],[‘A’,f,p,m,s,a,y,-,n])}.

### 4 Tag Prediction

#### 4.1 Method

The tag prediction task is to predict for a given word \(\bar{w}\) the set of all possible MSDs (i.e. tags). This task is broken down into the following stages:

**Segmentation** Split \(\bar{w}\) into stem \(Stm\) and suffix \(Suf\) by either method:

- **Method 1** using the segmentation rules generated by C1Log
- **Method 2** splitting the word into all possible stem-suffix pairs such that both the stem and suffix can be found in the *stem-tag* and *suffix-tag* lexicons.

**Tag prediction** Given a segmented word \(\bar{w} = Stm + Suf\), the prediction of the set of tags assigned to \(\bar{w}\) is mainly based on the MSDs in suf-tag lexicon which match the suffix. When the stem is present in the stem-tag lexicon, it is used as an additional constraint, limiting the number of MSD candidates.

**PoS matching** Produce the set of all MSDs selected by the suffix in the suf-tag lexicon, such that for each of them an MSD with the same PoS is selected by the stem in the stem-tag lexicon, i.e.:

\[\text{stem\_tag}(Stm,[\text{PoS}\_]),\text{suffix\_tag}(Suf,MSD),MSD\in[\text{PoS}])\]

**Suffix-based** If the previous step produces an empty set of MSDs, then its second constraint is dropped, and the set of tags generated is the set of all MSDs matching the suffix in the suf-tag lexicon: \text{suffix\_tag}(Suf, MS).

---
4.2 Evaluation

A variant of the well-known statistics precision and recall have been employed to evaluate the performance of our approach for tag prediction. First, some metric should be adopted for the purposes of comparison between predicted and correct MSDs. The simplest, binary yes/no metric which would count only perfect matches was considered too rigorous for the comparison of MSDs with up to 13 features. Instead, we employ a finer-grain metric based on the similarity function $sim(MSD_1, MSD_2)$ of two MSDs:

$$sim(MSD_1, MSD_2) = \#identical\ features$$

(1)

We extend this to the similarity between an MSD and a tag-set (i.e. a set of MSDs):

$$sim(MSD, TagSet) = \max \left( sim(MSD, MSD') \right), MSD' \in TagSet$$

(2)

For a given word $\hat{w}$, let the set of correct tags (MSDs) be $TagSet_c$, and the predicted set of tags $TagSet_p$. This similarity measure is incorporated in our definitions of precision and recall in the following way.

Let $L$ denote the test set of words.
Let $E$ denote the set of $(TagSet_c, TagSet_p)$ (correct-predicted) tagsets for every word $\hat{w}$ in $L$.

For any tagset $TagSet$, let $|TagSet|$ denote the total number of features in $TagSet$.

We define precision and recall as follows:

$$precision(E) = \frac{\sum_{(TagSet_c, TagSet_p) \in E} \sum_{MSD \in TagSet_p} sim(MSD, TagSet_c)}{|TagSet_p|}$$

(3)

$$recall(E) = \frac{\sum_{(TagSet_c, TagSet_p) \in E} \sum_{MSD \in TagSet_c} sim(MSD, TagSet_p)}{|TagSet_c|}$$

(4)

In other words, precision shows how closely the predicted tags match the gold standard. To compute precision, for each of the predicted MSDs the best match (the most similar MSD) is found in the set of correct tags, then the overall similarity for all predicted tags is found, and it is divided by the total number of features. Similarly, accuracy shows how well the correct tags are represented in the set of predicted tags.

4.3 Results

Using Segmentation Method 1 The segmentation rules learned with CLOG were applied on the test data. Out of 877 different words in the test set, 858
or 97.83% were covered by the rules. For the successfully segmented words, predictions were made for the set of corresponding MSDs. For each word in the test set, the set of predicted MSDs was in average 3.89 times larger than the set of correct MSDs (the set of correct MSDs contains in total 21742 features, as opposed to 84612 in the MSDs predicted). The figures for precision (82.35%) and recall (91.69%), as defined in Equations 3–4, can be interpreted as follows:

1. Precision can be seen as the correctness of the predictions made, whereas recall quantifies the ability to produce as many of the correct tags as possible. The result precision < recall means that our approach performs better on the latter task than on the former one, i.e., it is slightly over-general, more careful not to reject a correct MSD than not to generate an incorrect one. So, the metric used has a simple and plausible interpretation.

2. A high percentage of the MSDs predicted have a close match in the set of correct MSDs. Since |MSD_p| > |MSD_c|, that also means that many of the predicted MSDs are very similar to each other, differing only in a small percentage of features.

Using Segmentation Method 2 The results for this experiment are as follows: precision = 48.28%, recall = 99.50%.

5 Conclusions

This article introduces a method using a lexicon annotated with morphosyntactic features to learn rules for the prediction of those features for unseen words. The article also demonstrates the strength of the hybrid GA&ILP approach in learning segmentation rules from unannotated words. The main advantages of the approach are threefold. Firstly, the lexicons of stems and suffixes produced by the segmentation rules learned can reliably capture the information relevant to the word morphosyntactic tags. This can be seen from the 99.50% recall for segmentation method 2, where all combinations of stems and suffixes with matching MSDs were used to predict word tags. The second contribution of the hybrid approach is that it learns rules assigning a single segmentation to each covered word. The additional information that this segmentation brings to the tag prediction task is reflected in the considerable increase in precision. Finally, the hybrid approach only requires a relatively small list of unannotated words (10^5 to 10^6 as compared to the annotated corpora of 10^8 words used by Brill [1]) to learn segmentation rules, which can be used either for the segmentation of the words used for learning, or to segment unseen words. The unsupervised framework makes the application of the hybrid approach to word segmentation a possible way to apply corpus-based NLP methods requiring morphosyntactic tags to unannotated corpora. As the word constituents produced by the hybrid approach are closely related to the morphosyntactic features of the words, tagging the words in a corpus with their constituents produced by the segmentation rules could serve as a substitute for missing morphosyntactic tags.
6 Acknowledgements

The first author has been supported for this research by ESPRIT Projects 20237 ILP2, and 28623 ALADIN.

References