Weakly Supervised Morphology Learning for Agglutinating Languages Using Small Training Sets

Ksenia Shalonova
Computer Science,
University of Bristol
ksenia@cs.bris.ac.uk

Bruno Golénia
Computer Science,
University of Bristol
csbsgg@bristol.ac.uk

Abstract

The paper describes a weakly supervised approach for decomposing words into all morphemes: stems, prefixes and suffixes, using wordforms with marked stems as training data. As we concentrate on under-resourced languages, the amount of training data is limited and we need some amount of supervision in the form of a small number of wordforms with marked stems. In the first stage we introduce a new Supervised Stem Extraction algorithm (SSE). Once stems have been extracted, an improved unsupervised segmentation algorithm GBUMS (Graph-Based Unsupervised Morpheme Segmentation) is used to segment suffix or prefix sequences into individual suffixes and prefixes. The approach, experimentally validated on Turkish and isiZulu languages, gives high performance on test data and is comparable to a fully supervised method.

1 Introduction

The major function of morphological analysis is decomposition of words into their constituents - stems and prefixes/suffixes. In recent years Machine Learning approaches were used for word decomposition. There is a number of both unsupervised morphology learning systems that use ”raw” wordforms as training data (Creutz and Lagus, 2002; Goldsmith, 2001; Kazakov and Manandhar, 2001) and supervised morphology learning systems using segmented wordforms into stems and affixes as training data (Oflazer et al., 2001). The supervised morphology learning systems are usually based on two-level morphology (Koskenniemmi, 1983). There is also a weakly supervised approach that uses, for example, wordpairs as input, and this was applied mainly to fusional languages for stem extraction (Erjavec and Dzeroski, 2004). Our project concerns developing speech technology for under-resourced languages. For this type of applications we need a relatively fast, cheap (i.e. does not require large training sets), almost knowledge-free approach that gives high performance. We have chosen to use wordforms with marked stems as training data in order to fulfill the criteria mentioned above.

Morphological analysis is used in many practical Natural Language Processing applications such as Machine Translation, Text Mining, spell-checkers etc. Our near-term goal is the integration of the morphology learning algorithms into the language-independent Text-to-Speech (TTS) system for improvement of grapheme-to-phoneme rules, stress prediction and tone assignment. In particular, the morphology learning algorithms described in this paper will be incorporated into the available isiZulu TTS system for automatic prediction of lexical tones. In the isiZulu language lexical tone assignment depends on the morpheme boundary. The current isiZulu TTS system is tone-deaf due to the lack of morphological decomposition. A number of perception tests will be carried out in order to evaluate which performance of morphology decomposition is acceptable for TTS and will improve the quality of the synthesised speech. It seems that the unsupervised morphology learning systems can be relatively easy to implement from scratch, but their performance probably cannot be regarded as high enough to improve the performance of the synthesized speech. In order to overcome this problem we present a novel synthesis of supervised and unsupervised induction techniques for morphology learning.

Our approach consists of two parts: the new supervised stem extraction algorithm for agglutinating languages and the improved version of the unsupervised algorithm for segmenting affix sequences. In (Shalonova et al., 2009) the authors presented the function learning approach called TASR (Tree of Aligned Suffix Rules) for extracting stems in fusional languages given wordpairs (word in grammatical form - word in basic form). While this algorithm gives good performance for Russian and English, it gives quite poor performance for agglutinating languages as shown in Section 4. A new approach for stem extraction in agglutinating languages is required for two main reasons. Firstly, suffix (or prefix) sequences in agglutinating languages can be much longer than in fusional languages and TASR does not seem to be efficient on long affix sequences as it does not generalise data in the efficient way and generates too many specific rules. This leads to poor performance on unseen data. Secondly, in some agglutinating languages it could be easier for native speakers to provide a stem (i.e. to provide a list of wordforms with annotated stems), whereas in highly inflective fusional languages the stem is often strongly bound with suffix sequences, and providing a proper stem requires high linguistic expertise. TASR approach is more appropriate for word-and-paradigm or realizational morphology that focuses on the whole word form rather than on word segmentation. For example, in Russian the infinitive verb *govorit'* ('to speak') generates a set of grammatical forms or a paradigm - *govorivshij*, *govor'aschij*, *govorim* etc.

The second part of our approach is the improved version of GBUAS algorithm (Shalonova et al., 2009) that provides affix segmentation given unannotated affix sequences. Given stem boundaries in the training set, our method splits the input word into all morphemes: stems and prefixes/suffixes. Our two-stage approach is tested on the under-resourced language isiZulu containing both prefixes and suffixes, as well as on Turkish containing only suffixes. Turkish is the most commonly spoken of the Turkic languages (over 77 million people). isiZulu is the Bantu language with about 10 million speakers and it is the most widely spoken home language in South Africa. Both Turkish and isiZulu use agglutination to form new words from noun and verb stems. In comparison to Turkish, isiZulu is a tonal language. In contrast to East Asian languages, in isiZulu there are three steps for tone assignment: lexical, morphemic and terraced. For TTS the lexical and morphemic tones will need to be recovered from the lexicon and the grammar as the orthography has no tone marking. The terraced tone relation can in general be recovered and marked automatically from the tone sequence with a finite state model.

2 Stem Extraction Algorithm

The Stem Extraction Algorithm (SSE) is the supervised algorithm for stem extraction. The training data for the SSE represent wordforms with the marked stem boundaries. During the training stage we collect a set of all possible stem extraction rules from training data and assign precision measures to each rule. Each rule is of the form *L_R* where "_" is the stem boundary, L and R are the left and right graphemic contexts of a stem boundary of different lengths. We differentiate prefix *Lpref_Rstem* and suffix *Lstem_Rsuff* stem extraction rules that correspond to the rules containing the left-hand stem boundary and the right-hand stem boundary respectively. For example, the Turkish word *yer* ('earth') with the marked word boundary #ye_r# generates the following *Lstem_Rsuff* rules: #ye_r#, #ye_r, ye_r#, #ye_, ye_r, e_r#, _r#, ye_, e_r, _r, and e_, where the symbol '#' signifies the word initial and final positions. We are implementing similar feature vectors used for automatic pronunciation prediction based on the focal grapheme (in our case it is a stem boundary) and left/right graphemic contexts of different length (Davel and Barnard, 2008). The idea of implementing expanding context in NLP tasks is usually applied for two-level data like grapheme-to-phoneme mapping rules (Torkkola, 1993), whereas in our case we use it for one-level data.

The precision measure for each rule is calculated by the formula $p/(p+n+\varepsilon)$ where p and n are the number of positive and negative examples, and ε is used to cover the cases where there are no negative examples. A high precision is desirable and this occurs when there are high values of p and low values of n (i.e. many positive examples and few negative examples). Using negative examples in contrast to using only rule frequencies (or positive examples) improves the performance of the algorithm.

Definition 1. *The number of positive examples for the rule Lstem_Rsuff (or rule Lpref_Rstem) is the number of training instances of $Stem\_Suffixes$ (or $Prefixes\_Stem$) containing the substring L_R.*

Definition 2. *The number of negative examples for rule Lstem_Rsuff (or Lpref_Rstem) is the number of training instances $Stem\_Suffixes$ (or $Prefixes\_Stem$) such that $Stem + Suffixes$ (or $Prefixes + Stem$) contains substring L+R and $Stem\_Suffixes$ (or $Prefixes\_Stem$) does not contain substring L_R where '+' denotes string concatenation.*

In the above definitions '_' is a stem boundary.

Example 1. *Suppose we have only three isiZulu verbs: zi_bek_e, zi_nak_eke and a_hlul_eke. For the $Lstem\_Rsuff$ rule* ***'ek_e'****, the word zi_bek_e generates one positive example and the two other words zi_nak_eke and a_hlul_eke generate one negative example each.*

The approach given in Algorithm 1 aims to find the unique longest rule-pair '$Lpref\_Rstem$ and $Lstem\_Rsuff$' with the highest precision that is applied to the input wordform for stem extraction. In case the language does not have prefixes like Turkish, the longest rule $Lstem\_Rsuff$ with the highest precision is applied. The decision of using either a rule-pair or just a single suffix rule is influenced by prior knowledge that a particular language has got either both prefixes and suffixes like isiZulu or only suffixes like Turkish. From now on we will use the term 'rulepair' in application both to the rulepair '$Lpref\_Rstem$ and $Lstem\_Rsuff$' in case of isiZulu and to the rulepair ' and $Lstem\_Rsuff$' with an empty first element in case of Turkish.

Algorithm 1 Choosing rule pair for stem extraction.

```
input W = raw wordform; $P$ and $S$ are sets of unique $Lpref\_Rstem$ and $Lstem\_Rsuff$ rules
output $result\_rule\_pair$

$result\_rule\_pair \leftarrow \emptyset$
iMaxlength $\leftarrow \infty$
repeat
   $(p1,s1) \leftarrow$ getrulepair (P $\times$ S, W, iMaxlength)
   $(p2,s2) \leftarrow$ getrulepair (P $\times$ S $\setminus$ $(p1,s1)$, W, iMaxlength)
   iMaxlength $\leftarrow$ length$(p1,s1)$
until $(p1,s1) = \emptyset$ or precision $(p1,s1) \neq$ precision$(p2,s2)$ or length $(p1,s1) \neq$ length$(p2,s2)$
$result\_rule\_pair \leftarrow (p1,s1)$

function getrulepair(PS, W, iMaxlength)
ilength $\leftarrow 0$
r $\leftarrow \emptyset$
for all $(p,s) \in PS$ do
   if $(p,s)$ matches W then
      if length$(p,s) < iMaxlength$ and length$(p,s) >$ ilength then
         ilength $\leftarrow$ length$(p,s)$
         $r \leftarrow (p,s)$
      else
         if length$(p,s) =$ ilength and precision$(p,s) >$ precision$(r)$ then
            $r \leftarrow (p,s)$
         end if
      end if
   end if
end for
return $r$
end function
```

The search is carried out on the set of rule pairs matching an input raw wordform. The set is sorted by length first, and then by precision measure within each length category.

For example, if rulepairs have the following length-precision values:
'4-0.5,'4-0.5','4-0.2'

'3-0.4','3-0.3'
'2-0.3'
rulepair with the value 3-0.4 is selected.

The rulepair matches the input word if $Lpref\_Rstem$ and $Lstem\_Rsuff$ rules can be applied without contradicting each other. For example, the rule pair '#a_hl' and 'l_eke' matches the word *a_hlul_eke*, whereas the rule pair '#a_hlulek' and 'le_ke' does not match this word. For each input wordform the set of its own rulepair candidates is generated. The search in the algorithm among these rulepairs starts from the longest rulepairs, and this allows more specific rules and exceptions to be applied first, whereas the more general rules are applied if no specific rules cover the input wordform.

3 Graph-Based Unsupervised Morpheme Segmentation

In this section we extend GBUMS (Graph-Based Unsupervised Morpheme Segmentation) that segments sequences of prefixes and suffixes (Golénia et al., 2009). We propose an extension of GBUMS which uses the graph structure of GBUMS through a brute-force method. Our experiments showed the improved results on training set and allowed GBUMS to be run on the test sets for two languages: Turkish and isiZulu.

The algorithm GBUMS was originally developed in (Shalonova et al., 2009) under the name GBUSS (Graph-Based Unsupervised Suffix Segmentation) to extract suffix sequences efficiently and it was applied to Russian and Turkish languages on training sets. We refer to prefixes and suffixes generally as morphemes. GBUMS uses a morpheme graph in a bottom-up way. Similar to Harris (Harris, 1955), the algorithm is based on letter frequencies. However, when Harris uses successor and predecessor frequencies, they use position-independent *n*-gram statistics to merge single letters into morphemes until a stopping criterion is fulfilled.

In the morpheme graph, each node represents a morpheme and each directed edge the concatenation of two morphemes labelled with the frequencies in a M-corpus (see Figure 1). M-corpus is a list of morpheme sequences

Definition 3. *Let $M = \{m_i | 1 \leq i \leq |M|\}$ be a set of morphemes, let f_i be the frequency with which morpheme m_i occurs in a M-corpus of morpheme sequences, let $v_i = (m_i, f_i)$ for $1 \leq i \leq n$, and let $f_{i,j}$ denote the number of morpheme sequences in the corpus in which morpheme m_i is followed by morpheme m_j. The* morpheme graph $G = (V,E)$ *is a directed graph with vertices or nodes $V = \{v_i | 1 \leq i \leq |V|\}$ and edges $E = \{(v_i, v_j) | f_{i,j} > 0\}$. We treat $f_{i,j}$ as the label of the edge from v_i to v_j.*

In G, each node is initialised with a letter according to a M-corpus, then one by one, nodes are merged to create the real morphemes. To merge nodes, an evaluation function is required. In (Golénia et al., 2009), Golenia et al. employed the *Morph_Lift* evaluation function based on its relation to the *lift* of a rule for association rules in data mining (Brin et al., 1997).

Definition 4. *$Morph\_Lift$ is defined for a pair of morphemes m_1 and m_2 as follows:*

$$Morph\_Lift(m_1, m_2) = \frac{f_{1,2}}{f_1 + f_2} \quad (1)$$

From now on, we know how to merge nodes. Now, we need to figure out the most important part of GBUMS, which is the stopping criterion. The stopping criterion is to prevent overgeneralisation. In other words, the algorithm needs to be stopped before getting the initial M-corpus (since no merging is possible). This criterion is based on the Bayesian Information Criterion (BIC) and Jensen-Shannon divergence (Li, 2001).

BIC is used for selecting a model (set of morphemes) which fits a data set (M-Corpus) without being too complex. We want to point out that BIC is related to MDL. BIC is a trade-off between the maximum likelihood, the parameters of the model (probability and length of each morpheme) and the number of elements in the data set (frequency of each morpheme). A smaller value of BIC corresponds to a better model fit. The maximum of the Jensen-Shannon divergence is used in order to analyse the increase of log-likelihood among all possible models. The Jensen-Shannon divergence is defined as follows (Dagan et al., 1997):

Definition 5. *The* Jensen-Shannon divergence *is defined for two morphemes m_1 and m_2 as the de-*

crease in entropy between the concatenated and the individual morphemes:

$$D_{JS}(m_1,m_2) = H(m_1 \cdot m_2) - \frac{L_{m_1}H(m_1) + L_{m_2}H(m_2)}{N} \quad (2)$$

where $H(m) = -P(m)\log_2 P(m)$ $N = \sum_m Freq(m)$ *and* L_m *is the string length of* m*.*

Stopping criterion *requires that* $\Delta BIC < 0$ *which translates to:*

$$\max_{m_1,m_2} D_{JS}(m_1,m_2) \leq 2\log_2 N \quad (3)$$

Algorithm 2 The GBUMS morpheme segmentation algorithm

```
input  M-Corpus = List of Strings
output  M-CorpusSeg = List of Strings
  M-CorpusSeg ← SegmentInLetters(M-Corpus);
  Graph ← InitialiseGraph(M-CorpusSeg);
  repeat
    Max ← 0;
    for all (p,q) ∈ Graph do
      ML_Max ← Morph_Lift(p, q);
      if ML_Max > Max then
        Max ← ML_Max;
        pMax ← p;
        qMax ← q;
      end if
    end for
    Graph ← MergeNodes(Graph, pMax, qMax);
    M-CorpusSeg ← DeleteBoundaries(M-CorpusSeg, Label(pMax), Label(qMax));
    Graph ← AdjustGraph(M-corpusSeg, Graph);
  until StoppingCriterion(pMax, qMax, Max)
```

After several merging iterations, the output of the algorithm is the graph shown in Figure 1. The GBUMS is presented in Algorithm 2.
Note that the M-Corpus is completely segmented at the beginning of the algorithm. Then, the boundaries in the segmented M-Corpus are removed step by step according to a pair found in the graph with the maximum value for $Morph\_Lift$. When the stopping criterion is fulfilled, the segmented M-Corpus represents the morpheme sequences.

At this point we present our extension of GBUMS based on a brute-force heuristic which scores every possible segmentation of an input morpheme sequence using graph values. We consider the morpheme graph as a model where each morpheme sequence can be extracted by the *MGraph* function (eq. 4).

Definition 6. *We define MGraph of a morpheme sequence without boundaries x as follows:*

$$MGraph(x) = \arg\max_{t \subseteq x} \frac{1}{N_t - C_t} \sum_{m \in t} L_m log(f_m + 1) \quad (4)$$

where

- t *is a morpheme sequence with boundaries of* x*,*
- m *is a morpheme of* t*,*
- f_m *is the frequency of the morpheme* m*,*
- N_t *is the number of morphemes existing in the graph,*
- C_t *is the number of morphemes existing and contiguous in the graph.*

Firstly, as a post-processing procedure the *MGraph* function improves the performance on training data. Secondly, it permits the identification of unseen morphemes. That is why the model generated by GBUMS can be run on test data sets.

Example 2. *Let our final morpheme graph be as shown in Figure 1 where nodes represent suffixes and their frequencies.*
Let x="ekwe" be our input suffix sequence that we want to segment into individual suffixes. We split this input sequence into all possible substrings from individual characters up to size of the input string length: e-k-w-e, e-k-we, e-kw-e, ek-w-e, ..., ekwe.
Using equation 4, we evaluate each substring and select the one with the highest score as the correct segmentation. Here, we have 7 potential segmentations with a score higher than 0 (MGraph > 0), e.g: e-k-w-e = $(\log(3) + \log(3))/2 = 1.0986$, *ek-w-e* $= (2\log(4) + \log(3))/2 = 1.9356$ *and ek-we* =

$2\log(4) = 2.7726$.
Consequently, ek-we is chosen as the correct segmentation for the substring "ekwe".

We would like to highlight that our new method can identify unseen cases with M-Graph, for instance, in the previous example suffix "we" was not present in the training graph, but was correctly extracted.

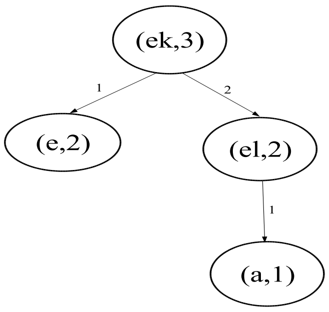

Figure 1: Example of a suffix subgraph in the training phase for isiZulu.

4 Results

Our experiments were based on Turkish data containing 1457 verbs and 2267 nouns, and isiZulu data containing 846 nouns and 931 verbs, with one single unambiguous segmentation per word.[1] Both isiZulu and Turkish data were uniquely sampled from the most frequent word lists.

Our first experiments compared TASR and the new SSE algorithm for stem extraction (10-fold cross validation assumes the following training and test set sizes: training sets containing 1311 wordforms for verbs and 2040 wordforms for nouns; test sets containing 146 wordforms for verbs and 227 wordforms for nouns). As can be seen from the Table 1, the performance of the SSE algorithm on Turkish data is much higher than that of TASR on the same dataset. As we mentioned in Section 1, TASR is not suitable for agglutinating languages with long suffix sequences. Although TASR algorithm gives an excellent performance on Russian, for most Turkish words it fails to extract proper stems.

[1]In agglutinating languages some wordforms even within one POS category can have several possible segmentations.

	Test	**FMea**
TASR	Nouns	20.7±6.8
	Verbs	12.6±5.9
SSE	Nouns	84.3±3.2
	Verbs	82.1±3.7

Table 1: Comparison of TASR and SSE for Turkish using 10-fold cross validation.

Our next experiments evaluated the performance of GBUMS on its own given unsegmented suffix sequences from Turkish nouns and verbs as training data. The performance on these training data increased by approximately 3-4 % in comparison to the results presented in (Shalonova et al., 2009). We would like to point out that the results in (Shalonova et al., 2009) are based on training data rather than on test data, whereas in the current paper we run our algorithms on test (or unseen) data. Our final experiments examined performance on the test sets and were run both on Turkish and isiZulu data. We compared our approach with Morfessor run both in supervised and in unsupervised mode. Although Morfessor is known as one of the best unsupervised morphology learning systems, it is possible to run it in the supervised mode as well (Spiegler et al., 2008). The training data for SSE+ GBUMS contained wordforms with marked stems. During training stage the SSE algorithm was collecting information about stem boundaries and the GBUMS algorithm was run on unlabelled suffix and prefix sequences from the same training set. The test stage for the SSE+GBUMS approach was run on "raw" wordforms by applying the SSE algorithm first for stem extraction and then running GBUMS algorithm for segmenting prefix or suffix sequences after the SSE has extracted stems. Training data for supervised Morfessor used the same wordforms as for the SSE+GBUMS training set and contained wordforms segmented into stems and affixes (i.e. words segmentated into all morphemes were given as training data). The test data for supervised Morfesor were the same as those used for SSE+GBUMS. Morfessor in unsupervised mode was run on "raw" wordforms as training data. To evaluate our current work we ap-

	Test	FMea
Supervised Morfessor	Nouns	74.6±2.3
	Verbs	84.5±2.2
SSE+ GBUMS	Nouns	78.8±2.4
	Verbs	76.9±0.7
Unsupervised Morfessor	Nouns	26.6±2.6
	Verbs	28.4±2.8

Table 2: Comparison of Morfessor and SSE+GBUMS for Turkish using 10-fold cross validation.

	Test	FMea
Supervised Morfessor	Nouns	76.7±1.6
	Verbs	88.5±2.4
SSE+ GBUMS	Nouns	87.9±1.9
	Verbs	84.5±2.5
Unsupervised Morfessor	Nouns	27.4±5.1
	Verbs	26.9±5.0

Table 3: Comparison of Morfessor and SSE+GBUMS for isiZulu using 10-fold cross validation.

plied the SSE+GBUMS approach for the under-resourced agglutinating language isiZulu containing both prefixes and suffixes and for Turkish containing only suffixes. The results show (Table 2 and Table 3) that our weakly supervised approach is comparable with the supervised Morfessor and decisively outperforms the unsupervised Morfessor. We think that it is useful to point out that unsupervised morphology learning systems in general require much larger training sets for better performance. F-measure is the harmonic mean of precision and recall, whereas precision is the proportion of true morpheme boundaries among the boundaries found, recall is the proportion of boundaries found among the true boundaries. In our experiments the GBUMS algorithm had no restrictions on affix length (Shalonova et al., 2009), but if there were restrictions, performance could be better. For isiZulu nouns our approach significantly outperformed supervised Morfessor, whereas for Turkish verbs SSE+GBUMS performed much worse. The best overall results obtained by GBUMS were based on the isiZulu nouns where about **53%** of all affixes were single letter affixes, whereas the worst results our approach gave for Turkish verbs where only about **12%** of affixes are composed of one letter. It is important to notice that the GBUMS algorithm, which is completely unsupervised, gives better results for extracting one letter affixes compared to Morfessor.

5 Conclusions

In the paper we described a weakly supervised approach for learning morphology in agglutinating languages. We were successful in our ultimate goal of synthesis of supervised and unsupervised induction techniques by achieving high performance on small amount of training data. Our weakly supervised approach is comparable with the supervised morphology learning system. As we are working with the languages for which linguistic resources are very limited (in particular words with morpheme boundaries), the developed method fulfills our goals of providing key components for speech and language products for such under-resourced languages. We speculate that the current performance might be improved by adding a small amount of completely ”raw” data to the training set.

The integration of our algorithms into working TTS systems is of key importance. As our near-term goal is the integration of morphology learning component into the currently working isiZulu TTS system, we will have to analyse the necessity of a Part of Speech Tagger (POS) and morphological disambiguation. In agglutinating languages some wordforms can be segmented in different ways (i.e. have different surface forms) and Machine Learning approaches normally select the most probable segmentation, and therefore our morphology disambiguation can be important. Morphological disambiguation for TTS can be considered a less complex problem than full morphological disambiguation as it can be linked, for example, to lexical tone disambiguation that may not require the full POS tag set. We intend to carry out user perception tests in order to evaluate the possible improvement in the isiZulu TTS quality after morphology information is added.

6 Acknowledgment

We would like to thank Kemal Oflazer from Sabanci University in Istanbul for his help and the Turkish data. We also thank Viktor Zimu and Etienne Barnard from CSIR in Pretoria for providing us isiZulu data. We also thank Roger Tucker for his support in the project. The work was sponsored by the EPSRC Grant EP/E010857/1 'Learning the morphology of complex synthetic languages'.

References

Brin, S., R. Motwani, J. Ullman, and S. Tsur. 1997. Dynamic itemset counting and implication rules for market basket data. In *ACM SIGMOD international conference on Management of data*, pages 255–264. ACM.

Creutz, M. and K. Lagus. 2002. Unsupervised discovery of morphemes. *Proceedings of the Workshop on Morphological and Phonological Learning of ACL-02*, pages 21–30.

Dagan, I., L. Lee, and F. Pereira. 1997. Similarity-Based Methods for Word Sense Disambiguation. *Thirty-Fifth Annual Meeting of the ACL and Eighth Conference of the EACL*, pages 56–63.

Davel, M. and E. Barnard. 2008. Pronunciation prediction with default refine. *Computer Speech and Language*, 22:374–393.

Erjavec, T. and S. Dzeroski. 2004. Machine learning of morphosyntactic structure: Lemmatising unknown Slovene words. *Applied Artificial Intelligence*, 18(1):17–40.

Goldsmith, J. 2001. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27:153–198.

Golénia, B., S. Spiegler, and P. Flach. 2009. UNGRADE: UNsupervised GRAph DEcomposition. In *Working Notes for the CLEF 2009 Workshop, CLEF 2009, Corfu, Greece.*

Harris, Z. 1955. From Phoneme to Morpheme. *Language*, 31(2):190–222.

Kazakov, D. and S. Manandhar. 2001. Unsupervised learning of word segmentation rules with genetic algorithms and inductive logic programming. *Machine Learning*, 43:121–162.

Koskenniemmi, K. 1983. *Two-level Morphology: A General Computational Model for Word Form Recognition and Production*. Ph.D. thesis, University of Helsinki.

Li, W. 2001. New stopping criteria for segmenting DNA sequences. *Physical Review Letters*, 86(25):5815–5818.

Oflazer, K., M. McShane, and S. Nirenburg. 2001. Bootstrapping morphological analyzers by combining human elicitation and machine learning. *Computational Linguistics*, 27(1):59–85.

Shalonova, K., B. Golenia, and P. Flach. 2009. Towards learning morphology for under-resourced languages. *IEEE Transactions on Audio, Speech and Language Procesing*, 17(5):956–965.

Spiegler, S., B. Golenia, K. Shalonova, P. Flach, and R. Tucker. 2008. Learning the morphology of Zulu with different degrees of supervision. *IEEE Spoken Language Technology Workshop*, pages 9–12.

Torkkola, K. 1993. An efficient way to learn English grapheme-to-phoneme rules automatically. *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 199–202.