INFORMATICA, 2004, Vol. 15, No. 2, 231–242

Designing HMM-Based Part-of-Speech Tagger for Lithuanian Language

Giedrė PAJARSKAITĖ, Vilma GRICIŪTĖ, Gailius RAŠKINIS
Center of Computational Linguistics, Vytautas Magnus University
Donelaičio 52, 3000 Kaunas, Lithuania
e-mail: pajgie@lycos.com, gvilma@lycos.com, idgara@vdu.lt

Jan KUPER
Faculty of Computer Science, University of Twente
P.O.Box 217 7500 AE Enschede, the Netherlands
e-mail: jankuper@cs.utwente.nl

Received: June 2002

Abstract. This paper describes a preliminary experiment in designing a Hidden Markov Model (HMM)-based part-of-speech tagger for the Lithuanian language. Part-of-speech tagging is the problem of assigning to each word of a text the proper tag in its context of appearance. It is accomplished in two basic steps: morphological analysis and disambiguation. In this paper, we focus on the problem of disambiguation, i.e., on the problem of choosing the correct tag for each word in the context of a set of possible tags. We constructed a stochastic disambiguation algorithm, based on supervised learning techniques, to learn hidden Markov model's parameters from hand-annotated corpora. The Viterbi algorithm is used to assign the most probable tag to each word in the text.

Key words: part of speech tagging, morphological disambiguation, HMM modeling, smoothing, hand-annotated corpus.

1. Introduction

Part-of-speech (POS) tagging is the problem of assigning to each word of a text the proper tag in its context of appearance. Solving this problem is a prerequisite for a number of Natural Language Processing (NLP) applications such as natural language generation, machine translation, information extraction and retrieval using natural language, text to speech synthesis, automatic written text recognition, grammar checking and others. Part-of-speech tagging is accomplished in two basic steps: morphological analysis and disambiguation. An isolated word is considered to be morphologically ambiguous if it can be assigned to more than one morphological category. This paper focuses on the problem of disambiguation, i.e., on the problem of choosing the correct tag for each word in context from the set of possible tags.

Our analyses showed that about 46% of Lithuanian text tokens (words) are ambiguous. This is a high percentage in comparison with English (27%) and French (40%)

(Armstrong *et al.*, 1995). The most common types of morphological ambiguities are the following:

- Homoforms, i.e., part of speech ambiguities (for example, the word *metu* can go any of three parts of speech: 1) verb *mesti(-ta,-tė)* meaning *to throw* 2) adjective *metus* meaning *throwable* 3) noun *metas* meaning *time*).
- Homographs, i.e., ambiguities related to the fact that letters just partially encode pronunciation of words. (for example, if the word *seniai* is pronounced by stressing the first syllable, it is a noun meaning *old people*; if the last syllable of *seniai* is stressed, it is an adverb meaning *long ago*).
- Gender ambiguities (for example, the word form *kauniečių* is a plural genitive case of both *kaunietis* and *kaunietė* which are masculine and feminine *inhabitants of Kaunas* respectively).
- Case ambiguities (for example, the word *mama* can represent any of three cases: nominative, instrumental and vocative, of the same noun *mama* (*mother*)).

More than half of all morphological ambiguities found in Lithuanian are homoforms. Thus, our research was focused on the POS disambiguation of this kind.

2. Related Work

The disambiguation problem can be approached by rule-based, stochastic, neural and mixed-type techniques depending on the type of knowledge available to guide the disambiguation process. The rule-based techniques are guided by linguistic language models (LM). Linguistic LM range from a few hundreds to several thousands of rules written by linguists (Oostdijk, 1991).

As construction of such models is human labor-intensive, stochastic techniques gained popularity. Stochastic methods build a statistical LM from a training corpus and use it to disambiguate word sequences. For example, the CLAWS system (Garside *et al.*, 1987), learns and uses bigram information. The Multext tagger (Armstrong *et al.*, 1995) and some other taggers try to reduce the amount of training data needed to estimate the language model, and use the Baum-Welch re-estimation algorithm to iteratively refine an initial model obtained from a small hand-tagged corpus. More sophisticated taggers, such as Brill's tagger (Brill, 1995), automatically learn a set of transformation rules that correct errors committed by the most-frequent-tag tagging rule. Stochastic techniques were used in our Lithuanian POS tagger as well.

The accuracy of stochastic taggers and taggers based on linguistic Constraint Grammars is reported to achieve 96–97% and 99% of correct tag assignments respectively.

3. Stochastic Part-of-Speech Tagger

Stochastic disambiguation consists of building a statistical language model and using this model to disambiguate a word sequence. Consequently, our disambiguation, using a

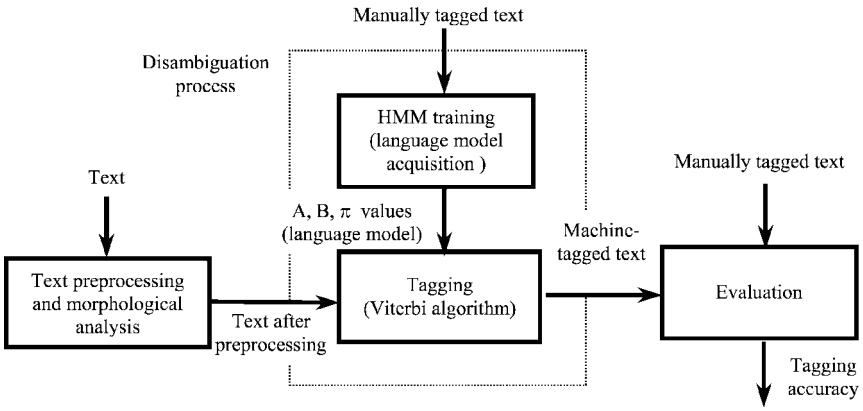

Fig. 1. The basic steps of part-of-speech tagging. The text on the left is analysed automatically. On the right, the disambiguation results of our program are compared with the same text tagged by humans.

Hidden Markov Model (HMM), was accomplished in two basic steps referred to as the training and tagging phase (Fig. 1).

In the training phase, HMM parameters were calculated for the probabilities of the occurrence of each tag-tag pair, or tag-word pair in a given context. Probabilities were stored in matrices and constituted the language model. In the tagging step, the language model was applied to select the most probable tag from the proposed set of candidate tags for each word in the text.

3.1. *Text Preprocessing and Morphological Analysis*

A text preprocessing step is required for the identification of words and of other relevant tokens in the text as well as for the mark-up of sentence boundaries. Morphological analysis must assign a set of potential annotations to each token or word.

Our mark-up of sentence boundaries was based on the assumption that the beginning of a new sentence was indicated by a capital letter preceded by a full stop, exclamation mark or question mark. Morphological analysis was performed by means of the tool "Lemuoklis" (Zinkevičius, 2000) that assigned all possible part-of-speech tags and other morphological information to each token in a text. Our tag set consisted of 18 part-of-speech tags (see Appendix A for their list). Fig. 2 shows an example of a text that resulted from text preprocessing and morphological analysis.

3.2. *HMM Training*

Let $w_1, w_2, \ldots, w_n$ be the representation of a complete sentence, where w_i indicates the ith word. The aim of morphological disambiguation is to find the sequence of lexical categories $t_1, t_2, \ldots, t_n$, where t_i is the tag corresponding to w_i, that maximizes the probability $P(t_1, t_2, \ldots, t_n | w_1, w_2, \ldots, w_n)$.

```
PR
Į             PL <į>
Mus IV        <aš>
kreipėsi      VM <kreiptis(-iasi,-ėsi)>
Jūsų          IV <tu>
ypatingi      BD <ypatingas>
ir            DL <ir> | JG <ir> |
kilnūs        BD <kilnus>
pasiuntiniai DK <pasiuntinys>
,             SZ
prašydami     VM <prašyti(-o,-ė)>
,             SZ
kad           DL <kad> | JG <kad> |
priimtume     VM <priimti(-ima,-ėmė)> | DK <priimtumas> |
Jus           IV <tu> | VM <justi(-nda,-do)> |
              VM <justi(-nta,-to)> |
PB
```

Fig. 2. Example of a text that results from text preprocessing and morphological analysis. "*PR*" and "*PB*" indicate beginning and ending of a sentence. One line of text describes one token: the token itself, and the list of possible part of speech tags separated by "|". Each list item contains a tag name written in capitals and a base form of the current token enclosed within "$<\ldots>$".

The definition of conditional probability entails that:

$$P(t_1,t_2,...,t_n|w_1,w_2,...,w_n) = \frac{P(t_1,t_2,...,t_n)\cdot P(w_1,w_2,...,w_n|t_1,t_2,...,t_n)}{P(w_1,w_2,...,w_n)}. \quad (1)$$

Since we are interested in finding the sequence $t_1, t_2, \ldots, t_n$ that gives the maximum value, the common denominator in all these cases will not affect the final result. Thus, the problem reduces to finding the sequence $t_1, t_2, \ldots, t_n$ that maximizes the expression

$$P(t_1, t_2, \ldots, t_n) \cdot P(w_1, w_2, \ldots, w_n|t_1, t_2, \ldots, t_n). \quad (2)$$

There are still no effective methods for calculating the probability (2) accurately, as it would require far too much data. But each of the two expressions in formula (2) can be approximated by probabilities that are simpler to collect by making some independence assumptions.

The probability of the sequence of categories $t_1, t_2, \ldots, t_n$

$$P(t_1, t_2, \ldots, t_n) = P(t_1) \cdot P(t_2|t_1) \cdot P(t_3|t_1, t_2) \cdot \ldots \cdot P(t_n|t_1, t_2, \ldots, t_{n-1}) \quad (3)$$

can be approximated by a series of probabilities based on a limited number of previous categories. We used bigram and trigram approximations given by formulas (4) and (5) respectively:

$$P(t_1, ..., t_n) \cong \prod_{i=1}^{n} P(t_i|t_{i-1}), \tag{4}$$

$$P(t_1, ..., t_n) \cong \prod_{i=1}^{n} P(t_i|t_{i-2}, t_{i-1}). \tag{5}$$

The bigram language model (4) approximates the probability $P(t_i|t_1, t_2, \ldots, t_{i-1})$ by the conditional probability of t_i given only one preceding category t_{i-1}, i.e., by $P(t_i|t_{i-1})$. The trigram language model (5) uses the conditional probability of one category given the two preceeding categories, that is, $P(t_i|t_{i-2}, t_{i-1})$. To account for the beginning of a sentence, we insert a pseudo-category $<s>$ at position 0 as the value of t_0.

The second probability $P(w_1, w_2, \ldots, w_n|t_1, t_2, \ldots, t_n)$ in formula (2) can be approximated by assuming that a word appears in a category independent of the words in the preceeding or succeeding categories. It is approximated by the product of the probabilities that each word occurs in the indicated part-of-speech:

$$P(w_1, w_2, ..., w_n|t_1, t_2, ..., t_n) \cong \prod_{i=1}^{n} P(w_i|t_i). \tag{6}$$

With these two approximations, the problem changes into finding the sequence $t_1, t_2, \ldots, t_n$ that maximizes the value of expression (7) or (8) for bigram or trigram LM respectively:

$$\prod_{i=1}^{n} P(t_i|t_{i-1}) \cdot P(w_i|t_i), \tag{7}$$

$$\prod_{i=1}^{n} P(t_i|t_{i-2}, t_{i-1}) \cdot P(w_i|t_i). \tag{8}$$

The language model can be realized using hidden Markov models. The hidden Markov model relies on three parameters, commonly referred to as the **A, B** and $\boldsymbol{\pi}$ matrices. The **A** matrix records the probabilities of the transitions between any two (bigram) or three (trigram) tags. The **B** matrix records the relation between the occurrence of a given tag and the set of words in which it occurs. The $\boldsymbol{\pi}$ matrix records the probability of a tag to occur in the initial state (i.e., at the beginning of a sentence).

A, B and $\boldsymbol{\pi}$ matrices can be estimated from a corpus of manually POS-tagged texts. Bigram probabilities, for instance, can be estimated simply by counting the number of times each pair of categories occurs compared to the individual category counts.

Let $W_1, W_2, \ldots, W_V$ be the set of different words (vocabulary) of a language where V is vocabulary size. Let $L_1, L_2, \ldots, L_M$ be the set of possible morphological categories

(tags) of a language where M is the number of different tags. Let the following frequencies be calculated from a given manually tagged text corpus:

$C(L_i)$ – frequency of tag L_i;
$C(L_j, L_i)$ – frequency of two consecutive tags L_jL_i;
$C(L_k, L_j, L_i)$ – frequency of three consecutive tags $L_kL_jL_i$;
$C(W_i, L_j)$ – number of times word W_i is assigned tag L_j;
$C(<s>)$ – number of sentences;
$C(<s>, L_i)$ – number of times a sentence begins with the tag L_i;
$C(<s>, L_j, L_i)$ – number of times a sentence begins with the pair of tags L_jL_i.

Then **A, B** and $\boldsymbol{\pi}$ matrices can be estimated as follows.

In the case of bigram LM:

$$A_{ij} = P(L_i|L_j) = \frac{C(L_j, L_i)}{C(L_j)}, \quad i, j = 1, \ldots, M, \tag{9}$$

$$B_{ij} = P(W_i|L_j) = \frac{C(W_i, L_j)}{C(L_j)}, \quad i = 1, \ldots, V,\ j = 1, \ldots, M, \tag{10}$$

$$\pi_i = P(L_i| <s>) = \frac{C(<s>, L_i)}{C(<s>)}, \quad i = 1, \ldots, M. \tag{11}$$

In the case of trigram LM:

$$A_{ijk} = P(L_i|L_j, L_k) = \frac{C(L_k, L_j, L_i)}{C(L_k, L_j)}, \quad i, j, k = 1, \ldots, M, \tag{12}$$

$$\pi_{0,i} = P(L_i| <s>) = \frac{C(<s>, L_i)}{C(<s>)}, \quad i = 1, \ldots, M, \tag{13}$$

$$\pi_{ij} = P(L_i| <s>, L_j) = \frac{C(<s>, L_j, L_i)}{C(<s>, L_j)}, \quad i, j = 1, \ldots, M. \tag{14}$$

A morphologically annotated corpus[1] consisting of 57100 word tokens was used in our experiments. The corpus was divided into two parts. HMM parameters were estimated on the training part of it that constituted 75% of the corpus. The training corpus had $V = 6331$ different word types, each being assigned to one or more out of $M = 18$ possible part-of-speech tags (see Appendix A). The remaining 25% part of the corpus was used for the evaluation of disambiguation accuracy.

3.3. *Smoothing*

Because any particular training corpus is finite, the bigram and trigram matrices are sparse. They have a very large number of zero probability bigrams (trigrams) that

[1]The corpus was compiled by the Center of Computational Linguistic at Vytautas Magnus University, Kaunas, Lithuania. It was composed of texts on history-related subjects. At the time we were finishing this article, the size of this morphologically annotated corpus reached 500 000 words, the corpus included carefully selected texts of different types.

should really have some non-zero probability. We used Add-One (Jurafsky, 2000) and Good-Turing (Gale, 1991) as smoothing techniques for re-evaluating some of the zero-probability and low-probability bigrams (trigrams), and assigning them non-zero value. These smoothing techniques were applied to **A** and $\boldsymbol{\pi}$ matrices.

Add-One smoothing consisted of taking the matrix of n-gram counts, before normalizing them into probabilities, and of adding one to all the counts. Smoothed bigram (trigram) probabilities $A^*_{ij}(A^*_{ijk})$ were computed as follows:

$$A^*_{ij} = P^*(L_i|L_j) = \frac{C(L_j, L_i) + 1}{C(L_j) + V}, \quad i, j = 1, \ldots, M, \tag{15}$$

$$A^*_{ijk} = P^*(L_i|L_j, L_k) = \frac{C(L_k, L_j, L_i) + 1}{C(L_k, L_j) + V}, \quad i, j, k = 1, \ldots, M. \tag{16}$$

Good-Turing smoothing re-estimated the amount of probability mass assigned to n-grams with zero or low count by looking at the number of n-grams with higher counts. Let N_c be the number of n-grams that occur c times, i.e., let N_0 be the number of n-grams that never occurred, N_1 the number of n-grams that occurred once, and so on. Smoothed bigram (trigram) probabilities $A^*_{ij}(A^*_{ijk})$ were computed as follows:

$$A^*_{ij} = P^*(L_i|L_j) = \frac{C(L_j, L_i) + 1}{C(L_j)} \cdot \frac{N_{C(L_j,L_i)+1}}{N_{C(L_j,L_i)}}, \quad i, j = 1, \ldots, M, \tag{17}$$

$$A^*_{ijk} = P^*(L_i|L_j, L_k) = \frac{C(L_k, L_j, L_i) + 1}{C(L_k, L_j)} \cdot \frac{N_{C(L_k,L_j,L_i)+1}}{N_{C(L_k,L_j,L_i)}},$$
$$i, j, k = 1, \ldots, M. \tag{18}$$

The value N_0 for the zero frequency was obtained by subtracting from the universe size V×V the sum of all observed different pairs $C(L_j, L_i)$(bigram case). We also assumed large counts $C(L_j, L_i)$ and $C(L_k, L_j, L_i)$ to be reliable, if $C(L_j, L_i) > k$ or $C(L_k, L_j, L_i) > k$, where k was set to 5. Rows of **A, B** and $\boldsymbol{\pi}$ matrices had to sum to 1, so probabilities they contained were normalized.

One of the assumptions of Good-Turing is the number N_c decreases monotonically with frequency c, or $N_{c+1} \geqslant N_c > 0$. This assumption was not always satisfied. In cases where equations (17) and (18) resulted in a zero probability, because of $N_{c+1} = 0$, Good-Turing smoothing was not applied. Instead expressions (9) and (12) were used.

3.4. *Tagging*

For the tagging phase we need to find the most likely path through the hidden Markov model given the input. Trying to find the most likely sequence of categories for a given sequence of words, we do not have to enumerate all the possible tag sequences. In fact, tag sub-sequences that end with the same tag can be joined together because of the Markov assumption that the next category only depends on the one (bigram case) previous tag in the sequence (Allen, 1995).

for $i = 1$ **to** M **do** {Initialization Step}
    $D_{i,1} = B_{1,i} \cdot \pi_i^*$
    $X_{i,1} = 0$

for $j = 2$ **to** N **do** {Iteration Step}
    for $i = 1$ **to** M **do**
        $k = \text{argmax}_{u=1,M}(D_{u,j-1} \cdot A_{i,u}^*) \cdot B_{j,i}$
        $D_{i,j} = (D_{k,j-1} \cdot A_{i,k}^*) \cdot B_{j,i}$
        $X_{i,j} = k$

$k = \text{argmax}_i D_{i,N}$ {Sequence Identification Step}
$t_N = L_k$
for $j = N - 1$ **to** 1 **do**
    $k = X_{k,j+1}$
    $t_j = L_k$

Fig. 3. The Viterbi algorithm applied to text tagging (bigram case). Given word sequence $w_1, w_2, \ldots, w_N$, and language model **A, B** and $\boldsymbol{\pi}$ the algorithm finds the most likely sequence of morphological categories $t_1, t_2, \ldots, t_N$ for this word sequence.

We tracked the probability of the best sequence ending with each possible category at each position using array $\mathbf{D}_{M \times N} = \{D_{i,j}\}$ where M is the number of morphological categories $L_1, L_2, \ldots, L_M$ and N is the number of words in the sentence $w_1, w_2, \ldots, w_N$. $D_{i,j}$ contained the probability for the best sequence up to position j, given that the word at this position w_j is tagged with category L_i. Another array, $\mathbf{X}_{M \times N}$, indicated for each category L_i and each position j what the preceding category was in the best sequence at position $j - 1$. The algorithm operated by computing values for these two arrays. This algorithm is known as the Viterbi algorithm (Jurafsky *et al.*, 2000) and is shown in Fig. 3.

While tagging an unknown text, some words were encountered that were absent in the training corpus and consequently were not taken into account by the language model ($B_{ji} = 0$). In such cases, we replaced B_{ji} by a very small[2] value ε.

For a problem involving N words and M lexical categories, the Viterbi algorithm guarantees to find the most likely sequence using $q \cdot N \cdot M^2$ steps, for some constant q, significantly better than the M^N steps required by tracking through all possible paths (Allen, 1995).

4. Evaluation

In order to assess the tagging accuracy of our Lithuanian POS tagger we have conducted three types of experiments by:

1. changing language models (bigram and trigram)

[2]We have chosen $\varepsilon = 10^{-8}$

2. changing smoothing methods (Add-One and Good-Turing)
3. changing the size of training corpus (50%, 75%, 100% of the original training corpus)

Tagging accuracy was computed by comparing machine-annotated and hand-annotated texts on a word-by-word basis. It was measured in percentage of correct tag assignments. The accuracy estimates we obtained are shown in Figs. 4 to 5.

Fig. 4 shows us that the use of trigram LM resulted in poorer performance in comparison to the bigram one, which may seem somewhat strange. In fact, trigram LM is much more sensitive to the lack of training data because of many trigrams are missing in a small training corpus. Our training corpus had a few tens of thousands of words. This is minuscule in comparison to manually annotated corpora of other languages that have millions of words. Further reduction in corpus size resulted in diminishing disambiguation accuracy (see Fig. 5). This is encouraging in the sense that disambiguation accuracy can be expected to increase and to achieve accuracy levels comparable to statistical POS taggers of other languages if our training corpus was extended by more hand-annotated texts.

Fig. 4 also shows us that both Add-One and Good-Turing smoothing techniques resulted in comparable disambiguation accuracy if the bigram LM was used. For the trigram LM, Good-Turing smoothing performed more accurately than Add-One smoothing. This can be explained by the difference in their smoothing schemes. Add-One results in sharp

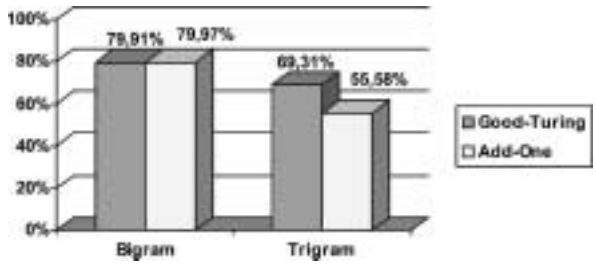

Fig. 4. Disambiguation accuracy obtained with bigram and trigram language models coupled with Add-One and Good-Turing smoothing techniques.

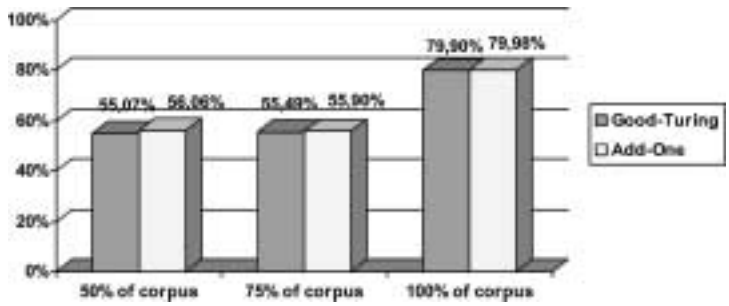

Fig. 5. Disambiguation accuracy obtained using bigram language model together with Add-One and Good-Turing smoothing techniques, and with different sizes of corpus for HMM training.

change in counts and probabilities because too much probability mass is moved to all the zeros. We can try to solve this problem by adding smaller values (add-half) to the counts.

The precision of automatic sentence boundary mark-up was another factor that affected disambiguation accuracy. Our assumption, that every full stop indicated the end of a sentence, greatly oversimplified the recognition of a sentence. The corpus contained lots of abbreviations, names and dates that were followed by full stops. A precise mark-up of sentence boundaries is very important as the stochastic disambiguation techniques are applied on a sentence by sentence basis. Sentence recognition is language dependent and should be refined in future.

5. Conclusions

In this paper we described preliminary experiments in designing HMM-based part-of-speech tagger for Lithuanian language. We focused our attention on the problem of disambiguation, i.e., on the problem of choosing the correct part-of-speech tag for each word in context from a set of possible tags.

We implemented HMM-based stochastic disambiguation procedure. It proceeded in two steps. First, HMM parameters were automatically estimated from the hand-annotated text corpora of 57100 words. Second, word sequences of unknown text were disambiguated assigning the most probable tag (Viterbi algorithm) to each word. We conducted a series of experiments using bigram and trigram language models, and using different smoothing techniques, such as Add-One and Good-Turing. Our Lithuanian part-of-speech disambiguator achieved near 80% disambiguation accuracy and is expected to perform better if the size of the training corpus is increased. The stochastic taggers for other languages, e.g., English, achieve 96–97% accuracy. As the technique described in this paper is based on a word order, it is an interesting topic of future research to find out whether the freedom of word order in Lithuanian influences the accuracy of the approach.

Acknowledgments

The authors express special thanks to dr. D. Hiemstra from the University of Twente, the Netherlands, for his support.

Appendix A

The Tag Set of Lithuanian Parts of Speech

TD	proper noun	PL	preposition
DK	noun	JS	interjection
IV	pronoun	ST	abbreviation
VM	verb	BU	verbal adverb
PV	adverb	AK	acronym
BD	adjective	IT	insertion
SK	numeral	IS	onomatopoeic interjection
DL	particle	NT	other parts of speech
JG	conjunction	SZ	punctuation

References

Allen, J. (1995). *Natural Language Understanding*, 2nd. ed. Benjamin/Cummings, Redwood City, CA.

Armstrong, S., P. Bouillon and G. Robert (1995). *Tools for Part-of-Speech Tagging* (Internal Report). Geneva, Switzerland.

Brill, E. (1995). Unsupervised learning of disambiguation rules for part-of-speech tagging. In *Proceedings of the 3rd Workshop on Very Large Corpora*. Massachusetts.

Charniak, E. (1993). *Statistical Language Learning*. Massachusetts Institute of Technology, Cambridge.

Frantz, A. (1996). *Automatic Ambiguity Resolution in Natural Language Processing*, Springer, Germany.

Gaivenis, K., and S. Keinys (1990). *Kalbotyros terminų žodynas*, Šviesa, Kaunas (in Lithuanian).

Gale, W. (1991). *Good-Turing Without Tears*, Technical report, AT&T Bell Laboratories, New Jersey.

Garside, R., G. Leech and G. Sampson (1987). *The Computational Analysis of English: a Corpus-Based Approach*, UK, London.

Jelinek, F. (1997). *Statistical Methods for Speech Recognition*. Massachusetts Institute of Technology, Cambridge.

Jurafsky, D., and J.H. Martin (2000). *Speech and Language Processing*. Prentice–Hall, New Jersey.

Oostdijk, N.H.J. (1991). *Corpus Linguistics and the Automatic Analysis of English*. Rodopi, Amsterdam.

Zinkevičius, V. (2000). Lemuoklis – morfologinei analizei. *Darbai ir dienos*, **24**, Vytauto Didžiojo universitetas, 245–274 (in Lithuanian).

G. Pajarskaitė (born in 1979) received BSc degree in computer science from the Faculty of Informatics of the Vytautas Magnus University in 2001. At present, she continues her studies at VMU. Her research interests are centered on the application of stochastic techniques to computational linguistic.

V. Griciūtė (born in 1979) received BSc degree in computer science from the Faculty of Informatics of the Vytautas Magnus University in 2001. At present, she continues her studies at VMU. Her research interests are centered on the problem of morphological disambiguation within computational linguistic.

G. Raškinis (born in 1972) received his MSc degree in artificial intelligence and pattern recognition from the University of Pierre et Marie Curie in Paris in 1995. He is a doctor of physical sciences (09P) since 2000. Presently, he is a researcher at the Center of Computational Linguistics of VMU and teaches at the Faculty of Informatics of VMU. His research interests include application of machine learning techniques to computational linguistics.

J. Kuper (born in 1951) studied mathematics at the University of Twente in Enschede, the Netherlands. His master's degree is in the foundations of geometry and set theory, and his PhD is on logical foundations of computer science. He worked at the universities of Leiden and Nijmegen. At the moment he works at the University of Twente as assistant professor in the field of natural language processing.

Paslėptų Markovo modelių taikymas, lietuvių kalbos žodžių daugiareikšmiškumo kalbos dalies atžvilgiu problemai spręsti

Giedrė PAJARSKAITĖ, Vilma GRICIŪTĖ, Gailius RAŠKINIS, Jan KUPER

Šiame straipsnyje pristatomas darbas, skirtas tekstinėje formoje pateiktų lietuvių kalbos žodžių morfologinės kalbos dalies nustatymui. Kalbos dalies nustatymas – tai užduotis priskirti kiekvienam rišlaus teksto žodžiui jį apibūdinančią kalbos dalį, remiantis to žodžio kontekstu. Kalbos dalies nustatymas įgyvendinamas dviem etapais: izoliuotai atliekama žodžių morfologinė analizė, kurios metu kiekvienam žodžiui priskiriama aibė galimų kalbos dalies reikšmių, ir nudaugiareikšminimas, kurio metu iš visų reikšmių paliekama viena, labiausiai tinkama tam kontekstui. Šiame darbe dėmesys koncentruotas ties nudaugiareikšminimo problema. Darbe panaudoti stochastiniai metodai, grindžiami paslėptais Markovo modeliais (HMM). HMM parametrams įvertinti, mokymui panaudoti ekspertų morfologiškai anotuoti tekstai. Maksimaliai tikėtinam keliui per HMM rasti realizuotas Viterbi algoritmas.