Computing and Informatics, Vol. 26, 2007, 627–647

A CASE STUDY OF ALGORITHMS FOR MORPHOSYNTACTIC TAGGING OF POLISH LANGUAGE

Marcin Kuta, Paweł Chrzaszcz

Institute of Computer Science, AGH University of Science and Technology
al. Mickiewicza 30, Cracow, Poland
e-mail: mkuta@agh.edu.pl, pchrzasz@student.agh.edu.pl

Jacek Kitowski

Institute of Computer Science, AGH University of Science and Technology
al. Mickiewicza 30, Cracow, Poland
&
Academic Computer Centre CYFRONET AGH
ul. Nawojki 11, Cracow, Poland
e-mail: kito@agh.edu.pl

Manuscript received 29 May 2007; revised 20 June 2007
Communicated by Michal Laclavík

Abstract. The paper presents an evaluation of several part-of-speech taggers, representing main tagging algorithms, applied to corpus of frequency dictionary of the contemporary Polish language. We report our results considering two tagging schemes: IPI PAN positional tagset and its simplified version. Tagging accuracy is calculated for different training sets and takes into account many subcategories (accuracy on known and unknown tokens, word segments, sentences etc.) The comparison of results with other inflecting and analytic languages is done. Performance aspects (time demands) of used tagging tools are also discussed.

Keywords: Machine learning, part-of-speech tagging, natural language processing

Mathematics Subject Classification 2000: 68T50, 68T05, 68T35

1 INTRODUCTION

With information systems becoming more and more complex it would be desirable to express users' demands in natural language. This requires sophisticated natural language processing tools, which cannot be developed without efficient part-of-speech (POS) tagging algorithms, i.e., algorithms assigning to each word values of categories like part of speech, gender, case, etc. A set of values of these categories is called a tag.

POS tagging is a foundation of applications like word sense disambiguation sentence chunking, syntactic and semantic parsing, automatic translation ontology construction from text or message understanding [5].

The tagger builds a model of language, i.e., distribution of tags over tokens, which is adjustable by many parameters. The parameters are determined during the training process on the basis of a set of examples, called a training set. The quality of a language model is verified against a test set.

If tagging analytic languages (e.g. English) seems to be well studied, efficient POS tagging of inflective or agglutinative languages is still a challenge due to large tagset sizes required for such languages.

In the case of inflective languages tags describe much more morphological categories than only part of speech and we speak then rather about morphosyntactic tagging than POS tagging.

Due to the MULTEXT-East project [29] based on the Orwell's 1984 annotated novel results on tagging of inflecting and agglutinative languages (Czech, Estonian, Hungarian, Romanian, Slovene) have appeared. However, as reported in [1] about POS taggers, *"their availability is highly dependent on the language, from almost unlimited numbers for English, over a few different POS taggers for German or Swedish, to practically nothing for a language like Polish."*

The aim of the paper is to provide evaluation of baseline tagging algorithms on Polish corpus, continuing in this way so far work done, e.g., for the Slovene [8] or German [28]. We tested freely available, universal tagging tools and the obtained results can be used as a reference point for comparison with more sophisticated, state of the art taggers. Our results are informally compared with those already obtained for the Polish language [19, 20].

The evaluation of these universal tools would be useful when natural language processing is intended to support modern information technologies based on semantics and ontology-based knowledge while not only tagging accuracy but also other factors, like performance and flexibility, are in focus. One of such examples is support for ontology generation from free text notes [17].

The rest of the paper is organized as follows. Section 2 presents state of the art in the area of tagging algorithms. Section 3 introduces the most important POS tagging libraries. Section 4 discusses algorithms selected to evaluation in more details. Section 5 describes the structure of evaluated corpora. A tagging experiment is discussed in Section 6 and the obtained results supplied in Section 7. Section 8 gives conclusions and short summary closes the paper.

2 OVERVIEW OF TAGGING ALGORITHMS

Up to now many approaches to the problem of POS tagging have been subject of research.

The first two algorithms belong to statistical methods. Statistical algorithms map sequence of tokens into sequence of tags with probability model, which describes the occurrence of the most probable sequence of tags for a given sequence of tokens.

Hidden Markov Model (HMM) taggers were first applied historically and are the most popular type of taggers. Such taggers are most suitable for POS tagging of the analytical languages due to their weak inflection. A gentle introduction to the HMM methodology can be found in [24].

Maximum entropy taggers aim to maximize the entropy function by joining the HMM approach with selection of binary features, reflecting dependency in the training corpus. The method was introduced into the NLP area by A. Ratnaparkhi [25]. Applying the maximum entropy method to inflecting languages comes with difficulties because of usually large tagset sizes and intra dependencies occurring in such languages.

The next two algorithms are called rule based methods. The rule based methods take into account a wide context, what is desirable in the case of languages containing distant syntax dependencies.

Memory-based learning (a.k.a. lazy learning, example-based learning) taggers acquire examples from training corpora, which are later used in the tagging process [6].

Transformation-based error-driven learning (a.k.a. transformation-based or Brill [2]) taggers are the most popular type of taggers not based on probabilistic models. The main advantage of exploited method is a possibility of supplying and modifying rules by linguists. Beside the training and test corpora the method requires additional patch corpus.

Support Vector Machines SVM approach to POS tagging has been investigated in [11]. The SVMTool achieved accuracy of 97.16 % for English and 96.89 % for Spanish.

Decision trees taggers have much in common with conventional HMM taggers, but in contrast to them estimate transition probabilities with binary decision trees [26].

Neural networks Neural networks as a POS tagging tool have been subject of research in [27]. The reported results for the English language are comparable with HMM approach. The disadvantage of the method is a slow training process.

Genetic programming A tagger making use of genetic programming operators (selection, crossing etc.) for rules acquisition has been proposed in [10].

Manual finite state rules In this approach linguistic distributional rules are converted to finite-state machines [30]. Expert knowledge may be required to write the rules.

The following two methods clearly divide task of morphosyntactic tagging into context free morphological analysis and context morphosyntactic disambiguation.

Hajič method The Hajič method is based on the maximum entropy approach. In order to relax computational complexity of the problem (Lagrangian minimisation) a naive Bayes assumption is made. The algorithm achieves satisfactory results for inflecting language (like Czech [14, 15]).

Rules of striking off The method is a modification of transformation-based tagging [3]. In the first stage each token is assigned many tags. Next, special rules eliminate (strike off) redundant tags.

The next method is not a standalone tagging algorithm but tries to compensate errors of different approaches.

Combined classifying A few independent taggers are involved in task in parallel or sequential manner [16]. Their outputs are compared and combined using several voting strategies. The combined system acts better than the most accurate component.

More thorough discussion of some of the above algorithms can be found in [7].

3 POS TAGGING LIBRARIES

POS tagging algorithms have been subject to many implementations. The implementations come as standalone programs or appear as part of versatile systems supporting research and development in NLP. The advantage of such complex frameworks is that besides POS tagging they provide other NLP services such as: sentence boundaries detection, named entity recognition, lemmatisation, chunking, syntactic parsing, etc. Frameworks provide also API to integrate the offered functionality with user's programs. The platforms described below are available at no cost for academic research.

QTag library for Java provides a probabilistic HMM tagger. QTag is chosen by authors of the Baseline Information Extraction (BALIE) system [33] as tagging component. BALIE is prepared for tagging of English, German, French, Spanish and Romanian. The small tagset consists of the tags common to all above languages.

MAXENT library for Java [36] is a part of openNLP environment [39]. The tagger exploits maximum entropy approach. It especially supports English (tagger pretrained on the Wall Street Journal and Brown Corpus) and Spanish.

NLTK library [38] is a NLP toolkit written in Python equipped with a HMM tagger. Without additional effort the tagger can be used to annotate English texts with tags from the Brown tagset. The simplicity of the Python language decides that NLTK is a choice when a rapid prototype of a NLP application is required. NLTK is distributed with various interesting corpus resources.

LingPipe [35] is a suite of Java libraries containing among others a HMM tagger. The tagger is initially prepared for working on general English texts (pretrained on the Brown corpus) and for applications in biomedical domain (pretrained on English biomedical corpora: GENIA and MedPost).

FreeLing [34] is a set of C++ tools that comes with two types of taggers: a HMM tagger and a relaxation labelling tagger [18]. Currently FreeLing supports English and Romance languages: Spanish, Catalan, Galician, and Italian. The library would be advised when overhead introduced by interpreted languages is unacceptable.

μ-TBL [37] system is a suite of tools for transformation-based learning. μ-TBL is easily extensible and efficient system that uses capabilities of the Prolog programming language to implement a generalized form of transformation-based learning. μ-TBL supports transformation-based and constraint grammar taggers.

4 EVALUATED ALGORITHMS

In our work we investigate further a selected set of four algorithms, discussed below in more detail.

Hidden Markov Model Given sequence of tokens, $w_1, \ldots, w_n$, the HMM tagger assigns a sequence of tags, $T = (t_1, \ldots, t_n)$, according to the formula

$$\hat{T} = \arg\max_T \prod_i^n p(w_i|t_i) \cdot p(t_i|t_{i-1}, \ldots, t_{i-N}), \tag{1}$$

where $p(w_i|t_i)$ is the conditional probability of occurrence of word w_i given tag t_i occurred and $p(t_i|t_{i-1}, \ldots, t_{i-N})$ is the conditional probability of occurrence of tag t_i given tag sequence $t_{i-1}, \ldots, t_{i-N}$ previously occurred.

Markov model of N-th order is called $(N+1)$-gram model. The most probable sequence of tags is computed with help of Viterbi algorithm. The learning of tagger is based on maximum likelihood estimation.

Maximum entropy The model assumes a set of binary features, f_j, is defined on the combination of a tag t_i and its context c. The probabilistic model is built from family of models

$$p(t_i, c) = \pi\mu \prod_j \alpha_j^{f_j(t_i,c)}, \tag{2}$$

where $p(t_i, c)$ stands for joint distribution of tags and contexts and π, μ are normalisation factors in order that $p(\cdot, \cdot)$ forms the probability function.

During the training of the tagger weights, α_j, are computed by iterative scaling procedure.

Memory-based learning Memory-based learning algorithms may differ in definition of similarity, the way the instances are stored in memory and the way the search through memory is conducted.

During the learning process memory-based taggers store in memory a set of examples (t_i, c_i), where t_i denotes the tag and c_i its context. Given a token w in context c, the memory-based tagger assigns it a tag t_k, such that distance between c and c_k is minimal.

The following distance metrics have been proposed: overlap metric, information gain weighting, chi-squared feature weighting, modified value difference metric, Jeffrey divergence metric and dot-product metric. The examples can be stored in memory as tables, trees or even self-organising maps.

A good representation of memory-based learning technique is the IGTree algorithm, which uses information gain metric and tree representation of stored examples with special heuristic of its searching.

Transformation-based error-driven learning The tagger starts with assigning a trivial sequence of tags to a given tokenised text. The target sequence of tags is determined by applying series of transformations. Each transformation, F, is a rule in the form: "Replace value of tag t with value y if current context c fulfils condition ϕ." The core of learning process is the algorithm for finding suitable transformations.

4.1 Sample Results of the Evaluated Algorithms Used for Other Languages

Accuracy of tagging algorithms applied to analytic language (English) and inflective language (Slovene) is compared in Table 1. Accuracy is defined as a ratio of the number of correctly tagged tokens to the number of all tokens (see Equation (3) for further details).

From the table one can infer that the HMM algorithm is best performing both for English and Slovene languages, but the results are not comparable directly. The accuracy can be influenced by several factors, e.g., profile of training and test corpus, structure of tagset, ambiguity of corpus, presence of morphological analyser, whether punctuation characters were excluded while computing accuracy (accuracy on tokens vs. accuracy on words), etc.

The reported results come from evaluation on the Wall Street Journal Corpus for English and on the Orwell's novel 1984 for Slovene.

Algorithm	Tagger	*accuracy* [%]	
		English	Slovene [8]
HMM	TnT [4]	96.7 [4]	89.22
Maximum entropy	MET [25]	96.63 [25]	86.36
Memory based	MBT [6]	96.4 [6]	86.42
Transformation based	RBT [2]	96.6 [2]	85.95

Table 1. Accuracy of tagging algorithms applied to analytic and inflecting languages (in square brackets the source of data is given)

5 STRUCTURE OF EVALUATED DATA

In the study a corpus of frequency dictionary of contemporary Polish [31], which is based on texts published between 1963 and 1967 is used. The tagged corpus available under GNU licence since 2001 consists of 5 parts of approximately equal size representing different themes: scientific texts, news, essays, fiction, plays and thus represents also different styles of the language.

In the purpose of taggers evaluation we examine 2 versions of the corpus: (1) structured with the IPI PAN positional tagset [32] (we call this tagset a complex tagset), (2) structured with the reduced version of complex tagset with only POS attribute considered (referred to as simple tagset further).

By a token we mean the smallest entity being subject of tagging. The specific feature of the IPI PAN tagset is that certain input lexemes are divided into sequence of tokens, each token has its tag assigned (e.g. lexeme `chciałbym` is represented in corpus by 3 tokens: `chciał-by-m`). Such a construction of tagset is motivated by the phenomenon called mobile or floating inflection and is more widely discussed in [22, 23] as a part of justification of the tagset design. A token is classified as a word segment if it contains at least one alphanumeric character (including Polish diacritics) or digit. The remaining tokens represent punctuation marks.

A small part of the corpus data (tokens) is inherently ambiguous and has more than one tag assigned. For the purpose of training all taggers require that each token has exactly one tag assigned in the training set. To fulfil this assumption at first the corpus is preprocessed to eliminate all cases of inherent ambiguity.

If token w_i has been assigned a sequence of correct tags, $t_{i1}, \ldots, t_{ik_i}$ (k_i – number of tags assigned to token w_i), in the original corpus, the annotation $(w_i; t_{i1}, \ldots, t_{ik_i})$ is replaced by $(w_i; t_{ij})$, with tag t_{ij} selected in random manner amongst $t_{i1}, \ldots, t_{ik_i}$.

5.1 Design of Data Structure for the Experiment

To carry out the experiment the corpus is processed in the following way. Five subcorpora S_i ($1 \leq i \leq 5$) are created from the available corpus, each subcorpus corresponding to different theme (see Figure 1). Each of the five subcorpora, S_i, is next divided into twenty subparts S_{ij} ($1 \leq i \leq 5$, $1 \leq j \leq 20$) of equal size so that $S_i = \cup_{j=1}^{20} S_{ij}$. Further, the subparts S_{ij} ($1 \leq j \leq 20$) are coalesced into parts

P_j according to formula $P_j = \bar{\cup}_{i=1}^{5} S_{ij}$, where $\bar{\cup}$ denotes coalescing operator defined in the following way: $P_\alpha = S_\beta \,\bar{\cup}\, S_\gamma$ $(\alpha, \beta, \gamma \in \mathbf{N})$ if and only if set P_α is created by appending set S_β to the end of S_α [1]. We get twenty parts P_j, each containing 5 % of the full corpus. Next, it suffices to coalesce relevant number of parts P_j to create training and test sets. The test set is created as $P_{10} \,\bar{\cup}\, P_{20}$ (ten percent of full corpus), remains constant for all experiments with a given tagset and is independent from the training sets. The training sets are generated in the following sizes: 5, 10, 20, 40, 80 and 90 percent of corpus. The entire process is depicted in Figure 1.

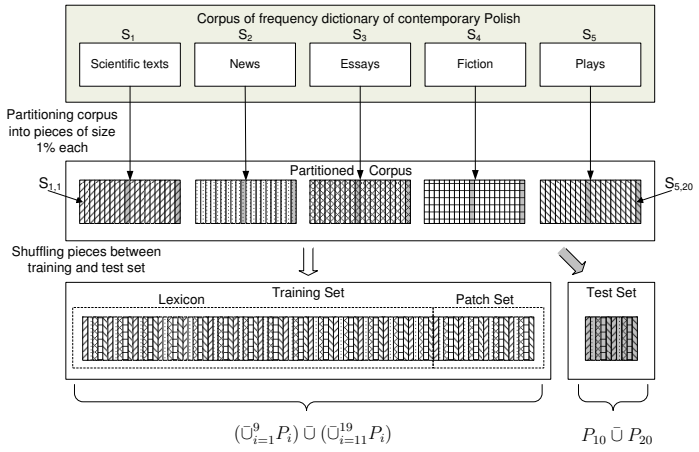


Fig. 1. Preparation of test set and example training corpus (the split into lexicon and patch corpus used by fnTBL tagger)

The most important parameters of corpus, test and training sets are summarized in Tables 2 and 3.

While presenting corpus parameters we face the choice of different ways of calculating mean token ambiguity, referred to in [12] as automatic and independent dictionary methods. The independent method takes into account, for a given token, all possible tags proposed by the morphological analyser, while the automatic method considers only those tags, which are found in corpus as a possible tagging of the considered token. As we do not investigate the influence of the morphological analyser on tagging accuracy, the automatic method was chosen for computing mean token ambiguity.

6 THE EXPERIMENT

In order to keep training and test corpus profiles similar and to ensure they are representative samples of the language, the scenario depicted in Figure 1 has been applied.

[1] $P_j = \bar{\cup}_i \, S_{ij}$ implies $P_j = \cup_i \, S_{ij}$, but the reverse does not hold in general

	Training set						Test	Full
	5 %	10 %	20 %	40 %	80 %	90 %	10 %	100 %
tokens	32 949	65 876	131 837	263 739	527 601	593 511	66 000	659 511
word segments	27 472	55 315	110 318	221 186	442 328	497 533	55 206	552 739
sentences	2 119	3 878	8 120	16 258	32 746	36 647	4 215	40 862
different tokens	11 712	19 689	32 828	52 754	81 567	87 160	19 580	92 942
	Simple tagset							
tagset size	28	28	29	29	30	30	29	30
ambiguous tokens, %	26.67	26.93	26.21	26.38	26.36	26.42	26.44	26.43
mean token ambig.	1.48	1.49	1.47	1.48	1.48	1.48	1.48	1.48
different base forms	6 797	10 249	15 450	22 522	32 114	33 890	10 215	35 736
	Complex tagset							
tagset size	653	756	896	1 031	1 206	1 228	812	1 270
ambiguous tokens, %	49.35	49.82	48.93	49.25	49.33	49.40	49.29	49.93
mean token ambig.	3.42	3.40	3.36	3.38	3.40	3.40	3.40	3.40
different base forms	6 796	10 248	15 450	22 522	32 114	33 890	10 217	35 739

Table 2. Corpus parameters

	Size of training set					
	5 %	10 %	20 %	40 %	80 %	90 %
unseen tokens, %	29.07	23.49	17.86	13.45	9.90	9.37
tokens with unseen lemmas, %	14.68	10.58	7.08	4.93	3.40	3.21
unseen different lemmas, %	61.47	49.88	37.28	27.31	19.20	18.07
	Simple tagset					
tokens with unseen tags	388	386	360	327	316	315
unseen different tags, %	3.48	3.48	0	0	0	0
	Complex tagset					
tokens with unseen tags	3 563	3 524	3 249	2 895	2 539	2 442
unseen different tags, %	28.69	20.69	13.42	9.11	5.17	5.17

Table 3. Parameters of test corpus in relation to training corpus size

We evaluate five taggers cited in Table 4, all of them are freely available. For their descriptions we refer the reader to [4, 9, 28].

6.1 Optimisation Algorithm

Given training and test sets a problem of finding optimal parameters of training process arises. The aim of the optimisation algorithm is to find, for each tagger independently, training parameters, which allow to obtain the highest accuracy on tagging.

Each of the considered taggers contains a set of parameters $p_1^k, \ldots, p_{n_k}^k$ (n_k is the number of parameters of k-th tagger, p_i^k stands for i-th parameter of k-th tagger, each p_i^k accepts the values belonging to set V_i^k) influencing tagging accuracy (quality

k	Tagger name	Algorithm	Package
1	T3	HMM	ACOPOST
2	TnT	HMM	TnT
3	MET	Maximum entropy	ACOPOST
4	fnTBL	Transformation based	fnTBL
5	ET	Memory based	ACOPOST

Table 4. Characteristics of taggers used in experiment

of solution). In order to find optimal set of parameters of the k^{th} tagger we had to approximate maximum of function $f_k : V_1^k \times V_2^k \times \cdots \times V_{n_k}^k \mapsto [0,1]$ describing tagging accuracy. The following optimization algorithm, making use of direction set (Powell's) method [21], has been applied:

1: **for** $k := 1$ to 5 **do**
2: **for** $i := 1$ to n_k **do**
3: $v_i^k :=$ default value of parameter p_i^k
4: **end for**
5: $iteration := 0$
6: $last := 0$
7: $finish :=$ **false**
8: **repeat**
9: **for** $i := 1$ to n_k **do**
10: **if** $last = i$ or ($last = 0$ and $iteration > 0$) **then**
11: $finish :=$ **true**
12: **continue**
13: **end if**
14: train k-th tagger on training set for different values of p_i^k belonging to V_i^k with other parameters frozen
15: find value $pMax$ of parameter p_i^k for which f_k (accuracy on all tokens measured on test set) is the biggest. If maximum of function f_k is reached for more than one value of parameter p_i^k, accuracy on sentences is deciding.
16: **if** $pMax \neq v_i^k$ **then**
17: $v_i^k := pMax$
18: $last := i$
19: **end if**
20: **end for**
21: $iteration := iteration + 1$
22: **until** $finish$
23: **end for**

Additionally the algorithm guarantees to find global maximum (with accuracy of one step) if f_k (viewed as a function of one parameter with others parameters frozen) is unimodal for each parameter and global maximum is located inside the space

$V_1^k \times V_2^k \times \cdots \times V_{n_k}^k$. While these assumptions cannot be guaranteed theoretically, they are of no high practical significance – experimental verification of the optimisation procedure has shown that high accuracy could be obtained (cf. Section 6). The algorithm does not evaluate gradient, what would be highly problematic for space of parameters with many dimensions taking discrete and binary arguments.

The experiments were performed at the ACC Cyfronet AGH-UST site under SUSE Linux Enterprise Server 9 on the SGI Altix 3 700 supercomputer equipped with 128 Itanium 2 processors and 256 GB memory. The advantage of using the Altix 3 700 machine was in availability of large operational memory.

To automate corpora preparation, tagset manipulations, taggers invocations and results evaluation, special framework in Java and Perl has been prepared. The JVM used was BEA JRockit 1.5, virtual machine optimised for Intel architectures.

The optimisation algorithm has been executed in semi-manual manner due to constraints and occurring instabilities like server restarts, JVM and tagger crashes or time limits imposed by queuing system.

7 RESULTS

Assuming the reference test corpus contains n tokens $w_1, \ldots, w_n$, token w_i is annotated in the corpus with tags $t_{i1}, \ldots, t_{ik_i}$ and tagger guesses for token w_i tag g_i, the accuracy is defined as follows:

$$accuracy = \frac{\#\text{correctly tagged tokens}}{\#\text{all tokens}} = \frac{\sum_{i=1}^{n} |\{g_i\} \cap \{t_{ij}\}|}{n}, \tag{3}$$

where tag t_{ij} was randomly selected from $\{t_{i1}, \ldots, t_{ik_i}\}$ during preprocessing stage of corpus data preparation for the experiment.

In most cases token w_i is annotated with exactly one tag, t_{i1}, but notation also reflects the possibility of inherent ambiguity.

Figures 2, 3 and 4, 5 show dependency between training set accuracy and size. Detailed results for taggers trained on 90 % of the corpus are gathered in Tables 5 and 6. Table 6 presents the mean wall clock time of execution of a tagger instance. The time relevant to initial disk operations (reading of configuration files, training and test corpora, internal taggers' files) was not taken into account.

The T3 tagger is not able to operate on larger tagsets, what explains lack of results for this tagger on complex tagset. Consecutive trainings of the fnTBL tagger on the same training set and with the same values of parameters lead to slightly different accuracy values achieved on test set. This is due to random feature of the fnTBL tagger.

8 CONCLUSIONS

Based on the gathered results the following conclusions can be drawn:

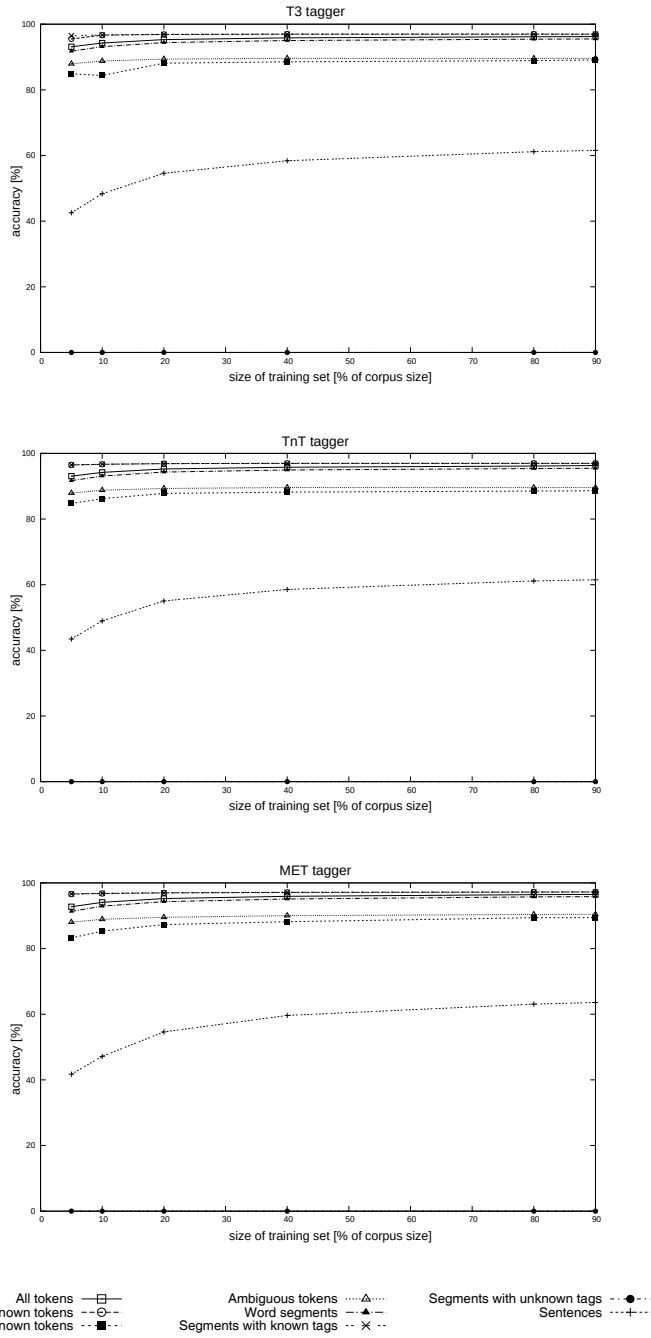


Fig. 2. Accuracy for simple tagset in relation to training corpus size (T3, TnT and MET taggers)

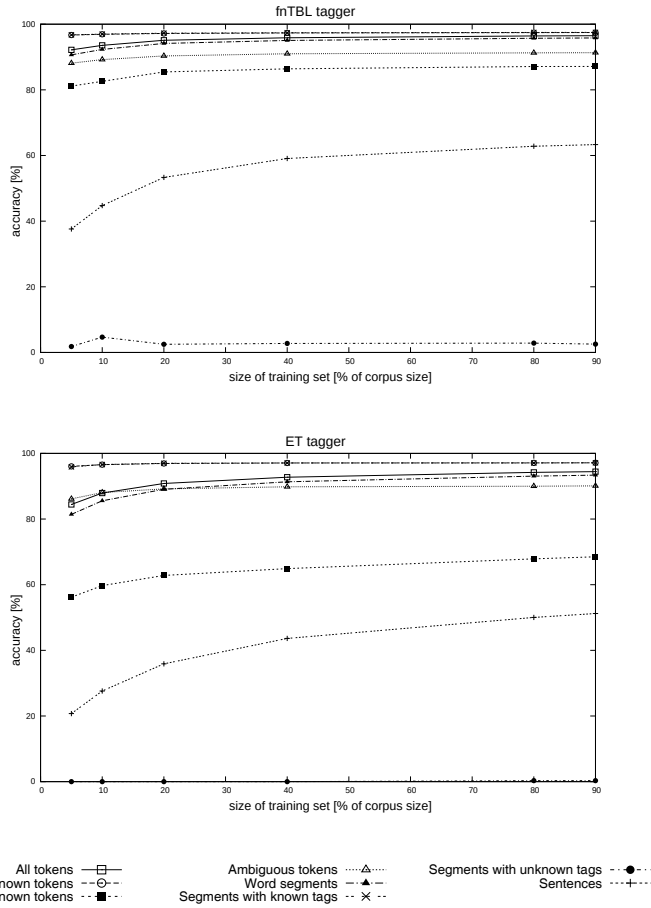


Fig. 3. Accuracy for simple tagset in relation to training corpus size (fnTBL and ET taggers)

- Optimisation procedure allows to achieve high tagging accuracy.
- The considered taggers differ slightly in tagging accuracy but significantly in speed performance.
- fnTBL tagger combines very good time performance with high accuracy (the highest accuracy on all tokens for both simple and complex tagset). Its training process is, however, significantly longer than in case of other taggers (except MET) and clearly depends on tagset size. It results from the fact that for large tagset the number of generated rules is significantly larger. If applied to texts with many tokens unseen during training, fnTBL is outperformed by TnT.

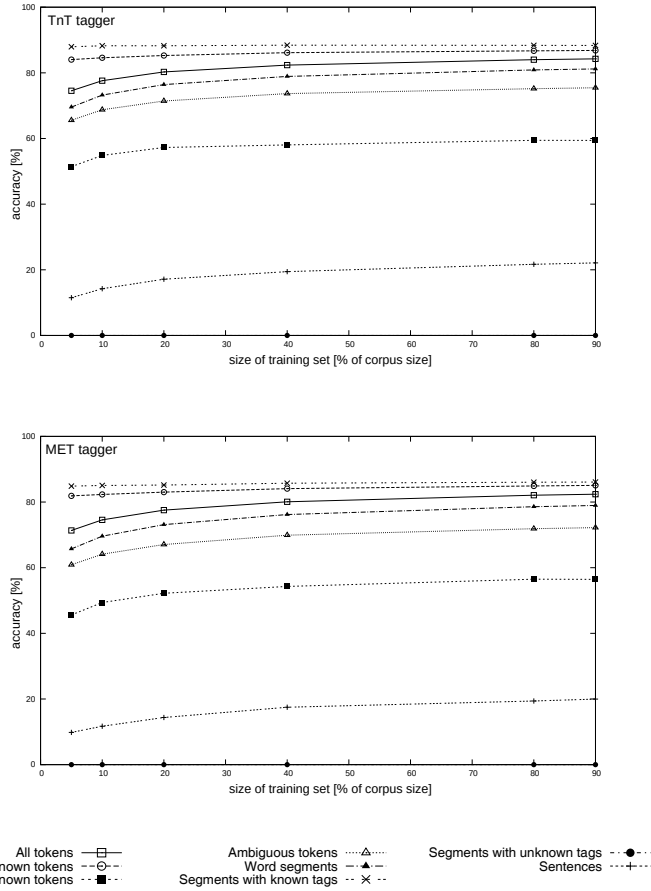


Fig. 4. Accuracy for complex tagset in relation to training corpus size (TnT and MET taggers)

fnTBL guesses correct tags for tokens with previously unseen tags in over one fourth of cases.

- TnT is definitely the fastest and remains insensitive to enlargement of tagset.
- T3 tagger is not suitable for large tagsets due to rapidly growing time and memory requirements. However, when applied to small tagset it achieves both satisfactory performance and accuracy.
- MET is characterized by extremely slow training and for large tagsets also by slow tagging. Thus, despite its high accuracy it compares unfavourably with TnT and fnTBL.

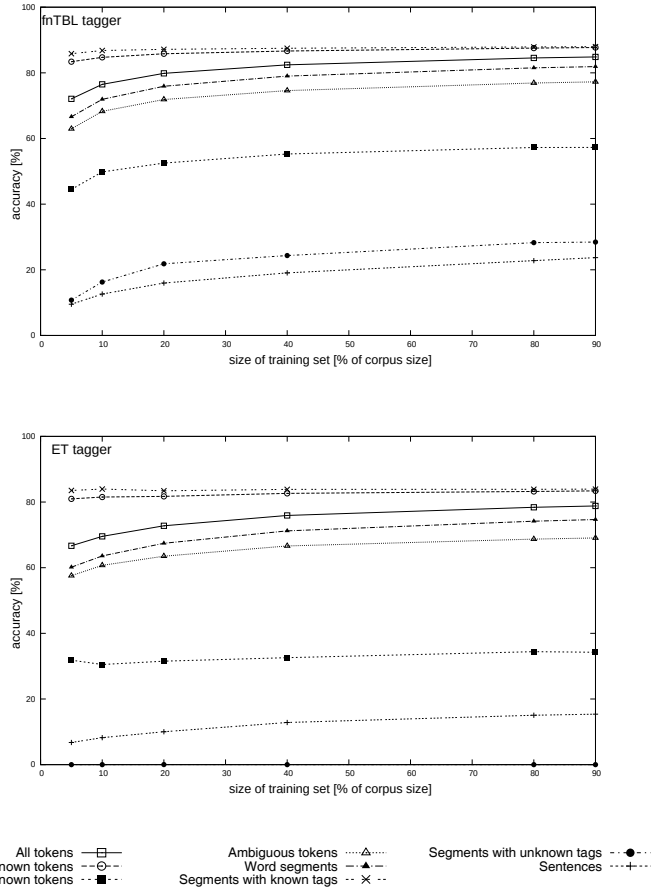


Fig. 5. Accuracy for complex tagset in relation to training corpus size (fnTBL and ET taggers)

- ET tagger is characterized by fast training, yet its accuracy is worse than that of the other taggers.
- The results are satisfactory if simple tagset information is required by NLP task, but all taggers achieve low results when considering the sentence accuracy. For the complex tagset the results would require improvement by a morphological analyser or more sophisticated algorithms (parallel combination of taggers, exponential method).

To summarize, there is no versatile tagger for morpho-syntactic tagging of Polish. If high accuracy is required, fnTBL tagger is most suitable. If fast training is crucial, TnT tagger would be advised.

	T3	TnT	MET	fnTBL	ET
	Simple tagset				
All tokens	96.24	96.20	96.49	96.50	94.45
Known tokens	96.97	96.98	97.23	97.46	97.13
Unknown tokens	89.10	88.64	89.42	87.14	68.51
Ambiguous tokens	89.59	89.59	90.46	91.32	90.11
Word segments	95.50	95.45	95.81	95.81	93.37
Word segments with known tags	96.93	96.94	97.24	97.52	97.12
Word segments with unknown tags	0	0	0	2.54	0.31
Unknown word segments	89.09	88.64	89.42	87.14	68.52
Sentences	61.56	61.51	63.58	63.32	51.22
	Complex tagset				
All tokens	–	84.27	82.39	84.86	78.80
Known tokens	–	86.84	85.07	87.71	83.41
Unknown tokens	–	59.42	56.45	57.29	34.25
Ambiguous tokens	–	75.45	72.20	77.26	69.07
Word segments	–	81.20	78.95	81.91	74.66
Word segments with known tags	–	88.35	86.08	87.98	83.94
Word segments with unknown tags	–	0	0	28.46	0
Unknown word segments	–	59.42	56.44	57.28	34.23
Sentences	–	22.11	19.97	23.70	15.37

Table 5. Tagging accuracy for taggers trained on 90 % of corpus, [%]

	T3	TnT	MET	fnTBL	ET
	Simple tagset				
number of iterations	74	20	43	30	54
Average training time [sec]	30	3	$42 \cdot 10^2$	$23 \cdot 10^2$	$2 \cdot 10^2$
Average tagging speed [tokens/sec]	$30 \cdot 10^2$	$220 \cdot 10^2$	$35 \cdot 10^2$	$9 \cdot 10^2$	$110 \cdot 10^2$
	Complex tagset				
number of iterations	–	20	5	19	62
Average training time [sec]	–	6	$977 \cdot 10^2$	$289 \cdot 10^2$	$4 \cdot 10^2$
Average tagging speed [tokens/sec]	–	$94 \cdot 10^2$	5	$16 \cdot 10^2$	$6 \cdot 10^2$

Table 6. Time performance of an instance of tagger trained on 90 % of corpus

Few work has been done in POS tagging of Polish. A rule-base tagger with three types of rules (acquired directly from training set, discovered by genetic algorithm and constructed manually) has been presented in [19] achieving 90.0 % accuracy. An architecture of the tagger has been developed in [20]. The tagging process is divided into three phases where grammatical class, number and gender, and case of the token are established subsequently. The tagger has achieved 93.11 % accuracy with a single classifier (C4.5 algorithm) and 93.53 % accuracy with multiclassifier approach.

If setting these results against ours, the following important factors have to be taken into account:

- Accuracy values are evaluated on different corpora, which are not of equal quality (corpus of frequency dictionary of the contemporary Polish language vs. IPI PAN Corpus).
- In our work we do not exploit morphological analyser, special dictionaries or manually created rules.
- Accuracy values are computed differently – compare Equation (3) to [20], where accuracy is computed only on part of speech, number and gender, and case. Other morphological categories are not taken into account. Additionally, in the latter work noun and gerund forms are treated as indistinguishable.

  Finally, we shortly compare our results with the case of Slovene [8] (cf. Table 1).

- For all algorithms our results are slightly lower than those reported for Slovene. This may arise from two reasons:
  - Polish is more difficult for morphosyntactic tagging than Slovene.
  - We used corpus not homogeneous thematically, being compilation from various sources, containing varied styles, e.g. news and plays (drama) while the Slovene corpus is uniform (the Orwell's novel).
- For Slovene the HMM tagger (TnT) proved to be the best choice. In the case of Polish a choice between the Brill (fnTBL), HMM (TnT) and even maximum entropy taggers (MET) should be considered (see previous conclusions).
- Surprisingly, according to Džeroski et al., maximum entropy tagger, as the only one, has the ability to predict correctly tags for word segments with unknown tags (in one third of cases). Our experience claims that this ability should be attributed to fnTBL tagger and in very residual degree to ET tagger.

9 SUMMARY

In the paper we evaluated five baseline taggers on the corpus of frequency dictionary of contemporary Polish. The obtained results have shown that they differ slightly in accuracy but significantly in performance speed. We are convinced that they could be useful in NLP support for many contemporary applications, like semantic web, knowledge-supported workflow construction or contract-based formation of virtual organisation, that constitute the challenges of modern information technologies.

In the future we plan to investigate more sophisticated techniques for morphosyntactic tagging of the Polish language.

Acknowledgments

Authors thank to Prof. K. Boryczko for his valuable remarks on batch queueing systems. ACC CYFRONET AGH is also acknowledged for the computing time.

REFERENCES

[1] Borin, L.: Enhancing Tagging Performance by Combining Knowledge Sources. Association Suédoise de Linguistique Appliquée (ASLA) Symposium Corpora in Research and Teaching, pp. 19–31, Växjö University, Sweden, November 11–12, 1999.

[2] Brill, E.: Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part of Speech Tagging. Computational Linguistics, Vol. 21, 1995, No. 4, pp. 543–565.

[3] Brill, E.: Unsupervised Learning of Disambiguation Rules for Part of Speech Tagging. In Proceedings of the Third Workshop on Very Large Corpora, Association for Computational Linguistics, pp. 1–13, Massachusetts, USA, 1995.

[4] Brants, T.: TnT – A Statistical Part-of-Speech Tagger. In Proceedings of the 6th Applied Natural Language Processing Conference (ANLP 2000), pp. 224–231, Seattle, Washington, USA, April 29–May 4, 2000.

[5] Byrne, E.: A Logical Framework for Identifying and Explaining Unexpected News. Computing and Informatics, Vol. 25, 2006, No. 2–3, pp. 153–171.

[6] Daelemans, W.—Zavrel, J.—Berck, P.—Gillis, S.: MBT: A Memory-Based Part of Speech Tagger-Generator. In Proceedings of the 4th Workshop on Very Large Corpora, pp. 14–27, Copenhagen, Denmark, 1996.

[7] Dębowski, Ł.: Tagging and Morphosyntactic Disambiguation. A Review of Methods and Software, IPI PAN Reports, No. 934, Warsaw, Poland, Nov. 2001 (in Polish).

[8] Džeroski, S.—Erjavec, T.—Zavrel, J.: Morphosyntactic Tagging of Slovene: Evaluating PoS Taggers and Tagsets. In the Proceedings of the 2nd International Conference on Language Resources and Evaluation (LREC 2000), pp. 1099–1104, Athens, Greece, 31 May–2 June 2000.

[9] Florian, R.—Ngai, G.: Fast Transformation-Based Learning Toolkit Manual. John Hopkins University, USA, 2001, `http://nlp.cs.jhu.edu/~rflorian/fntbl`.

[10] Gaweł, B.: Application of Genetic Programming Methods to Annotation of Words in Polish Text. M. Sc. thesis, Wrocław University of Technology, Wrocław, Poland, 2001 (in Polish).

[11] Giménez, J.—Màrquez, L.: SVMTool: A General POS Tagger Generator Based on Support Vector Machines. In Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC '04), pp. 43–46, Lisbon, Portugal, 2004.

[12] Hajič, J.: Morphological Tagging: Data vs. Dictionaries. In Proceedings of the First Conference on North American Chapter of the Association for Computational Linguistics, pp. 94–101, Seattle, Washington, USA, 2000.

[13] Hajič, J.—Hladká, B.: Probabilistic and Rule-Based Tagger of an Inflective Language – A Comparison. In Proceedings of the 5th Conference on Applied Natural Language Processing, pp. 111–118, Washington DC, USA, 1997.

[14] Hajič, J.—Hladká, B.: Tagging Inflective Languages: Prediction of Morphological Categories for a Rich, Structured Tagset. In Proceedings of 17th International Conference on Computational Linguistics, Vol. 1, pp. 483–490, Montréal, Quebec, Canada, 1998.

[15] Hajič, J.—Krbec, P.—Květoň, P.—Oliva, K.—Petkevič, V.: Serial Combination of Rules and Statistics: A Case Study in Czech Tagging. In Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL 2001), pp. 260–267, Toulouse, France, July 9–11, 2001.

[16] van Halteren, H.—Zavrel, J.—Daelemans, W.: Improving Data Driven Wordclass Tagging by System Combination. In Proceedings of the 36th Annual Meeting on Association for Computational Linguistics, Vol. 1, pp. 491–497, Montréal, Canada, 1998.

[17] Kuta, M.—Polak, S.—Palacz, B.—Miłoś, T.—Słota, R.—Kitowski, J.: TeToN – A Jena-Based Tool for Text-to-Ontology Approach. In Proceedings of the Cracow Grid Workshop '06, ACC Cyfronet AGH, Cracow, 2006 (to appear).

[18] Padró, L.: A Hybrid Environment for Syntax-Semantic Tagging. Ph. D. thesis, Dept. Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, February 1998.

[19] Piasecki, M.—Gaweł, B.: A Rule-Based Tagger for Polish Based on Genetic Algorithm. In Proceedings of Intelligent Information Processing and Web Mining Conference, pp. 247–255, Gdańsk, Poland, June 13–15, 2005.

[20] Piasecki, M.—Wardyński, A.: Multiclassifier Approach to Tagging of Polish. In Proceedings of First International Symposium on Advances in Artificial Intelligence and Applications, pp. 169–178, Wisła, Poland, November 6–10, 2006.

[21] Press, W.—Teutolsky, S.—Vetterling, W.—Flannery, B.: Numerical Recipes in C: The Art of Scientific Computing (second ed.). Cambridge University Press, 1992.

[22] Przepiórkowski, A.—Woliński, M.: The Unbearable Lightness of Tagging. A Case Study in Morphosyntactic Tagging of Polish. In Proceedings of the 4th International Workshop on Linguistically Interpreted Corpora (LINC '03, EACL 2003), pp. 109–116, Budapest, Hungary, April 12–17, 2003.

[23] Przepiórkowski, A.—Woliński, M.: A Flexemic Tagset for Polish. In Proceedings of the Workshop on Morphological Processing of Slavic Languages (EACL 2003), pp. 33–40, Budapest, Hungary, April 12–17, 2003.

[24] Rabiner, L. R.: A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Proceedings of the IEEE, Vol. 77, 1989, Issue 2, pp. 257–286.

[25] Ratnaparkhi, A.: A Maximum Entropy Model for Part-of-Speech Tagging. In Proceedings of the First Conference on Empirical Methods in Natural Language Processing, pp. 133–142, University of Pennsylvania, USA, 1996.

[26] Schmid, H.: Probabilistic Part-of-Speech Tagging Using Decision Trees. In Proceedings of International Conference on New Methods in Language Processing, pp. 44–49, Manchester, England, 1994.

[27] Schmid, H.: Part-of-Speech Tagging with Neural Networks. In Proceedings of the International Conference on Computational Linguistics, pp. 172–176, Kyoto, Japan, 1994.

[28] Schröder, I.: A Case Study in Part-of-Speech Tagging Using the ICOPOST Toolkit, Technical report FBI-HH-M-314/02, Department of Computer Science, University of Hamburg, Hamburg, Germany, 2002.

[29] Véronis, J.: Multext-East Resources. http://www.lpl.univ-aix.fr/projects/multext-east.

[30] Voutilainen, A.: A Syntax-Based Part of Speech Analyser. In Proceedings of the 7th Conference of the European Chapter of the Association for Computational Linguistics (EACL '95), pp. 157–164, Dublin, Ireland, 1995.

[31] Corpus of Frequency Dictionary of Contemporary Polish. http://www.mimuw.edu.pl/polszczyzna.

[32] IPI PAN Corpus Resources. http://korpus.pl.

[33] Baseline Information Extraction System. http://balie.sourceforge.net.

[34] Freeling Project. http://garraf.epsevg.upc.es/freeling/index.php.

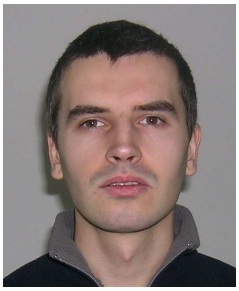
[35] LingPipe Project. http://www.alias-i.com/lingpipe.

[36] MAXENT Package. http://maxent.sourceforge.net.

[37] μ-TBL Tools for Transformation-Based Learning. http://www.ling.gu.se/~lager/mutbl.html.

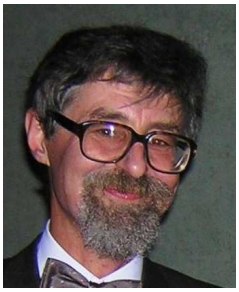
[38] Natural Language Toolkit. http://nltk.sourceforge.net/index.php.

[39] openNLP Project. http://opennlp.sourceforge.net.



Marcin Kuta received the M. Sc. degree in computer science in 2001 at the AGH University of Science and Technology in Kraków (Poland). Since 2003 he works at the Institute of Computer Science of the AGH University of Science and Technology, where he teaches compiler techniques and formal languages. His research interests comprise natural language processing, ontology usage, question answering systems and knowledge engineering.

Paweł Chrzaszcz is currently a masters student of computer science at the AGH University of Science and Technology in Kraków (Poland). His research interests are in natural language processing, computational complexity problems, parallel and distributed computing environments. At present he works as a developer of web-based applications.



Jacek Kitowski professor of computer science, graduated in 1973 at the Electrical Department of the AGH University of Science and Technology in Kraków (Poland). He obtained Ph. D. in 1978 and D. Sc. (habilitation) in 1991 in computer science from the same University. He is the Head of the Computer Systems Group at the Institute of Computer Science of the AGH University of Science and Technology in Cracow, Poland. Full professor since 2001. He also works for the Academic Computer Centre CYFRONET-AGH, where he is responsible for developing high-performance systems. He is the author or co-author of about 200 scientific papers. His topics of interest include, but are not limited to, large-scale computations, multiprocessor architectures, high availability systems, network computing, Grid services and Grid storage systems, knowledge engineering. He participates in program committees of many conferences, and has been involved in many national and international projects, most notably in EU IST CrossGrid, EU IST Pellucid and EU IST K-WfGrid projects. At present he participates in EU GREDIA and EU `int.eu.grid` projects.