Morphological Analysis of the Slovak Language

Daniel HLADEK, Jan STAS, Jozef JUHAR

Department of Electronics and Multimedia Communications, Faculty of Electrical Engineering and Informatics, Technical University of Kosice, Park Komenskeho 13, 04200 Kosice, Slovak Republic

daniel.hladek@tuke.sk, jan.stas@tuke.sk, jozef.juhar@tuke.sk

DOI: 10.15598/aeee.v13i4.1491

Abstract. *This paper proposes a new statistic-based method of segmenting words by identification of a suffix. Ability to identify suffix can improve morphological analysis by allowing the classifier to assign tags to words previously unseen in the training corpus. Identified suffix of the word can be used to improve the accuracy of the part-of-speech tagging or other natural language processing tasks.*

Keywords

Hidden Markov model, morphological analysis, part-of-speech tagging.

1. Introduction

The Slovak language has complex rules for word inflection. As there are many possible word forms, classification of contexts for tasks such as Part-Of-Speech (POS) tagging, lemmatization or semantic analysis gets harder because of the larger search space and more difficult training of model of the classifier.

The most important contribution to the automated morphological analysis of the Slovak language was a proposal of the morphologically annotated corpus of the by Slovak National Corpus, although public availability of the corpus is limited. The corpus defines a set of POS tags and annotation guidelines for the Slovak language and is still the most used set of morphological tags [2].

There is only a small number of POS tagging systems adapted for the Slovak language. Most of the used approaches use some form of a hidden Markov model. Averaged perceptron classifier MORCE [3] proposes a similar approach to Maximum Entropy and can utilize a large number of context features. A web interface to a tool proposing multiple possible morphological annotations for given text is available at this address (http://morpholyzer.fiit.stuba.sk:8080/PosTagger/ index.jsp).

Hidden Markov Model (HMM) utilizing ID3 regression tree TREE TAGGER [10] has been trained and used for tagging Slovak part of the Aranea effort [1].

HMM-based DAGGER [4] has been used for the preparation of language model for speech recognition [8] and several other corpora.

Each of these classification systems uses some form of word analysis for improvement of classification accuracy and efficient constraining of the search space of possible tags.

One of the possible approaches to identify morphological suffix of the word is to describe each inflection rule and include it in the system. The advantage of this rule-based approach is that result for covered cases will be very precise, but the unknown input can not be recognized. A classical example is Porter stemmer [7] which describes special formal grammar language to describe suffix stripping rules.

The other possible approach is to examine lexicon and look for repeating patterns, analyze it statistically and extract rules that might be relevant. This approach is more general and able to describe any given lexicon, in a way independent of the lexicon contents or language rules. [6], [9] propose statistical stem identification, [11] is method for statistical suffix identification.

This paper presents a statistical way to analyze a given lexicon and extract list of common suffixes and list of common rules for suffix identification. A suffix identified by the algorithm should represent a morphological form of a word in a feasible way. The experimental section will show that the presented approach efficiently improves methods for context classification.

2. Statistical Suffix Identification Algorithm

FThe first step of the suffix extraction algorithm is to construct a sorted lexicon of reversed words. Words with similar suffixes will be together in this lexicon, and the following stem should identify groups of words with similar features that can be together in the same group with the same identified suffix. Sorting of the lexicon eases searching for words with long suffices.

The group consists of words with the same suffix. Each possible group of words from the lexicon can be evaluated by a "fitness" that describes how well the words match together. If fitness of a group of words can be calculated then it is possible to search for a division of a lexicon that will be evaluated the best-find a lexicon division that will yield the best sum of group fitness.

The fitness $F(s)$ for $i-th$ word $w(i)$ in vocabulary V with suffix s is calculated as:

$$F(s) = \sum_{s \in w(i), w(i) \in V}^{i} 2L(s) - L(w(i)), \qquad (1)$$

where $L(s)$ is length of a suffix s and $L(w(i))$ is length of $i-th$ word in vocabulary V. In other words, evaluation of the suffix s is two times number of character of the suffix minus number of letters for all words having the suffix.

The same can be written as equation:

$$F(s) = 2N_s L(s) - \sum_{s \in w(i), w(i) \in V}^{i} L(w(i)), \qquad (2)$$

where N_s is number of words in vocabulary V having suffix s. The second term of the equation expresses a number of letters of all words with suffix s in vocabulary.

This equation says that group of short words with the longest possible suffix is evaluated as the best.

The next step of the division of lexicon into groups is to find a candidate suffix to be evaluated. As it is expressed in Eq. (2), it is supposed that a long word contains the longest suffix and group of words with this suffix will have the best fitness. All possible suffixes from a long word are taken, and their respective groups are evaluated. Suffix with the best fitness F_V is the result. A long word is a word whose neighbors in the reversed sorted lexicon are shorter.

The last step of the algorithm is to remove all words in vocabulary with the identified suffix and while the vocabulary is not empty, continue searching for the next long word and evaluating its candidate suffices.

To conclude, the suffix identification algorithm can be shortly described as:

- Compose a reversed sorted vocabulary V.
- Take a candidate long word w_l, as the first word whose neighbours in the vocabulary are shorter.
- Evaluate all possible suffices s_i from a candidate long word w_l with fitness $F(s_i)$.
- Take the best suffix s_i and its corresponding word group out of lexicon and add it to the result set.
- If the vocabulary V is not empty, take the next long word and continue algorithm with evaluation of possible suffices.
- If the vocabulary is empty, end.

The result of the algorithm is set of suffixes with corresponding words.

The resulting rule base can be used for morphological analysis of any given text in the correct language. If a rule for splitting a word is included in the rule-base, it can be used. If a word is unknown to the rule-base, a suffix subtraction algorithm can be used to choose the best matching suffix for a word.

The input of the algorithm is a set of possible suffices and a word to be split.

Suffix subtraction algorithm:

- If a word is too short, end without suffix identification.
- Strip three characters from the left side of the word. It is supposed that these characters are certainly not part of the suffix.
- If the remaining part is too short to be a suffix, end without suffix identification.
- Search the remaining part of the word in the suffix set.
- If the suffix is found, end.
- If the suffix is not found, cut one more character from the left and search for the shorter suffix (continue with step 3).

3. Morphological Analysis Using HMM Model and Suffix Identification

One of the possible uses of the proposed suffix identification method is part-of-speech tagging. The following section will describe how to compose a basic hidden Markov model and how to use it for context classification. Most of the current state-of-the-art morphological analyzers is based on this theory. The proposed algorithm for POS tagging is based on HMM as well.

3.1. The Baseline HMM Model Components

The model is calculated from the training corpus. The corpus is analyzed, and counts of interesting events are calculated. These counts are converted to probabilities using some modification of Maximum Likelihood method. After the model and its parameters is estimated, Viterbi algorithm is used to find the best matching sequence of tags to the presented sequence of words.

The baseline HMM model consists of these components:

- **State set** is set of all possible states that have been seen during the training phase. In the case of POS tagging, one state corresponds to one POS tag.
- **State-transition model** expresses the probability of a state according to the occurrence of the previous state. Our implementation algorithm can take two preceding states into the account (second-order HMM). Modeling of more than two preceding states is not practical because computational complexity rises exponentially with a number of states, and higher order state-transition model is harder to smooth (estimate the probability of unseen n-grams).
- **Observation set** In the case of the baseline POS tagging, one observation corresponds to one word. It is easy to find all possible states for a word seen in the training set, but for unseen words all possible states must be examined. If there is no additional heuristics, it is hard to make a classification for words unseen in the training phase. This is a big issue in the case of highly inflectional language, where one basic form of a word can have many inflections.
- **State-observation probability model** expresses the probability of a state according to presented observation. Again, it is hard to estimate accurately state probabilities only according to word-tag pairs in the training corpus.

If some unknown observation or state sequence is presented to the model and it has not been seen in the training corpus, the baseline maximum likelihood method gives zero probability. To assign a non-zero probability to unseen events ,a smoothing method is required. A smoothing method usually artificially manipulates empirical event counts to move some part of probability mass to the events unseen in the training phase but expected during the testing phase.

3.2. The Viterbi Algorithm

The naive approach to find the best sequence of states for the given observation sequence is to inspect each possible state combination, compute its probability given to the model and choose a sequence with the highest probability. It is clear that this approach would work only on very constrained search space that is unrealistic. The Viterbi algorithm belongs to a group of "dynamic programming" algorithms, and it allows to perform a search in state-space in quadratic time.

The first step of the Viterbi algorithm is to construct a Viterbi trellis. The trellis consists of state-observation pairs as nodes and possible state-transitions as arcs. Each arc in the trellis has a Viterbi value assigned. The trellis is constructed recursively. After the trellis is constructed, it is possible to find the best sequence of transitions using evaluation of Viterbi trellis arcs, where the search starts from the rightmost part of the trellis and goes back to the front.

States k in the Viterbi trellis for the first presented observation are evaluated using state-observation model P_o as:

$$V_1(k) = P_o(w_1|k). \tag{3}$$

Recursively, states k for each next $t-th$ presented observation w_t are evaluated using transition probability model P_t, state-observation model P_o and previous Viterbi values $V_{t-1}(k)$ for each possible past state x according to the equation:

$$V_t(k) = P_o(w_t|k) \max_{x \in S} P_t(k|x) V_{t-1}(k). \tag{4}$$

Searching of the maximum previous Viterbi value also means the best previous state for the given present state. After all states are evaluated, and the best last state is known for each state, it is possible to find the best path, starting from the very last and very best state.

3.3. Additional State-Observation Model Smoothing

HMM classification has been many times proven useful for POS tagging. If a number of word forms of a language is low and so is a number of morphological tags, the baseline HMM model can be pretty effective. On the other hand, if a training corpus is small and number of possible word forms and morphological classes is high, state-transition model and state-observation model becomes sparse and as a consequence the Viterbi search assigns zero value to a perfectly possible state-observations.

The accuracy of the HMM-based classifier depends on the quality of training data and smoothing techniques for adjusting state-transition matrix and observation probability matrix. The Slovak language is morphologically rich and as a consequence the state set is large. It will be explained how the identified suffix can be used to improve the basic algorithm for context classification.

Smoothing is a way moving a part of a probability mass to those events that have not been observed in the training database, but it is likely that they will occur in the process of classification. Training data never contain all possible event, and the classifier must be able to generalize seen the event and be able to deal with unseen events.

The proposed algorithm attempts to solve this issue by better smoothing of the state-observation model by using additional state-observation model that takes morphological features of the language into the account. The final state-observation probability is calculated as a linear combination of the two models.

The proposed improvement takes special issues if the inflective languages into the account and utilizes suffix of a word as a feature for additional smoothing of the observation probability matrix. State transition matrix is smoothed using classical Knesser-Ney technique, known from the language modeling.

The proposed Dagger algorithm adjusts the final state-observation probability model P_o to take suffix s of a word w into the account, as it is expressed in the equation:

$$P_o(k|w) = (1-\lambda)P_w(w|k) + \lambda P_s(k|s), \qquad (5)$$

constant λ expresses the weight of the additional smoothing model and is manually set to some small value, e.g. 0.001. P_w is the baseline state-observation model calculated from word and state k occurrences, P_s is additional state-observation probability model, calculated from occurrences of word suffices and states in the training set.

This smoothing technique allows to distinguish between states of previously unseen words by observing its suffix.

The second major improvement is the constraining of the search space by observing which states occurred together with which word. A special morphological lexicon is constructed during training, and each observed word contains a list ob occurring states. The second part of the morphological lexicon takes suffices into the account. A list of states occurring with the word suffix is recorded as well.

When constructing set of possible states for the given words, the following back-off scheme is used:

- If presented word is in the lexicon, its state set is used.
- If the word is not present, its suffix is found using suffix splitting rules or suffix subtraction algorithm as it was presented above.
- If the word suffix is found, states in the suffix part of the morphological lexicon are used.
- If the word is too short or too long for suffix extraction, the unknown tag is proposed.

If a morphological lexicon is used, only valid word-tag combinations have to be searched. This feature increases both classification speed and accuracy.

The main goal of the experiments is to prove the usefulness of the proposed suffix identification method for the task of POS tagging of the Slovak text. Our hypothesis is that the suffix identification should improve smoothing of the state-observation model and can effectively constrain search space.

The first experiment evaluates the classifier on a pre-tagged Slovak part of the web2corpus project [5]. In the first step, the corpus is divided into training and testing part, where each tenth sentence goes to the testing set and the rest goes to training. A short summary of the evaluation corpus is in the Tab. 1.

Tab. 1: Evaluation corpus characteristics.

	Sentences	**Tokens**
Testing set	111 577	3 064 178
Trainnig set	1 004 188	27 607 502

The training part is used to create automatically a set of rules for suffix identification as it has been described above. This rule-base is used as supporting information for training the HMM model. The trained model is evaluated on the testing set by calculation of classification error rate (fraction of the bad classification results over all results when comparing to the reference tagging).

In the second experiment, a comparison with another HMM-based algorithm with suffix identification is performed. The TreeTagger algorithm has been selected, and its model has been trained on the same training data. The training data were prepared according to the instructions - end of sentences were marked and open class-lexicon has been constructed from the training corpus. Numeral tags were excluded from the lexicon as it was advised by the author. The tagging error rate has been calculated using the same methodology as with the first experiment.

Results are summarized in Tab. 2. It can be seen that classification accuracy of the classifier utilizing our proposed suffix identification method is comparable to the suffix processing of the TreeTagger.

Tab. 2: Evaluation of experiments.

	Correct tokens	**Incorrect tokens**	**Err**
Dagger	2 926 649	137 529	0.04488
Tree Tagger	2 924 760	139 418	0.04549

4. Conclusion

The presented experiments show that the proposed word splitting method provides useful information about word morphological form. The method is language-neutral, so it might be useful to use the method for morphological analysis of other languages where most of the morphological features are hidden in the suffix of the word.

This word segmentation method can also be useful for many other natural language processing tasks such as information retrieval, automatic correction or semantic parsing.

The presented algorithm can also be inverted and used for stem identification. After some more generalization is designed, the principle of the suffix identification can be extended for identification of even smaller parts of words than suffix.

Acknowledgment

The research presented in this paper was supported by the Ministry of Education, Science, Research and Sport of the Slovak Republic under the project VEGA 1/0075/15 (66 %) and the Research & Development Operational Programme funded by the ERDF under the project implementation University Science Park TECHNICOM for Innovation Applications Supported by Knowledge Technology, ITMS: 26220220182 (34 %).

References

[1] BENKO, V. Yet Another Family of (Comparable) Web Corpora. In: *17th International Conference Text, Speech and Dialogue.* Brno: Springer, 2014, pp. 247–256. ISBN 978-3-319-10816-2. DOI: 10.1007/978-3-319-10816-2_31.

[2] GARABIK, R. and M. SIMONIKOVA. Slovak Morphosyntactic Tagset. *Journal of Language Modelling.* 2012, vol. 0, no. 1, pp. 41–63. ISSN 2299-8470. DOI: 10.15398/jlm.v0i1.35.

[3] SPOUSTOVA, D., J. HAJIC, J. VOTRUBEC, P. KRBEC and P. KVETON. The Best of Two Worlds: Cooperation of Statistical and Rule-Based Taggers for Czech. In: *45th Annual Meeting of the Association for Computational Linguistics.* Prague: Association for Computational Linguistics, 2007, pp. 67–74. ISBN 978-0-7695-3982-9. DOI: 10.1109/ICCEA.2010.86.

[4] HLADEK, D., J. STRAS and J. JUHAR. Dagger: The Slovak morphological classifier. In: *54th International Symposium ELMAR.* Zadar: IEEE, 2012, pp. 195–198. ISBN 978-1-4673-1243-1.

[5] MAJLIS, M. and Z. ZABOKRTSKY. Dagger: Language Richness of the Web. In: *8th International Conference on Language Resources and Evaluation.* Istambul: ELRA, 2012, pp. 23–25. ISBN 978-2-9517408-7-7.

[6] PAIK, J. H. and S. K. PARUI. A Fast Corpus-Based Stemme. *Transactions on Asian Language Information Processing.* 2011, vol. 10, iss. 2, no. 8, pp. 1–16. ISSN 1530-0226. DOI: 10.1145/1967293.1967295.

[7] PORTER, M. F. An algorithm for suffix stripping. *Program.* 2006, vol. 40, iss. 3, pp. 211–218. ISSN 0033-0337. DOI: 10.1108/00330330610681286.

[8] RUSKO, M., J. JUHAR, M. TRNKA, J. STRAS, S. DARJAA, D. HLADEK, R. SABO, M. PLEVA, M. RITOMSKY and M. LOJKA. Slovak Automatic Dictation System for Judicial Domain. In: *5th Language and Technology Conference.* Poznan: Springer, 2014, pp. 16–27. ISBN 978-3-319-08958-4. DOI: 10.1007/978-3-319-08958-4_2.

[9] SAHARIA, N., K. M. KONWAR, U. SHARMA and J. K. KALITA. An Improved Stemming Approach Using HMM for a Highly Inflectional Language. In: *14th International Conference Computational Linguistics and Intelligent Text Processing.* Samos: Springer, 2013, pp. 1720–1731. ISBN 978-3-642-37247-6. DOI: 10.1007/978-3-642-37247-6_14.

[10] SCHMID, H. TreeTagger - a language independent part-of-speech tagger. *Ludwig Maximilians Universitat Munchen* [online]. 1995. Available at: http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/.

[11] SNAJDER, J., B. D. BASIC and M. TADIC. Automatic acquisition of inflectional lexica

for morphological normalisation. *Information Processing & Management.* 2008, vol. 44, iss. 5, pp. 1720–1731. ISSN 0306-4573. DOI: 10.1016/j.ipm.2008.03.006.

About Authors

Daniel HLADEK was born in Kosice, Slovakia in 1982. He received his M.Sc. from Artificial Intelligence in 2007 and Ph.D. in 2009. He is currently working as an assistant professor at the Department of Electronics and Multimedia Communications, Faculty of Electrical Engineering and Informatics, Technical University of Kosice with focus on natural language processing, speech and audio processing and intelligent decision methods. He is author and co-author of more than 30 conference and journal papers from this area.

Jozef JUHAR was born in Poproc, Slovakia in 1956. He received his M.Sc. in Radioelectronics from in 1980 and Ph.D. in 1991. He is currently working as a Full Professor and Head of Department at the Department of Electronics and Multimedia Communications, Faculty of Electrical Engineering and Informatics, Technical University of Kosice. He is author and co-author of more than 220 scientific papers. His research interests include digital speech and audio processing, speech synthesis and development in spoken dialogue and speech recognition systems.