UNGRADE:UNsupervised GRAph DEcomposition

Bruno Golénia, Sebastian Spiegler and Peter Flach
Machine Learning Group, Computer Science Department, University of Bristol, UK
{goleniab, spiegler, flach}@cs.bris.ac.uk

Abstract

This article presents an unsupervised algorithm for word decomposition called UNGRADE (UNsupervised GRAph DEcomposition) to segment any word list of any language. UNGRADE assumes that each word follows the structure prefixes, a stem and suffixes without giving a limit on the number of prefixes and suffixes. The UNGRADE's algorithm works in three steps and is language independent. Firstly, a pseudo stem is found for each word using a window based on Minimum Description Length. Secondly, prefix sequences and suffix sequences are found independently using a graph algorithm called graph-based unsupervised sequence segmentation. Finally, the morphemes from previous steps are joined to provide a segmented word list. We focus purely on the segmentation of words, thus, we employ a trivial method for labeling each morpheme which is the segment of the morpheme itself. UNGRADE is applied to 5 languages (English, German, Finnish, Turkish and Arabic) and results are provided according to their gold standard.

Categories and Subject Descriptors

I.2 [**ARTIFICIAL INTELLIGENCE**]: H.3.1 Learning; I.2.7 Natural Language Processing; I.2.8 Graph and tree search strategies

General Terms

Algorithms, Performance, Experimentation

Keywords

MDL, Unsupervised learning, Word morphology, Word decomposition, Stopping criterion

1 Introduction

Morphological analysis is concerned with the process of segmenting a given corpus of words into a set of morphemes. Morphemes are the smallest units bearing a meaning in a word. In a corpus, the quantity of different morphemes is usually smaller than the quantity of different words. The purpose of the Morpho Challenge is to learn these morphemes using an unsupervised algorithm on different languages. Such a morphological analyser is useful for various applications like speech synthesis, information retrieval or machine translation where a dictionary of morphemes must be provided. Indeed, creating a dictionary of morphemes for a speech synthesis application cannot be directly carried out from raw data on all languages. The reason is that for languages like Turkish, there are many possible combinations of morphemes, therefore the number of words becomes too large to be used naively as a dictionary in a application. Basically, using millions of words as a dictionary instead of set of morphemes is not feasible. The development of a such morphological analyser to segment words can be realised by linguistic experts or with a machine. Unfortunately, linguistic experts are rare and an expensive resource for some languages such as Zulu. As a result, it is worth to study machine learning approaches to reduce the quantity of work needed to create a vocabulary of morphemes by linguistic experts. In the past, research in computational linguistics mainly focused on unsupervised morphological analysis for large datasets with approaches like Linguistica [9] and Morfessor [4]. Recently, Shalonova et al. [14] have introduced an efficient semi-supervised approach for small data sets. Afterwards, Spiegler et al. compared the approach with Morfessor [15]. In this paper, we extend the semi-supervised approach for large data sets and make it unsupervised through a pre-process based on a window. We make the assumption that each word for any language has prefixes, a stem and suffixes. We do not restrict the number of prefixes and suffixes for our algorithm. We decide to label morphemes by their segment and not their grammatical categories. The method that we propose can be broken down into three phases and be processed to any unlabeled corpora. The first step consists of identifying the stem in each word, we propose the utilisation of a window with Minimum Description Length (MDL) to cover the problem. In second step for finding sequences of prefixes and suffixes, we apply an unsupervised algorithm which has been simplified for large datasets. Finally, in last step we aggregate the results from the first and the second step.

2 Stem extraction using a window of letters

Extraction of morpheme sequences are a hard task in languages where the word form includes sequences of prefixes, stem and sequences of suffixes. However, by finding the boundaries of the stem first, it is possible to extract prefix sequences and suffix sequences efficiently. In order to extract the stems, we look for a pseudo stem in the middle of each word, which is most often a position overlapping the real stem. We develop a heuristic to seek the most probable stem given a word through a window by using the MDL principle. We define a window by two boundaries within a word between letters. In other words, a window is a sub string of a word. During initialisation, the width of the window is defined as a letter in the middle of a word. Thereafter, an algorithm is used to shift or increase the width of the window from its initial point to its left and/or to its right side. Consequently, an evaluation function is used for each window and repeated for the best windows up to no better windows are found. The final boundaries are considered as the limit of the stem in the word. We iterate the algorithm for each word in the corpus, the evaluation function used is the *MDL window score*.

Definition 1. *Let* $win = (l_{win}, u_{win})$ *be a window with a lower boundary* l_{win} *and an upper boundary* u_{win}*. Given a word w and a window win the* MDL window score *is defined by:*

$$MDL\_Window\_Score(win, w) = log_2(u_{win} - l_{win} + 1) + log_2(np\_sub\_string(w, l_{win}, u_{win})) \qquad (1)$$

where np_sub_string denotes the n-gram probability of the window win in the word w.

As soon as the algorithm has been applied to each word from a corpus, we process the left side of each window to extract the prefixes. In a similar way, we process the right side of the window to extract the suffixes. To do so, we use an extension of the GBUSS algorithm presented in the next section.

3 Graph-Based Unsupervised Morpheme Segmentation

The algorithm GBUSS (Graph-Based Unsupervised Suffix Segmentation) was developed in [10] to extract suffix sequences efficiently and applied for Russian and Turkish languages on a training set in [10, 14]. Afterwards, we use GBUSS to extract independently both prefix and suffix sequences, instead of only suffix sequences. We refer to prefixes and suffixes generally as morpheme. We call L-corpus (R-corpus) the list of words obtained from the left-side (right-side) of the windows. In an independent manner, we use the term M-corpus for a L-corpus (R-corpus) in a prefix (suffix) graph-based context. We call GBUMS (Graph-Based Unsupervised Morpheme Segmentation) the extended version of the GBUSS algorithm for morpheme extraction. Moreover, we present one improvement from GBUSS, a simplified evaluation function for merging nodes which replace FPPEdgeL (An evaluation measure for context score and suffix frequency adapted from Yoshida et al. [16]). In this section, we describe GBUMS to extract sequence of morphemes.

3.1 Morpheme graph for agglomerative morpheme extraction

GBUMS uses a morpheme graph in a bottom-up fashion. Similar to Harris [11], we base our algorithm on letter frequencies. However, where Harris builds on successor and predecessor frequencies, we use position-independent n-gram statistics to merge single letters to morphemes until a stopping criterion has been met. In the morpheme graph, each node represents a morpheme and each directed edge the concatenation of two morphemes labeled with the frequencies in a M-corpus.

Definition 2. *Let $M = \{m_i | 1 \leq i \leq n\}$ be a set of morphemes, let f_i be the frequency with which morpheme m_i occurs in a M-corpus X 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 $V = \{v_i | 1 \leq i \leq k\}$ 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 X, then, step by step nodes are merged to create the real morphemes. To merge nodes an evaluation function is necessary. In [14], Shalonova et al. proposed one based on frequency and entropy. For large data sets, due to high computational costs we simplify the equation and do not take in account the entropy. Moreover, a pair (m_1, m_2) adequate for merging was characterised by the following constraints:

- low frequency for the individual morphemes m_1 and m_2
- high frequency of the concatenation $m_1 \cdot m_2$
- short sequences $m_1 \cdot m_2$

The approach that we present does not take in consideration the last constraint and can be viewed as the *lift* of a rule for association rules in data mining [3]. Consequently, we name the evaluation function *Morph_Lift*.

Definition 3. *$Morph\_Lift$ is defined for a pair of morphemes m_1 and m_2 as follows:*

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

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 over-generalisation. In other words, we need to stop the algorithm before getting the initial M-corpus (since no merging is possible). Four stopping criteria have been presented in [14] to treat this issue. Only the most efficient has been utilised in this article. The criterion comes from [12]. This criterion is based on the Bayesian information Criterion (BIC) and Jensen-Shannon divergence. The latter is defined as follows [5]:

Definition 4. *The* Jensen-Shannon divergence *is defined for two morphemes m_1 and m_2 as the decrease in entropy between the concatenated and the individual morphemes:*

$$D_{JS}(m_1, m_2) = H(m_1 \cdot m_2) - \frac{L_{m_1} H(m_1) + L_{m_2} H(m_2)}{N} \tag{3}$$

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 SC1 *requires that $\Delta BIC < 0$ which translates to:*

$$\max_{s_1, s_2} D_{JS}(s_1, s_2) \leq 2 \log_2 N \tag{4}$$

We stress that the BIC is equal to MDL except that the BIC sign is opposite [8]. To sum up, the GBUMS (Algorithm 1) follows these steps:

1. build a morpheme graph from all one-letter morphemes.

Algorithm 1 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 ← DeleteBoundariesBetween(M-CorpusSeg, Label(Max_p), Label(Max_q));
    Graph ← AdjustGraph(M-corpusSeg, Graph);
  until StoppingCriterion(pMax, qMax, Max)
```

2. search a pair that maximises $Morph\_Lift$.

3. merge the pair found by step 2.

4. test stopping criterion, if valid go back to step 2 else stop.

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 $Morph\_Lift$. Since, the stopping criterion is violated the segmented M-Corpus represents the sequence morphemes. In this section, we have presented a method to extract sequences of prefixes and sequences of suffixes. In the next section, we show the final step and the full algorithm to obtain a segmented corpus from raw words.

4 UNsupervised GRAph DEcomposition

In previous sections, we showed how to find prefix sequences, pseudo stem and suffix sequences without linking them. The final step of the segmentation algorithm is straightforward, it is made up of the aggregation (concatenation) of the sequence of prefixes, pseudo stem and sequence of suffixes found with the MDL Window and GBUMS for each word. The complete algorithm called UNGRADE (UNsupervised GRAph DEcomposition) including all phases is summarized in (Algorithm 2).

Algorithm 2 The UNGRADE algorithm

```
input  Corpus = List of Strings
output  SegmentedCorpus = List of Strings
  for all word ∈ Corpus do
    win ← (MiddlePosition(word), MiddlePosition(word));
    MDL_Window_Score(win, word);
  end for
  (L-Corpus, pseudo_stem, R-Corpus) ← SplitCorpusByWindow(Corpus);
  L-CorpusSeg ← GBUMS(L-Corpus);
  R-CorpusSeg ← GBUMS(R-Corpus);
  SegmentedCorpus ← Aggregate(L-CorpusSeg, pseudo_stem, R-CorpusSeg);
```

Note that the execution of GBUMS for L-corpus and R-corpus can be processed in parallel to reduce significantly the running time.

5 Related work

Much research on *Unsupervised segmentation of morphology* has focused on statistical approaches according to the work of Harris [11, 6] and tuning of parameters according to a language like Gaussier [7]. Brent (1993) presented the MDL theory for Computational linguistic problems with a probabilistic approach using the spelling of words [1]. Afterwards, Brent (1995) defined an approach for finding suffixes in a language [2]. Unfortunately, Brent's approach required a special tagging of the data. Subsequently, Goldsmith (2001) used MDL [9] to combine the results of multiple heuristics based on statistic like [6, 7] in a software called *Linguistica*. Goldsmith defined a model for MDL with *signature*. However, Linguistica was only focused on European languages. More recently, Creutz et al. [4] (2002) presented *Morfessor* with two new approaches to discover morphemes named *Morfessor baseline* and *Morfessor Categories-MAP*. The former method was based on MDL in a recursive method. The latter one, the most efficient, combined Maximum Likelihood and Viterbi for an optimal segmentation. Morfessor was developed independently of languages and provided good results. Lately, Paramor (2007) developed by Monson, in a similar way to Goldsmith with *signatures* used *paradigm* without using MDL. Paramor worked in two steps and provided results as good as Morfessor. In 2008, Results from Paramor and Morfessor were combined and provided better results than one of them alone [13].

6 Processing and results

In order to test UNGRADE, we used the Morpho Challenge data sets of 2009 which contains English (384903 words), German (1266159 words), Finnish (2206719 words), Turkish (617298 words), Arabic non-vowelized (14957 words) and Arabic vowelized (19244 words). Before running UNGRADE on the different data sets, we decided to use a pre-processing algorithm to remove marginal words and potential noise. To do so, we analysed the word length distribution to remove infrequent short and long words. We came up to the following length range for each data set (Table 1). Therefore, we used smaller data sets as input to UNGRADE. After running UNGRADE, in order to segment the remaining words of the original data sets, we used a segmented corpus from the output of UNGRADE as a model of segmentation. Thus, we applied this model to each word not present in the input data set of UNGRADE.
The evaluation measure used is the *F-measure* which is the harmonic mean between the Precision and the Recall.

$$F-measure = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{5}$$

The precision is computed as follows: A sample of words is randomly chosen from the result file. Then, a pair of words is defined for each morpheme of each word in the sample randomly by a word sharing the same morpheme from the result file. Finally, each pair of words is compared to the same pair in the gold standard to check if they have the same morphemes in common.
Similarly, the recall is calculated except that the sampling and the pairs are defined using the gold standard. Also, the checking of morphemes is carried out on the result file and not on the gold standard.
We stress that the morphemes labels that we use are the spelling of the morphemes and not their grammatical categories. The final results used a linguistic morpheme analysis gold standard and are showed on (Figure 1 and Table 2).

The F-measure for German, English, Turkish and Finnish are of the same order of magnitude (between 33.44% and 37.11%). Surprisingly, Arabic non-vowelized provided the worst (26.78%) and Arabic vowelized the best (54.36%) F-measure among all languages. The divergence in F-measure for Arabic is explained by the average word length. For Arabic, the non-vowelized words are almost on average twice shorter for the same amount of morphemes on average. We can note that the precision is higher than recall for all data sets except English. The low level of precision in English is due to the low morpheme number distribution on average. This observation is confirmed in Arabic (vowelized) where the morpheme number distribution on average is higher and therefore gives a high precision. It is interesting to remark that even if the starting point to look for a pseudo stem of the UNGRADE algorithm is less correct for Turkish (Turkish does not have prefixes), the results are quite competitive with Finnish. To sum up, UNGRADE is more efficient for a language with long words on average and a high number of morphemes by word on average.

Table 1: Pre-processing of data sets

Language	Minimum length	Maximum length
English	3	13
German	5	18
Finnish	5	25
Turkish	5	16
Arabic (non-vowelized)	2	9
Arabic (vowelized)	3	18

Table 2: Results from the Morpho Challenge 2009

Language	Precision	Recall	F-Measure
English	.2829	.5389	.3711
German	.3902	.2925	.3344
Finnish	.4078	.3305	.3651
Turkish	.4667	.3016	.3664
Arabic (non-vowelized)	.8348	.1595	.2678
Arabic (vowelized)	.7215	.4361	.5436

7 Conclusion and future directions

An unsupervised three-step algorithm for word segmentation UNGRADE has been presented. UNGRADE is defined on the general assumption on the structure of words for languages that each word contains prefixes, a stem and suffixes. UNGRADE uses a basic labeling of the morphemes which is by their segment. Results on the Morpho Challenge 2009 for 5 languages (English, German, Turkish, Finnish, Arabic vowelized and Arabic non-vowelized) show encouraging payoff for this simple algorithm on all languages. Results point out that UNGRADE performs on average with 35.92% of F-measure for English, German, Turkish and Finnish. Particularly, outcomes demonstrate that UNGRADE work well on Arabic vowelized with more than 54% of F-measure. UNGRADE illustrate high performance for languages with long words including a high number of morphemes. An important future work should use different starting points for looking to the pseudo stem at for instance the beginning of the word, the end of the word, etc. Afterwards, a committee could choose the best segmentation under some MDL criterion which may improve significantly the results.

Acknowledgments

We would like to thank our team colleagues Roger Tucker and Ksenia Shalonova for consultations on general morphological analysis. The work was sponsored by EPSRC grant EP/E010857/1 *Learning the morphology of complex synthetic languages*.

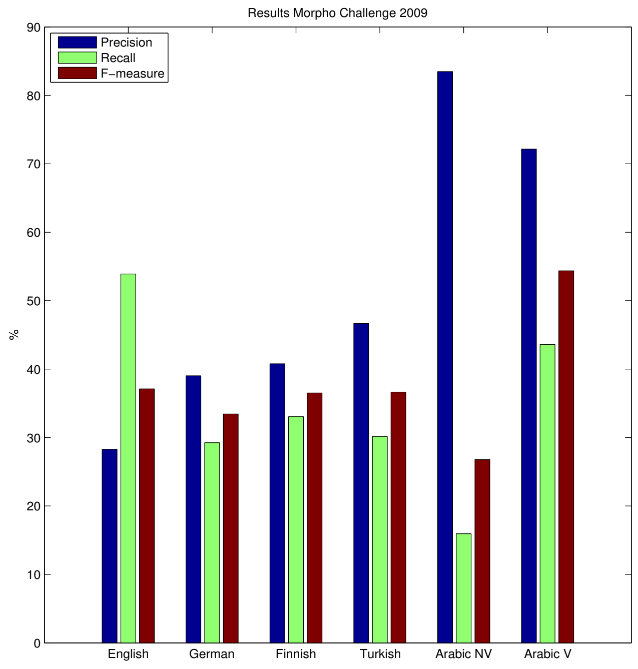

Figure 1: Results from the Morpho Challenge 2009

References

[1] M. R. Brent. Minimal generative models: A middle ground between neurons and triggers. In *15th Annual Conference of the Cognitive Science Society*, pages 28–36, 1993.

[2] Michael R. Brent, Sreerama K. Murthy, and Andrew Lundberg. Discovering Morphemic Suffixes A Case Study In MDL Induction. In *In Fifth International Workshop on AI and Statistics,Ft*, pages 264–271, 1995.

[3] S. Brin, R. Motwani, J. D.Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. In *SIGMOD '97: Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, pages 255–264, New York, NY, USA, 1997. ACM.

[4] M. Creutz and K. Lagus. Unsupervised discovery of morphemes. In *Proceedings of the ACL-02 workshop on Morphological and phonological learning*, pages 21–30, Morristown, NJ, USA, 2002. Association for Computational Linguistics.

[5] I. Dagan, L. Lee, and F. Pereira. Similarity-Based Methods for Word Sense Disambiguation. *Proceedings of the Thirty-Fifth Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, pages 56–63, 1997.

[6] H. Déjean. Morphemes as Necessary Concept for Structures Discovery from Untagged Corpora, 1998.

[7] E. Gaussier. Unsupervised Learning of Derivational Morphology From Inflectional Lexicons. In *ACL '99 Workshop Proceedings: Unsupervised Learning in Natural Language Processing*, 1999.

[8] Y. Geng and W. Wu. A Bayesian Information Criterion Based Approach for Model Complexity Selection in Speaker Identification. In *Advanced Language Processing and Web Information Technology, 2008. ALPIT '08. International Conference on*, pages 264–268, July 2008.

[9] J. Goldsmith. Unsupervised Learning of the Morphology of a Natural Language. *Computational Linguistics*, 27:153–198, 2001.

[10] B. Golénia. Learning rules in morphology of complex synthetic languages. Master's thesis, University of Paris V, 2008.

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

[12] W. Li. New Stopping Criteria for Segmenting DNA Sequences. *Physical Review Letters*, 86(25):5815–5818, 2001.

[13] C. Monson, J. Carbonell, A. Lavie, and L. Levin. *ParaMor: Finding Paradigms across Morphology*. Springer Berlin / Heidelberg, 2008.

[14] K. Shalonova, B. Golénia, and P.Flach. Towards Learning Morphology for Under-Resourced Fusional and Agglutinating Languages. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(5):956–965, July 2009.

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

[16] M. Yoshida and H. Nakagawa. Automatic Term Extraction Based on Perplexity of Compound Words. *Natural Language Processing IJCNLP 2005*, pages 269–279, 2005.