Proceedings of the 2014 Federated Conference on Computer Science and Information Systems pp. 163–169
DOI: 10.15439/2014F107
ACSIS, Vol. 2

Ladder Tagger — Splitting Decision Space to Boost Tagging Quality

Mariusz Paradowski, Adam Radziszewski
Institute of Informatics, Wrocław University of Technology
Wybrzeże Stanisława Wyspiańskiego 27, 50-370 Wrocław
Poland

***Abstract*—This paper describes a part of speech tagger. The tagger is based on a set of probability mixture models. Each mixture model is responsible for tagging of a specific class of words, sharing similar context properties. Probability mixture models contain 25 various mixture components. The tagger is tested on Polish language and compared to other available taggers.**

I. Introduction

PART-OF-SPEECH (POS) tagging is a common and well-researched Natural Language Processing (NLP) task. It is the process of assigning POS tags to words and word-like units (*tokens*) in text. In languages with rich morphology the tags usually include significantly more information than just parts-of-speech, e.g. nouns may be specified for values of number, gender and case, adverbs may be specified for degree. In such a setting, the task is referred to as morphosyntactic tagging.

In this paper we present a novel approach to Part-of-Speech tagging (POS tagging), where the labelling is performed in a non-sequential manner using an array of simple probability mixture models. The models are derived directly from the training data. The approach has been developed for Polish, an inflective language that is typically described with a rich, positional tagset. However, very little language-dependent information is in fact employed and it is reasonable to expect that the approach will work equally well for other positional tagsets.

The presented version of the tagger deals only with disambiguation of the grammatical class (roughly corresponding to Part-of-Speech). In spite of that, it already makes use of the information available in the whole tags.

II. Tagging of inflective languages

Tagging of inflective languages is a not an easy task. One of the main difficulty is the relatively free word order, which makes sequences of n-grams much less frequent than in English [1]. The other major difficulty is related to the size and characteristics of tagsets. English tagsets usually define 40–200 different tags, while the 1-million-token manually annotated part of the National Corpus of Polish [2] contains about 1000 different tags. This is a frequently cited reason of low performance of taggers for inflective languages [3]–[5].

Tagging of inflective languages is usually performed in two stages: morphological analysis and morphological disambiguation [6]. The first stage is essentially dictionary look-up, resulting in attaching sets of tags attached to each token. The proper tagging happens during the disambiguation phase — tags that are recognised as contextually inappropriate are removed from the sets. The other technique that is commonly used is called *tiered tagging* (originating from a work by Tufiş [7]). This technique assumes splitting of positional tags according to some groups (tiers) of grammatical categories that the tags consist of. A sentence is disambiguated iteratively, one tier at a time.

Both techniques are used in three taggers made for Polish language during the last five years. PANTERA [8] is built upon an adaptation of the Brill's transformation-based learning algorithm. The tagger employs three tiers, but also makes uses of modified rule templates that operate on the level of particular tagset attributes (grammatical categories) instead of whole tags, which is an important enhancement when dealing with positional tagsets. Also, the sets of tags assigned in the course of morphological analysis are used to constrain possible transformations: if any transformation would result in generation of a tag not accounted for by the morphological dictionary, the transformation is cancelled.

WMBT [9] is a memory-based tagger that introduces morphological analysis and tiered tagging to the standard memory-based tagging framework, namely MBT [10]. The tagger uses as many tiers as there are attributes in the tagset (plus one for the grammatical class). The algorithm iterates over tiers; tagging of one tier involves classification of subsequent tokens with a k-Nearest Neighbour classifier. The classification process benefits from a rich feature set, including values of particular attribute (grammatical class, number, gender and case for tokens surrounding the token being tagged), but also tests for morphological agreement on number, gender and case. WMBT was later enhanced with a simple handling for unknown words [11]. The procedure assumes collecting separate case bases for known and unknown words.

WCRFT [11] is a modification of the WMBT tagger, where a linear-chain first-order Conditional Random Field (CRF) is used instead of the k-NN classifier. Also, instead of classifying independently each token, the CRF model is used to classify a whole sentence at a time. A separate model is trained for each tier and these models are run sequentially when performing disambiguation. The feature set is taken directly from WMBT. Also, similar unknown word handling procedure is employed.

Concraft [12] is based on a new mathematical model

978-83-60810-58-3/$25.00

devised for the purpose of morphosyntactic tagging, namely Constrained Conditional Random Fields (CCRF). CCRF is an extension of the CRF model where additional constraints are imposed on the set of labels that may be produced for each token in the output sequence. The constraints are used to enforce that each output tag belongs to the set of tags generated during morphological analysis. The model used for disambiguation is second-order. An interesting feature of the tagger is that disambiguation is performed on all layers simultaneously instead of using the standard tiered tagging scheme. Concraft also contains a separate model for handling of unknown words, which is based on first-order CCRF.

III. Ladder Tagger

In this paper we propose a new part-of-speech tagger called *Ladder Tagger*. Formally, part-of-speech tagging may be modelled as a sequence I of dependent multi-class decision problems $w \in I$. Decision problems w are statistically dependent because previously made decisions (tag assignments) influence further ones.

Natural language structure is highly complex, often very flexible, full of exceptions. The main idea of the proposed method is to divide the sequence of decision problems I into n subsets of problems with similar properties:

$$D_1 \subseteq I, D_2 \subseteq I, ..., D_n \subseteq I, \tag{1}$$

$$D_1 \cap D_2 \cap ... \cap D_n = \emptyset. \tag{2}$$

Each of these subsets $D_i : i \in \{1, ..., n\}$ should model a part of language complex structure. Statistical dependence of decision problems $w \in I$ makes the final tagging result dependent on the order in which they are solved. Our idea is to solve decision problems from the easiest to the most difficult ones (hence the name of the proposed tagger) in a non-sequential manner. The major advantage of this approach is that both **preceding** and **succeeding** tagging results may influence the current result.

The second contribution of the proposed tagger is highly intense usage of tagging context information. In probabilistic tagging smoothing plays a key role. Smoothing models may be more or less complex, but they should provide statistically significant information regarding token context. Of course smoothing parameters are highly dependent on the decision problem to solve. In the proposed tagger we use as much as 25 probability smoothing components, both taking into account preceding and succeeding tokens. Smoothing parameters are independently estimated for each set of decision problems.

The general outline of the tagging process is presented in Fig. 1. First, we assign each token (decision problem) to a set of decision problems according to its predefined properties. The first set D_1 contains the easiest, unambiguous tokens (they are trivial single-class cases). Remaining sets $D_2, ..., D_n$ contain difficult, multi-class decision problems. Sets of decision problems are now solved according the their difficulty.

Require: I – sequence of decision problems (tokens to be POS-tagged)
Ensure: t – part-of-speech tagging of I
1: $D_1 \leftarrow \emptyset$, $D_2 \leftarrow \emptyset$, ..., $D_n \leftarrow \emptyset$
2: **for all** $w \in I$ **do**
3: $m \leftarrow$get-problem-type(w)
4: $D_m \leftarrow D_m \cup \{w\}$
5: **end for**
6: **for all** $D \in \{D_1, D_2, ..., D_n\}$ **do**
7: **for all** $w \in D$ **do**
8: $t_w \leftarrow$solve-problem(w)
9: **end for**
10: **end for**
11: **return** t

Figure 1. Ladder Tagger algorithm

Our proposal follows the two-stage scheme described in the previous section: tagger's input must be first subjected to morphological analysis and the core algorithm is responsible for morphological disambiguation. In other words, it is assumed that each token is attached a set of *possible tags*[1] and each decision problem consists in selecting one of its elements. The information on *possible tags* is used during both training and normal tagger performance (disambiguation) to designate *ambiguity classes*.

A. Subsets of decision problems

The main goal of decision problems division is to build better, more specific classifiers. One of the key issues in case of POS tagging is proper usage of the available training information. Different type of information should be used in case of e.g. frequent known words, rare known words and unknown words. Thus, the following **classes of decision problems** are defined:

1) specific *known words* (SKN),
2) frequent *known words* (FKN),
3) possible *grammatical class (POS) ambiguity classes* of known words (GKN),
4) rare *known words* (RKN),
5) *word suffixes or prefixes* for unknown words (UNK).

Decision problem subsets are instances of the above classes. Specific known words (SKN) class contains very frequent words with lots of grammar exceptions. List of these words should be pre-specified by expert linguists. The list should not be too long, because classifiers for each specific known word are trained separately. Frequent known words (FKN) class contains frequently used words, generally following grammar rules. Due to their high frequency in the corpus they have a large training data available. *Ambiguity classes* class is much less specific, it often covers multiple known words. Number

[1]Possible according to the morphological analyser employed. Even if a word form is known to the analyser, the returned set of tags may be incomplete or erroneous. Given the evolving nature of language, it is impossible to create an exhaustive morphological dictionary that will be valid for any text.

of known instances of a given word is another important criterion. If a word has only one instance in the training set, any parameter estimation is not possible for such a word. Such words are considered rare and are tagged according to ambiguity classes.

The employed two-stage scheme is bound to face the problem of unknown words. Word forms of some tokens will be not recognised by the morphological analyser, hence no set of possible tags (ambiguity class) will be available. Such tokens are treated as a separate class (UNK). Tagging of such words is language-specific. For Polish (and also for many Slavic languages), word suffixes and, to some extent, prefixes, give useful inflectional information that helps make the correct decision.

During tagging process, a token is treated as an unknown word if the analyser failed to provide a set of possible tags. For our model to work, the UNK class must also be explicitly present in the training data. Fortunately, this information is easy to obtain. To make sure that the sets of possible tags present in the training corpus reflect the actual behaviour of the morphological analyser, we followed the procedure called *reanalysis of the training data* that was proposed in [11]. It assumes feeding the training data through the morphological analyser actually used to update the information on tags *possible* for each token. In case of word forms that were unknown to the analyser, the set of possible tags always consists of the proper tag (taken from the original training data) and the *unknown word tag* (`ign`). What results is consistent assignment of tokens to ambiguity classes, but also, explicit marking of unknown words.

B. Decision model

The proposed tagger applies Bayesian decision model. Maximum a posteriori decision rule is used:

$$t_w^* = \arg\max_{t \in g_w} p(t|w), \tag{3}$$

where: w – a word being currently tagged, t_w^* – chosen tag, g_w – set of possible POS tags for word w.

Class probability $p(t|w)$ is estimated from available training corpus data. Each decision problem subset has its own subset of corpus. Corpus subsets are built according to predefined rules. Rules of corpus subset generation for possible classes of decision problems are given in Tab. I.

Table I
RULES OF CORPUS SUBSET GENERATION FOR VARIOUS DECISION PROBLEM CLASSES.

problem class	word form	grammatical classes	prefix suffix
SKN	match	match	—
FKN	match	match	—
GKN	—	match	—
RKN	—	match	—
UNK	—	—	match

Exemplary, word *to* (eng. *it*) is belongs to the class of specific known word problems. The training set consists of all instances of word *to* with matching set of possible grammatical classes.

C. Probability smoothing

Probability smoothing is one of the key components in statistical approaches to tagging. There is no sufficient data to get well estimated probability distributions of word sequences. One of the most common approach to probability smoothing is Jelinek-Mercer smoothing [13]. The basic model is defined as follows:

$$p(w_0|w_1)_J = \lambda p(w_0|w_1)_D + (1-\lambda)p(w_0)_D, \tag{4}$$

where: $\lambda \in \langle 0;1 \rangle$ and $p(w_0|\cdot)_D$ is estimated directly from the data.

The Jelinek-Mercer model is further extended to Witten-Bell [13] smoothing and it takes a recurrent form:

$$p(w_0|s_i)_R = \lambda_i p(w_0|s_i)_D + (1-\lambda_i)p(w_0|s_{i-1})_R, \tag{5}$$

where: $\lambda_i \in \langle 0;1 \rangle$, s_i is the context of word w_0.

The above model $p(w_0|s_i)_R$ can be simply rewritten into non-recurrent equation and it takes the basic form of a well known probability mixture model. Thus, the probability smoothing used in the proposed approach is defined as follows:

$$p(w_0|s_i)_M = \sum_{j=0}^{i} \alpha_j p(w_0|s_j)_D, \tag{6}$$

where:

$$\alpha_i = \lambda_i \prod_{j=0}^{i-1}(1-\lambda_j), \quad \sum_{j=0}^{i} \alpha_j = 1. \tag{7}$$

Estimation of α_i parameters is not straightforward, and can be done in several different ways. In more complex language models (like the one proposed), the number of parameters may be high.

D. Mixture model probabilities

One of the key components to successful tagging is the definition of smoothing probabilities. As shown above, smoothing may be represented as a mixture probability. Component probabilities are estimated on four different types of data: word forms (w_i), grammatical class (POS) ambiguity classes (g_i), ambiguity classes of whole tags (c_i) and *tagging results* (grammatical classes assigned by the tagger) (t_i), where i stands for relative position of a word in the tagged corpus. In case the word in not yet tagged and a *tagging results* (t_i) is used, the probability is estimated as 0. Given the above events, various conditional probabilities can be defined. We use total 25 probability mixture model components, as shown in Tab. II. For instance, $p(t_0|x_0, g_{-1}, w_{+1})$ represents probability of tag t_0, given that:

1) decision class specific parameters x_0 on position 0 are matched,
2) grammatical class (POS) ambiguity classes g_{-1} on position -1 matches,
3) word form w_{+1} on position $+1$ matches.

Table II
PROBABILITY MIXTURE COMPONENTS USED FOR PROBABILITY SMOOTHING, x_0 REPRESENTS A DECISION CLASS SPECIFIC EVENT. FOR *SKN* AND *FKN* $x_0 = (w_0, g_0)$, FOR *GKN* AND *RKN* $x_0 = g_0$ AND FOR *UNK* x_0 IS DEFINED IN TERMS OF SUFFIXES AND PREFIXES.

$p(t_0\|\cdot)$	word forms (w)	g-class(g)	f-class(c)	tags(t)
$p(t_0\|x_0, w_{+1}, w_{+2})$	two succ.	–	–	–
$p(t_0\|x_0, w_{-1}, w_{-2})$	two prec.	–	–	–
$p(t_0\|x_0, w_{-1}, w_{+1})$	two neigh.	–	–	–
$p(t_0\|x_0, w_{-1}, g_{+1})$	first prec.	first succ.	–	–
$p(t_0\|x_0, g_{-1}, w_{+1})$	first succ.	first prec.	–	–
$p(t_0\|x_0, w_{-1}, t_{+1})$	first prec.	–	–	first succ.
$p(t_0\|x_0, t_{-1}, w_{+1})$	first succ.	–	–	first prec.
$p(t_0\|x_0, t_{-2}, t_{-1})$	–	–	–	two prec.
$p(t_0\|x_0, t_{-1}, t_{+1})$	–	–	–	two neigh.
$p(t_0\|x_0, c_{-2}, c_{-1}, c_{+1})$	–	–	three neigh.	–
$p(t_0\|x_0, w_{-1}, c_{+1})$	first prec.	–	first succ.	–
$p(t_0\|x_0, c_{-1}, w_{+1})$	first succ.	–	first prec.	–
$p(t_0\|x_0, c_{-2}, c_{-1})$	–	–	two prec.	–
$p(t_0\|x_0, c_{-1}, c_{+1})$	–	–	two neigh.	–
$p(t_0\|x_0, w_{+1})$	first succ.	–	–	–
$p(t_0\|x_0, w_{-1})$	first prec.	–	–	–
$p(t_0\|x_0, t_{-1}, g_{+1})$	–	first succ.	–	first prec.
$p(t_0\|x_0, g_{-1}, g_{+1})$	–	two neigh.	–	–
$p(t_0\|x_0, c_{+1})$	–	–	first succ.	–
$p(t_0\|x_0, c_{-1})$	–	–	first prec.	–
$p(t_0\|x_0, t_{+1})$	–	–	–	first succ.
$p(t_0\|x_0, t_{-1})$	–	–	–	first prec.
$p(t_0\|x_0, g_{+1})$	–	first succ.	–	–
$p(t_0\|x_0, g_{-1})$	–	first prec.	–	–
$p(t_0\|x_0)$	–	–	–	–

For the easiness of reading, mixture model components shown in Tab. II are sorted according to their assumed generality. More specific general estimates are shown at the top, more general ones are shown at the bottom of the table. The main idea of probability smoothing is preserved. Mixture components from the top of the table will have low recall, however high precision. Those from the bottom will have high recall and lower precision.

E. Tagger parameter estimation

The proposed tagger has a set of parameters, as described above, which need to be estimated before the tagging process. Parameter estimation is done at the training set. Given the smoothing model has n components and that there are m sets of decision problems, there are total nm parameters to be estimated. For each set of decision problems its smoothing model parameters are estimated independently.

Mixture parameters estimation is considered as a validation set tagging quality optimization problem. Validation sets are built on top of the training set using Leave-One-Out routine. Tagging quality is defined as a classic *recognition accuracy*. Training process is defined as maximization of recognition accuracy:

$$[\alpha]^* = \arg\max_{[\alpha]} \frac{1}{|I|} \sum_{w \in I} |\{t_w^*\} \cap \{r_w\}|, \tag{8}$$

where: r_w – reference tag for word w, t_w^* – tagging result of word w from the validation set.

Random-restart hill climbing method is used as the optimization tool. Exemplary estimates of smoothing parameters are given in Tab. III. Estimated mixture weights may provide an interesting insight into the language structure. They may show the importance of different contexts in various cases of language usage. Exemplary, word *to* (eng. *it*) has a very high importance of succeeding context, while in case of word *i* (eng. *and*) proceeding context is much more important. Thus, results of training may provide further information (and have done so in the past) how to extend available set of contexts.

F. Computational and memory complexity analysis

Proposed tagging approach is can be classified as a *lazy recognition method*, because it estimates probabilities directly from the data during the recognition phase. Thus, it requires memorization of the whole training data, i.e., the tagged corpus. Estimation of mixture probabilities requires iteration through the data, but can be limited only to its small subsets. The whole training set is organized as a *hash map*. Each element of the hash map represents a *decision class specific event* x_0. It contains a list of all indexes of event x_0 appearance in the training set. For example, to estimate $p(t_0|x_0, w_{+1})$ where: $x_0 = (w_0, g_0)$, w_0 is a specific word and g_0 a specific grammatical class, the tagger accesses the hash map with key (w_0, g_0) and extracts a list of relevant indexes. Given the list of indexes, it iterates through the training set and checks only for w_{+1} match condition. This results in a large speedup of tagging process, because size m of the relevant index is always much smaller than the size n of the whole training set.

As a result, memory complexity is equal to $T(kn)$ where n is the size of the corpus and $k = 5$ is the number of predefined decision problem classes (see Tab. I). The number of predefined classes is constant and practically should not

Table III
EXEMPLARY PROBABILITY MIXTURE MODEL PARAMETERS (WEIGHTS) FOR EACH SPACE SUBDIVISION.

$p(t_0\|\cdot)$	*to*(*it*)/SKN	*czy*(*if*)/SKN	*i*(*and*)/SKN	*frequent*/FKN	*rare*/RKN
$p(t_0\|x_0, w_{+1}, w_{+2})$	0.148	0.130	0.050	0.203	0.031
$p(t_0\|x_0, w_{-1}, w_{-2})$	0.069	0.154	0.252	0.086	0.083
$p(t_0\|x_0, w_{-1}, w_{+1})$	0.046	0.104	0.110	0.030	0.005
$p(t_0\|x_0, w_{-1}, g_{+1})$	0.056	0.010	0.004	0.075	0.005
$p(t_0\|x_0, g_{-1}, w_{+1})$	0.039	0.046	0.018	0.007	0.005
$p(t_0\|x_0, w_{-1}, t_{+1})$	0.013	0.003	0.073	0.018	0.041
$p(t_0\|x_0, t_{-1}, w_{+1})$	0.019	0.100	0.022	0.067	0.104
$p(t_0\|x_0, t_{-2}, t_{-1})$	0.046	0.013	0.018	0.037	0.005
$p(t_0\|x_0, t_{-1}, t_{+1})$	0.003	0.040	0.004	0.003	0.005
$p(t_0\|x_0, c_{-2}, c_{-1}, c_{+1})$	0.013	0.043	0.004	0.011	0.072
$p(t_0\|x_0, w_{-1}, c_{+1})$	0.135	0.090	0.041	0.060	0.083
$p(t_0\|x_0, c_{-1}, w_{+1})$	0.023	0.046	0.087	0.060	0.166
$p(t_0\|x_0, c_{-2}, c_{-1})$	0.046	0.003	0.091	0.015	0.005
$p(t_0\|x_0, c_{-1}, c_{+1})$	0.003	0.013	0.004	0.026	0.020
$p(t_0\|x_0, w_{+1})$	0.072	0.003	0.004	0.037	0.072
$p(t_0\|x_0, w_{-1})$	0.115	0.016	0.022	0.086	0.088
$p(t_0\|x_0, t_{-1}, g_{+1})$	0.003	0.003	0.073	0.011	0.020
$p(t_0\|x_0, g_{-1}, g_{+1})$	0.023	0.026	0.022	0.037	0.010
$p(t_0\|x_0, c_{+1})$	0.069	0.016	0.004	0.052	0.119
$p(t_0\|x_0, c_{-1})$	0.029	0.006	0.027	0.011	0.020
$p(t_0\|x_0, t_{+1})$	0.003	0.040	0.004	0.030	0.005
$p(t_0\|x_0, t_{-1})$	0.003	0.020	0.009	0.007	0.005
$p(t_0\|x_0, g_{+1})$	0.006	0.026	0.009	0.003	0.005
$p(t_0\|x_0, g_{-1})$	0.006	0.026	0.018	0.015	0.010
$p(t_0\|x_0)$	0.003	0.010	0.018	0.003	0.005

grow any more ($k = const$), thus memory complexity is $T(kn) = O(n)$. Computational complexity for tagging a single word is equal to $O(im)$, where i is the number of mixture components and m is the size of x_0 relevant index, where $m \ll n$.

IV. EXPERIMENTAL VERIFICATION

In this section we describe our experiments aiming at evaluation of the proposed disambiguation method (Ladder Tagger). Its results are compared to the three Polish taggers described in Sec. II.

As noted earlier, the present variant of the tagger deals only with disambiguation of the grammatical class, hence the scope of evaluation is also limited to labelling with grammatical classes, while the ability to assign full morphosyntactic tags is not assessed.

The experiments described here are made using the manually annotated part of the National Corpus of Polish [2], version 1.0. This part consists of 86 thousand sentences and 1.2 million tokens and we call it NCP in short. Each token is labelled with exactly one tag, belonging to the NCP tagset [14]. The tagset defines 35 grammatical classes (besides one special class reserved for unrecognised forms — `ign`).

A. Experimental protocol

It has been argued [15] that taggers should be evaluated as whole systems that process plain text files into tagged corpora. Such an approach provides insight into tagging errors made at every possible stage, including tokenisation, morphological analysis and disambiguation. This is a close approximation of real-life scenario of tagger application where only text is available, possibly divided into paragraphs, but with no linguistic pre-processing such as manual division into tokens.

We employ this approach here. The taggers are assessed using a metric called *accuracy lower bound* [15] that is defined as the percentage of tokens from the reference corpus (manually divided into tokens and labelled with tags) that fulfil two conditions:

1) the token is present at the tagger output (no change in tokenisation took place),
2) the tagger classified the token with the same label as in the reference corpus.

In other words, every change in tokenisation is penalised as a tagging error and tags attached to tokens that are subjected to segmentation change are not even checked.

The second condition refers to "the same label". As our evaluation is limited to tagging with grammatical classes, the label is understood as grammatical class extracted from full morphosyntactic tag.

All the experiments described here were performed using ten-fold cross-validation. Each experiment is run against the same partitioning of the data. Each run n for a tagger T consists of the following steps:

1) Training data part n is used to train tagger model M_n^T. Before training proper, the training part is subjected to morphological reanalysis as described in Sec. III-A.
2) Testing data part n ($Test_n^T$) is converted to plain text. The division into paragraphs is preserved and marked with two newline characters.
3) Testing data in plain text part n is tagged with the trained model M_n^T and its output is saved to Out_n^T.
4) Out_n^T is compared to $Test_n^T$ and value of accuracy lower

Table IV
ACCURACY LOWER BOUND MEASURED FOR **ALL** TOKENS

Split	LT	WMBT	WCRFT s2	Concraft 5.0	Rank
1	97.22%	96.74%	97.16%	97.13%	1st
2	97.19%	96.70%	97.11%	97.06%	1st
3	97.16%	96.73%	97.07%	97.00%	1st
4	97.28%	96.83%	97.24%	97.08%	1st
5	97.25%	96.73%	97.11%	97.08%	1st
6	97.18%	96.63%	97.02%	96.96%	1st
7	97.21%	96.75%	97.12%	97.05%	1st
8	97.30%	96.79%	97.22%	97.18%	1st
9	97.29%	96.79%	97.22%	97.15%	1st
10	97.23%	96.77%	97.19%	97.15%	1st
μ	97.23%	96.75%	97.15%	97.08%	1st

Table V
ACCURACY LOWER BOUND MEASURED FOR **KNOWN** TOKENS

Split	LT	WMBT	WCRFT s2	Concraft 5.0	Rank
1	97.60%	97.43%	97.55%	97.46%	1st
2	97.53%	97.37%	97.47%	97.35%	1st
3	97.54%	97.35%	97.45%	97.32%	1st
4	97.62%	97.46%	97.60%	97.41%	1st
5	97.60%	97.39%	97.49%	97.39%	1st
6	97.47%	97.21%	97.33%	97.22%	1st
7	97.54%	97.38%	97.47%	97.34%	1st
8	97.63%	97.41%	97.53%	97.46%	1st
9	97.62%	97.41%	97.55%	97.39%	1st
10	97.55%	97.42%	97.52%	97.44%	1st
μ	97.57%	97.38%	97.50%	97.38%	1st

bound (for grammatical class) is calculated.

B. Tagging quality

Table IV presents the observed values of accuracy lower bound across ten runs. The average value (μ) is given in the last row. The experiment shows that Ladder Tagger consistently outperforms all other taggers with respect to grammatical class tagging. The improvement over the second best tagger (WCRFT) corresponds to 2.9% drop in error rate.

We also took the opportunity to measure separate values of accuracy lower bound for two classes of token: known and unknown words. Results for known words (Table V) present the same trends as the overall results: Ladder Tagger consistently outperforms other taggers.

Results observed for unknown words (Table VI) are different. It is evident that Concraft is achieving best results for unknown words. This may be attributed to the advantage given to Concraft by its sophisticated model to deal with unknown words (tag guessing module). Ladder Tagger is placed second or third in the ranking.

V. SUMMARY

This paper presented a part of speech (POS) tagger. Presented POS tagger is based on a set of simple probability mixture models. A list of 25 mixture components is defined. They describe various contexts of the tagged word. Parameters of mixture models are estimated using random-restart hill climbing method. Tagging accuracy of all and known tokens is highest among all tested taggers. Tagging accuracy of unknown tokens (not recognized by tag guessing module) is ranked either second or third.

Table VI
ACCURACY LOWER BOUND MEASURED FOR **UNKNOWN** TOKENS

Split	LT	WMBT	WCRFT s2	Concraft 5.0	Rank
1	85.13%	75.19%	85.13%	86.94%	3rd
2	86.09%	74.80%	85.37%	87.52%	2nd
3	85.48%	77.56%	85.45%	87.14%	2nd
4	86.27%	76.22%	85.40%	86.44%	2nd
5	86.09%	75.64%	84.95%	87.02%	2nd
6	87.02%	76.19%	86.23%	87.76%	2nd
7	86.21%	75.98%	85.63%	87.50%	2nd
8	86.35%	76.28%	86.96%	87.72%	3rd
9	86.58%	76.90%	86.50%	89.45%	2nd
10	86.93%	76.23%	86.63%	87.98%	2nd
μ	86.22%	76.10%	85.83%	87.55%	2nd

The results obtained are encouraging enough to extend the presented approach to cover full positional tagset — and this is a priority for us at the moment. The other worthy line of research is to improve handling of unknown words.

ACKNOWLEDGMENT

This work was financed by Innovative Economy Programme project POIG.01.01.02-14-013/09 (http://www.ipipan.waw.pl/nekst/).

REFERENCES

[1] S. Sharoff, "What is at stake: a case study of Russian expressions starting with a preposition," in *Proceedings of the Workshop on Multiword Expressions: Integrating Processing*. Association for Computational Linguistics, 2004, pp. 17–23.

[2] A. Przepiórkowski, R. L. Górski, M. Łaziński, and P. Pęzik, "Recent developments in the National Corpus of Polish," in *Proceedings of the Seventh International Conference on Language Resources and Evaluation, LREC 2010*. Valletta, Malta: ELRA, 2010.

[3] B. Vidová-Hladká, "Czech language tagging," Ph.D. dissertation, Charles University, Faculty of Mathematics and Physics, Prague, 2000.

[4] J. Hajič, P. Krbec, P. Květoň, K. Oliva, and V. Petkevič, "Serial combination of rules and statistics: A case study in Czech tagging," in *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 2001. doi: 10.3115/1073012.1073047 pp. 268–275. [Online]. Available: http://dx.doi.org/10.3115/1073012.1073047

[5] M. Piasecki and G. Godlewski, "Effective architecture of the Polish tagger," in *Text, Speech and Dialogue*, vol. 4188. Brno, Czech Republic: Springer, 2006. doi: 10.1007/11846406_27 pp. 213–220. [Online]. Available: http://dx.doi.org/10.1007/11846406_27

[6] J. Hajič and B. Vidová-Hladká, "Tagging inflective languages: Prediction of morphological categories for a rich, structured tagset," in *Proceedings of the COLING - ACL Conference*. ACL, 1998. doi: 10.3115/980845.980927 pp. 483–490. [Online]. Available: http://dx.doi.org/10.3115/980845.980927

[7] D. Tufiş, "Tiered tagging and combined language models classifiers," in *Text, Speech and Dialogue*, ser. Lecture Notes in Computer Science, V. Matousek, P. Mautner, J. Ocelíková, and P. Sojka, Eds. Springer Berlin / Heidelberg, 1999, vol. 1692, pp. 843–843. [Online]. Available: http://dx.doi.org/10.1007/3-540-48239-3_5

[8] S. Acedański, "A morphosyntactic Brill tagger for inflectional languages," in *Advances in Natural Language Processing*, ser. Lecture Notes in Computer Science, H. Loftsson, E. Rögnvaldsson, and S. Helgadóttir, Eds. Springer Berlin / Heidelberg, 2010, vol. 6233, pp. 3–14. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-14770-8_3

[9] A. Radziszewski and T. Śniatowski, "A memory-based tagger for Polish," in *Proceedings of the 5th Language & Technology Conference, Poznań*, 2011.

[10] W. Daelemans, J. Zavrel, A. Van den Bosch, and K. van der Sloot, "MBT: Memory-Based Tagger, version 3.2." ILK, Tech. Rep. 10-04, 2010.

[11] A. Radziszewski, "A tiered CRF tagger for Polish," in *Intelligent Tools for Building a Scientific Information Platform*, ser. Studies in Computational Intelligence, R. Bembenik, Ł. Skonieczny, H. Rybiński, M. Kryszkiewicz, and M. Niezgódka, Eds. Springer Berlin Heidelberg, 2013, vol. 467, pp. 215–230. ISBN 978-3-642-35646-9. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-35647-6_16

[12] J. Waszczuk, "Harnessing the CRF complexity with domain-specific constraints. The case of morphosyntactic tagging of a highly inflected language," in *Proceedings of the 24th International Conference on Computational Linguistics (COLING 2012)*, Mumbai, 2012, pp. 2789–2804.

[13] S. F. Chen and J. Goodman, "An empirical study of smoothing techniques for language modeling," in *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, ser. ACL '96. Stroudsburg, PA, USA: Association for Computational Linguistics, 1996. doi: 10.3115/981863.981904 pp. 310–318. [Online]. Available: http://dx.doi.org/10.3115/981863.981904

[14] A. Przepiórkowski, "A comparison of two morphosyntactic tagsets of Polish," in *Representing Semantics in Digital Lexicography: Proceedings of MONDILEX Fourth Open Workshop*, V. Koseska-Toszewa, L. Dimitrova, and R. Roszko, Eds., Warsaw, 2009, pp. 138–144.

[15] A. Radziszewski and S. Acedański, "Taggers gonna tag: an argument against evaluating disambiguation capacities of morphosyntactic taggers," in *Text, Speech and Dialogue*, ser. Lecture Notes in Computer Science, P. Sojka, A. Horák, I. Kopeček, and K. Pala, Eds. Springer Berlin Heidelberg, 2012. doi: 10.1007/978-3-642-32790-2_9 pp. 81–87. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-32790-2_9