Computing and Informatics, Vol. 29, 2010, 203–225

APPLICATION OF WEIGHTED VOTING TAGGERS TO LANGUAGES DESCRIBED WITH LARGE TAGSETS

Marcin Kuta, Wojciech Wójcik, Michał Wrzeszcz
Jacek Kitowski

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

Manuscript received 7 July 2008; revised 2 July 2009
Communicated by Milan Rusko

Abstract. The paper presents baseline and complex part-of-speech taggers applied to the modified corpus of Frequency Dictionary of Contemporary Polish, annotated with a large tagset. First, the paper examines accuracy of 6 baseline part-of-speech taggers. The main part of the work presents simple weighted voting and complex voting taggers. Special attention is paid to lexical voting methods and issues of ties and fallbacks. TagPair and WPDV voting methods achieve the top accuracy among all considered methods. Error reduction 10.8 % with respect to the best baseline tagger for the large tagset is comparable with other author's results for small tagsets.

Keywords: Part-of-speech tagging, combination tagger, weighted probability distribution voting tagger, TagPair tagger

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

1 INTRODUCTION

Part of speech (POS) tagging is a key topic for areas like natural language processing (NLP), information extraction, and machine translation, being the first step in a long chain of complicated text transformations. POS tagging is considered to be well understood and almost resolved problem for languages that need few tags for their depiction. Application of POS tagging to languages described with large tagsets is far more difficult because of

1. apparently fewer algorithms available and tagging tools usable,
2. lower accuracy values reached by standard algorithms,
3. memory requirements often exceeding resources of ordinary machines,
4. time demands considerably higher than in applications to small tagsets,

what may be unacceptable from the point of view of a user [15].

The possible remedy for insufficient accuracy is to combine a few baseline algorithms into one system attaining higher accuracy. The paper examines accuracy of 6 baseline algorithms applied to morphosyntactic tagging of Polish. Next, simple weighted voting taggers and complex voting methods are investigated. The paper pays attention to lexical voting methods, which have not been explicitly considered before. Up to our knowledge it is the first so exhaustive arrangement of results of baseline and combined taggers for the Polish language.

We investigate behaviour of weighted voting algorithms proposed in literature, and our lexical methods when applied to Polish – a language described with large tagsets. Polish and all Slavic languages belong to the family of inflective languages, while English or French are analytic languages. A small tagset is sufficient for description of words of analytic languages. Due to inflection, significantly higher number of tags is required for Slavic languages. Tagging with large tagsets is much more difficult and achieved tagging accuracy is smaller than in case of tagging with small tagsets. The focus of the paper is to show that in spite of these difficulties collective methods working on a large tagset achieve error reduction rate similar to collective methods working on a small tagset.

The paper takes also deep insight into ties and fallbacks effects, negligible when combined methods are employed to languages with small tagsets, but present in high degree in case of applications to large tagsets.

The paper operates on the following concepts. A morphosyntactic tag is a set of attributes, attributes describe morphological categories. The tagset is referred to as a complex tagset (due to its large size and presence of many morphological categories). A simple tagset is also considered, where POS is present but other, more specific morphological categories are ignored. A token is the smallest entity being subject of tagging, corresponding approximately to the general notion of a word, but certain shorter segments may be also recognised as tokens. A word segment is a token containing at least one letter or digit.

The rest of the work is organised as follows. Section 2 gives overview of baseline tagging algorithms with special focus on methods investigated in the paper. Voting methods are presented in Section 3. Details of the conducted experiments are described in Section 4. Section 5 provides and discusses results of the experiments. Conclusions close the paper. The accuracy of cited methods for English is reported on the Penn Treebank.

2 OVERVIEW OF BASELINE TAGGING ALGORITHMS

Up to now many approaches to the issue of POS tagging have been proposed. They can be divided between statistical and rule based algorithms.

The rule based methods take into account a wide context, what is desirable in the case of languages containing long-distance 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 [7].

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 advantage of the exploited method is a possibility of supplying and modifying rules by linguists. Beside the training and test corpora the method requires an additional patch corpus.

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

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.

Statistical algorithms map a sequence of tokens into a sequence of tags with a probability model, which describes occurrence of the most probable sequence of tags for a given sequence of tokens. Generative methods model joint probability (e.g. hidden Markov model) whereas discriminative ones model conditional probability (e.g. conditional random fields, support vector machines, boosting).

Matrix of collocational probabilities indicates the relative likelihood of co-occurrence of all pairs of tags. The matrix, innovation of the CLAWS algorithm [19], was the first step towards statistical algorithms in POS tagging area. The CLAWS algorithm has, however, exponential time complexity and is of historical significance now.

Hidden Markov Model (HMM) taggers are most suitable for POS tagging of the analytic languages due to their weak inflection. The most probable sequence of tags is computed with the Viterbi algorithm and the learning of tagger parameters done with the Baum-Welch algorithm. A gentle introduction to the HMM methodology is given in [23].

Maximum entropy taggers aim at maximizing 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 [24]. Applying the maximum entropy method to inflecting languages comes with difficulties because of usually large tagset sizes and intra dependencies occurring in such languages.

Hajič method. The Hajič tagger disambiguates results provided by independent morphological analyser. The Hajič method originates from the maximum entropy approach. In order to avoid computational complexity of the maximum entropy approach a naive Bayesian assumption is made. In exchange, a significantly higher number of features can be examined. The algorithm achieves satisfactory results for inflecting language (like Czech [10, 11]).

Dependency networks use explicitly context of both preceding and following tag to determine a tag of the current token. It stays in contrast to traditional models, where unidirectional (usually left to right) approach is taken. The drawback of the method is that a found sequence of tags is not guaranteed to be the maximum likelihood one. System presented in [29] has 97.24 % accuracy on the Penn Treebank WSJ.

Conditional Random Fields (CRFs) [18] are of recently growing interest example of an undirected graphical model that avoids label bias. The form of an applied potential function is inspired by the maximum entropy approach. The CRF trained on 50 % of the Penn Treebank is reported to achieve 95.73 % accuracy and claimed to outperform the HMM [18].

A POS tagging task may be understood as a pattern recognition problem, thus algorithms from machine learning area are applicable, too.

Support Vector Machines(SVM). The SVM is a supervised machine learning algorithm for binary classification, but N-class problems are converted to binary ones without much trouble.

Boosting. The AdaBoost (Adaptive Boosting) is a metaalgorithm for constructing a 'strong' classifier, $H(x) = \mathrm{sgn}(\sum_t \alpha_t h_t(x))$, as a linear combination of simple 'weak' classifiers $h_t(x)$. The MI version of AdaBoost, applied to POS tagging of the Penn Treebank corpus achieves 96.72 % accuracy [1].

Decision trees taggers estimate transition probabilities with decision trees, which are built with the help of concepts from information theory like information gain or gain ratio. The representatives of tree building algorithms are CART, ID3, C4.5 and C5.0 – the newer and commercial version of the C4.5 algorithm. The TreeTagger [26], making use of ID3, is reported with 96.36 % accuracy on the Penn Treebank.

Neural networks as a POS tagging tool have been subject of several researches. Reported results for the English language are comparable with the HMM approach. A slow training process may be the disadvantage of the method. Among others, the following architectures have been explored: multilayer network with the backpropagation training algorithm (96.22 % accuracy) [27], network of linear separators with the Winnow update algorithm (97.20 % accuracy) [25], voted perceptron based training (97.07 % accuracy) [6].

Genetic programming. A tagger making use of genetic programming has been proposed in the context of building a tagger of Polish [21].

2.1 Exploited Algorithms

In the paper we first evaluate a selected set of algorithms, discussed below in more detail. These algorithms are 'building blocks' to create and investigate complex methods. The results of baseline algorithms are also important in their own right if confined to usaged of the (best) base tagger.

Hidden Markov Model. Given a 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 a word w_i given a tag t_i occurred and $p(t_i|t_{i-1}, \ldots, t_{i-N})$ is the conditional probability of occurrence of the tag t_i given a tag sequence $t_{i-1}, \ldots, t_{i-N}$ previously occurred. The Markov model of N^{th} order is called the $(N+1)$-gram model.

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

$$p(t_i, c) = \frac{1}{Z} \exp\left(\sum_{j} \lambda_j f_j(t_i, c)\right), \tag{2}$$

where $p(t_i, c)$ stands for joint distribution of tags and contexts and $1/Z$ is a normalisation factor in order that $p(\cdot, \cdot)$ forms the probability function.

The aim of the training procedure is to select reliable features, f_j, and to compute their weights, λ_j.

Memory-based learning. These algorithms are distinguished by definition of similarity of examples, 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 the distance between c and c_k is minimal.

The most important similarity metrics used are: overlap metric, modified value difference metric, Jeffrey divergence metric, and dot-product metric. The features are weighted with such metrics as information gain or chi-squared. 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 a 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.

Support vector machines. The SVM maps input vectors from $\mathbb{R}^n$ to a higher dimensional space where a linear hyperplane is constructed, which separates positive examples from negative ones with the maximal margin. The mapping is done with kernel functions, of which linear, polynomial, radial basis (RBFs) and sigmoid functions are the most used. The SVM approach to POS tagging has been addressed in [9] with accuracy of 97.16 % for English and 96.89 % for Spanish achieved.

Tree tagger. Tree tagger is based on Markov models and exploits additionally decision trees to improve its work [26]. The tag sequences are computed with the slightly modified Equation (1). To predict probability parameters the maximum likelihood estimation is used. In order to solve data sparseness problem, TreeTagger applies decision tree (thus reducing of the number of contextual parameters) instead of smoothing probabilities. The tree is built from a training set with the ID3 algorithm and weighted information gain.

Table 1 presents evaluated baseline algorithms and their implementations used within the research.

Tagger	Algorithm	Tagger name	Abbreviation
T1	HMM	TnT [5]	TnT
T2	Maximum entropy	MXPost [24]	MXP
T3	Transformation based	fnTBL [8]	fnTBL
T4	Memory based	MBT [7]	MBT
T5	SVM	SVMTool [9]	SVM
T6	Decision trees	TreeTagger [26]	TrTg

Table 1. Taggers used in experiments with their abbreviations employed in the paper

3 COLLECTIVE METHODS

Instead of inventing new baseline methods, which would attain higher accuracy due to larger internal complexity there is an alternative way of proceeding. As different methods represent different approaches and formalisms, they should manifest diversity and scattering (measured with complementarity, *comp*, values) in committed errors. Consequently, selection of the correct tag among those proposed by all the baseline taggers improves the whole tagging process, as we suppose that majority of taggers should be correct at the same time.

Independent baseline taggers are combined as component taggers in parallel (simple voting methods) or sequential manner (stacked methods) to form one complex tagging system. Most of voting and stacked methods were contributed and then evaluated on English by Brill [4] and van Halteren et al. [13, 14].

3.1 Simple Weighted Voting Methods

Each base tagger proposes an output tag or tags, according to its internal algorithm. Each tag is suggested with certain weight from range $[0, 1]$, values of weights are fixed within training process of a voting tagger. The tag which receives the highest number of votes is selected as an output tag of a whole combined system. Voting taggers are distinguished by a way vote strength of their baseline taggers is calculated.

Let us introduce the necessary measures. Assuming the reference test corpus contains n tokens $w_1, \ldots, w_n$, a token w_i is annotated in the corpus with a tag t_i $(1 \leq i \leq n)$, a tagger A guesses for a token w_i a tag t_i^A, the accuracy of the tagger A is defined as follows:

$$\text{accuracy} = \frac{\#\text{correctly tagged tokens}}{\#\text{all tokens}} = \frac{\sum_{i=1}^{n} \delta(t_i^A, t_i)}{n}, \tag{3}$$

where δ is the Kronecker delta function.

If a tagger B guesses a tag t_i^B, the complementarity of the tagger B to the tagger A, *comp*$(B|A)$, is given by [4]:

$$\begin{aligned} \text{comp}(B|A) &\overset{\text{df}}{=} 1 - \frac{\#\text{common errors of taggers } A \text{ and } B}{\#\text{errors of tagger } A} \\ &= \frac{\sum_{i=1}^{n}(1-\delta(t_i^A, t_i)) \cdot \delta(t_i^B, t_i)}{\sum_{i=1}^{n}(1-\delta(t_i^A, t_i))}. \end{aligned} \tag{4}$$

Given a tag X, the tag precision and tag recall of the tagger A on the tag X can be calculated according to formulas:

$$\begin{aligned} \text{prec}_X &\overset{\text{df}}{=} \frac{\#\text{tokens tagged and annotated with } X}{\#\text{tokens tagged with } X} \\ &= \frac{\sum_{i=1}^{n} \delta(t_i, X) \cdot \delta(t_i^A, X)}{\sum_{i=1}^{n} \delta(t_i^A, X)}, \end{aligned} \tag{5}$$

$$\begin{aligned} \text{recall}_X &\overset{\text{df}}{=} \frac{\#\text{tokens tagged and annotated with } X}{\#\text{tokens annotated with } X} \\ &= \frac{\sum_{i=1}^{n} \delta(t_i, X) \cdot \delta(t_i^A, X)}{\sum_{i=1}^{n} \delta(t_i, X)}. \end{aligned} \tag{6}$$

We propose below some new measures and on that basis introduce new simple voting tagging methods.

Given a word form W, the lexical accuracy on the word form W, lexical tag precision, and lexical tag recall on the word form W and the tag X, are defined for the tagger A in a following way:

$$\begin{aligned} \mathit{lexAcc}_W &\overset{\mathrm{df}}{=} \frac{\#\text{tokens } W \text{ correctly tagged}}{\#\text{tokens } W} \\ &= \frac{\sum_{i=1}^{n} \delta(t_i, t_i^A) \cdot \delta(w_i, W)}{\sum_{i=1}^{n} \delta(w_i, W)}, \end{aligned} \tag{7}$$

$$\begin{aligned} \mathit{lexPrec}_{W,X} &\overset{\mathrm{df}}{=} \frac{\#\text{tokens } W \text{ tagged and annotated with } X}{\#\text{tokens } W \text{ tagged with } X} \\ &= \frac{\sum_{i=1}^{n} \delta(t_i, X) \cdot \delta(t_i^A, X) \cdot \delta(w_i, W)}{\sum_{i=1}^{n} \delta(t_i^A, X) \cdot \delta(w_i, W)}, \end{aligned} \tag{8}$$

$$\begin{aligned} \mathit{lexRecall}_{W,X} &\overset{\mathrm{df}}{=} \frac{\#\text{tokens } W \text{ tagged and annotated with } X}{\#\text{tokens } W \text{ annotated with } X} \\ &= \frac{\sum_{i=1}^{n} \delta(t_i, X) \cdot \delta(t_i^A, X) \cdot \delta(w_i, W)}{\sum_{i=1}^{n} \delta(t_i, X) \cdot \delta(w_i, W)}. \end{aligned} \tag{9}$$

Simple weighted voting methods and measures for computing their vote strength are summarised in Table 2. The simplest Majority method assigns always one vote to each component tagger. Vote strength may be calculated more precisely and depends on: overall accuracy (TotalPrecision method), or proposed tag (TagPrecision and PrecisionRecall methods). Within PrecisionRecall method each tagger supports also tags suggested by remaining taggers.

We extended the idea of precise calculation of vote strength and proposed additional methods, where relevant measures depend also on the word form of the token being currently tagged. Vote strength of these new 'lexical' methods depends on: tagged word form (LexAccuracy), proposed tag and tagged word form (LexPrecision, LexPrecisionRecall). The LexPrecisionRecall method uses also lexical equivalent of the tag recall measure.

3.2 Complex Weighted Voting Methods

Complex weighted voting methods represent a family of stacked methods. Stacked methods are second order methods – output of baseline taggers serves as a training material for a second order tagger. The specific feature of stacked methods, compared to simple voting methods, is that they can choose a tag proposed by minority of baseline taggers or even a tag from outside the set of tags proposed by baseline taggers. We examine further 3 representatives of complex weighted voting methods: van Halteren's TagPair, Weighted Probability Distribution Voting (WPDV) and our modification of TagPair – LexPair.

Voting method	Tag X	Tag Y
Majority	1	0
TotalPrecision	*accuracy*	0
TagPrecision	$prec_X$	0
PrecisionRecall	$prec_X$	$1 - recall_Y$
LexAccuracy	$lexAcc_W$	0
LexPrecision	$lexPrec_{W,X}$	0
LexPrecisionRecall	$lexPrec_{W,X}$	$1 - lexRecall_{W,Y}$

Table 2. Vote strength of a component tagger in various simple voting methods; W stands for a currently tagged token, X stands for a tag proposed by a component tagger, Y ($Y \neq X$) represents each tag proposed by opposition

TagPair. Let $\mathcal{T} = \{\mathrm{T}1, \mathrm{T}2, \ldots, \mathrm{T}k\}$ be the set of baseline taggers, $|\mathcal{T}| = k$. Given k baseline taggers, $\binom{k}{2}$ combination taggers (pairs) are created. A combination tagger C_{ij} ($1 \leq i < j \leq k$) corresponding to a pair of baseline taggers ($\mathrm{T}i$,$\mathrm{T}j$) votes each tag tag_T, for which the probability $P(tag_\mathrm{T} \mid tag_i, tag_j)$ is nonzero, where tag_i, tag_j stand for tags proposed by baseline taggers $\mathrm{T}i$, $\mathrm{T}j$. This probability specifies at the same time the weight of vote of the tagger C_{ij} for the tag tag_T and is estimated from the tuning corpus. If tags tag_i, tag_j have never been proposed simultaneously by taggers $\mathrm{T}i$ and $\mathrm{T}j$ in the tuning data, the probability P cannot be computed for any tag_T (data sparseness problem). In such case a fallback strategy is undertaken – smoothed estimate of P, i.e., the value $\frac{1}{2}(P(tag_\mathrm{T} \mid tag_i) + P(tag_\mathrm{T} \mid tag_j))$, constitutes a vote of C_{ij}.

The vote of the tagger TagPair for a tag tag_T is a sum of votes of components C_{ij} for that tag. The TagPair method is formalized in Algorithm 1.

LexPair. Similarly to weighted voting methods, we extend the idea of TagPair and propose the LexPair method. LexPair defines a vote of a combination tagger C_{ij} for a tag tag_T to be equal $P(tag_\mathrm{T} \mid tag_i, tag_j, tok)$, provided the current token is tok and P is nonzero.

In analogical way, the value $\frac{1}{2}(P(tag_\mathrm{T} \mid tag_i, tok) + P(tag_\mathrm{T} \mid tag_j, tok))$ defines a vote strength for a tag tag_T if a pair tag_i, tag_j has never been encountered in the tuning data as a simultaneous proposition of taggers $\mathrm{T}i$ and $\mathrm{T}j$ for a token tok. The LexPair method is defined formally in Algorithm 2.

Weighted Probability Distribution Voting (WPDV). This method is van Halteren's extension of TagPair. The TagPair method takes combination taggers being pairs of taggers, i.e., elements C_{ij} such that $C_{ij} \in 2^\mathcal{T}$ and $|C_{ij}| = 2$. WPDV removes cardinality restriction and admits any element from $2^\mathcal{F}$ as a combination tagger, where $\mathcal{F}$ denotes a set of features, $\mathcal{F} = \{f_1 = v_1, \ldots, f_m = v_m\}$, f_i – feature, v_i – value. The set $\mathcal{F}$ is a generalization of $\mathcal{T}$, but for our aims it is sufficient to assume that $\mathcal{F} = \mathcal{T}$. A component tagger of N-th order, $C_{i_1 \ldots i_N} = (\mathrm{T}i_1, \ldots, \mathrm{T}i_N)$ votes tag tag_T with value $w_N \cdot P(tag_\mathrm{T} \mid tag_{i_1}, \ldots, tag_{i_N})$, where w_N

Algorithm 1 The TagPair method

```
foreach $tag_T \in$ tagset do
  $vote[tag_T] := 0$
end for
foreach pair of taggers $C$ do
  foreach $tag_T \in$ tagset do
    if $P(tag_T \mid tag_{i_1}, tag_{i_2})$ is defined then
      $vote[tag_T] \mathrel{+}= P(tag_T \mid tag_{i_1}, tag_{i_2})$
    else if $P(tag_T \mid tag_{i_1})$ is defined and $P(tag_T \mid tag_{i_2})$ is defined then
      $vote[tag_T] \mathrel{+}= \frac{1}{2}(P(tag_T \mid tag_{i_1}) + P(tag_T \mid tag_{i_2}))$
    else
      $choosedTag := Majority$
      goto $finish$
    end if
  end for
end for
$choosedTag := \arg\max_{tag_T \in \text{tagset}} vote[tag_T]$
$finish$: return $choosedTag$
```

Used variables have the following meaning:

C – a component tagger composed from baseline taggers $\mathrm{T}i_1, \mathrm{T}i_2$,

tag_{i_1}, tag_{i_2} – tags proposed by baseline taggers $\mathrm{T}i_1, \mathrm{T}i_2$,

Majority – function of majority voting among tags suggested by base taggers with random determination of ties.

denotes a scaling factor. Choosing good values of weights w_N is not a straightforward task. One possibility is to assume $w_N = N!$, in this way high order component taggers are prevented from being dominated by low order elements when summing votes [12, 14].

To prevent exponential explosion of the number of component taggers with the growing size of $\mathcal{T}$, only components consisting of maximally 4 taggers were considered.

In contrast to TagPair, WPDV avoided fallbacks to simpler models. Instead, cutoff strategy was applied. If necessary probability could not be estimated, vote of relevant component tagger was set to 0. Another applied improvement was a frequency threshold (set to 2 in our work), which meant that a feature had to occur at least 2 times in the tuning data to be taken into vote computations [12].

A formal scheme for choosing a tag for a token *tok* with WPDV is given in Algorithm 3.

Let us consider the computational complexity of the WPDV tagger. The training phase is performed only once. Its complexity equals $\mathcal{O}(s|\mathcal{T}|)$, where s is the

Algorithm 2 The LexPair method

```
foreach $tag_T \in$ tagset do
    $vote[tag_T] := 0$
end for
foreach pair of taggers $P$ do
    foreach $tag_T \in$ tagset do
        if $P(tag_T \mid tag_{i_1}, tag_{i_2}, tok)$ is defined then
            $vote[tag_T]$+=$P(tag_T \mid tag_{i_1}, tag_{i_2}, tok)$
        else if $P(tag_T \mid tag_{i_1}, tok)$ is defined and $P(tag_T \mid tag_{i_2}, tok)$ is defined then
            $vote[tag_T]$+=$\frac{1}{2}(P(tag_T \mid tag_{i_1}, tok) + P(tag_T \mid tag_{i_2}, tok))$
        else
            $choosedTag := Majority$
            goto $finish$
        end if
    end for
end for
$choosedTag := \underset{tag_T \in \text{tagset}}{\arg\max}\ vote[tag_T]$
$finish$: return $choosedTag$
```

number of tokens in the training set, as in this phase we gather data about all component taggers in one place. The computational complexity of tagging one token equals $\mathcal{O}(s2^{|\mathcal{T}|})$. If we constrain to component taggers belonging to a subset $D, |D| = d$, as we do in the experiments, then the complexity reduces to $\mathcal{O}(s2^{|D|}) = \mathcal{O}(s2^d)$.

4 EXPERIMENTS SETUP

For experiments we applied the modified corpus of Frequency Dictionary of Contemporary Polish [16], which is an improved version of the corpus [31], and can be obtained from [32]. The m-FDCP corpus is annotated with the modified version of the IPI PAN tagset [22], containing 9 morphological categories. Five themes: scientific texts, news, essays, fiction and plays, are represented in the corpus in balanced manner.

The main parameters of the m-FDCP corpus, training and test sets are given in Table 3 after [16].

At first the whole corpus was divided into training and test corpora in 90 %/10 % ratio and baseline taggers T1, ..., T6 have been trained on the training set and applied to tagging the test set. Next the sort of 9-fold validation procedure was additionally applied to the training corpus to create tuning corpora (one tuning corpus for each of 6 baseline taggers) in the following way. The training corpus was split into 9 equal parts. Each part, standing for 10 % of the whole corpus, was tagged by each baseline tagger, trained on the remaining 8 parts. Finally 9 parts

Algorithm 3 The WPDV method

```
foreach $tag_T \in$ tagset do
  $vote[tag_T] := 0$
end for
foreach combination tagger $C \in D$ do
  foreach $tag_T \in$ tagset do
    if $P(tag_T \mid tag_{i_1}, \ldots, tag_{i_k})$ is defined and $freq(C) \geq cutoff$ then
      $vote[tag_T]{+}{=}w(C) \cdot P(tag_T \mid tag_{i_1}, \ldots, tag_{i_k})$
    end if
  end for
end for
if $freq(C) < cutoff$ for all combination taggers $C \in D$ then
  $choosedTag := Majority$
  goto $finish$
end if
$choosedTag := \arg\max_{tag_T \in \text{tagset}} vote[tag_T]$
$finish$: return $choosedTag$
```

Variables have the following meaning:

C – a component tagger composed from baseline taggers $\mathrm{T}i_1, \ldots, \mathrm{T}i_k$,

D – a subset of $2^{\mathcal{T}}$ ($D \subseteq 2^{\mathcal{T}}$) chosen to moderate exponential explosion of the number of component taggers. A component tagger belonging to space D consists of maximally 4 baseline taggers.

$tag_{i_1}, \ldots, tag_{i_k}$ – tags proposed by baseline taggers $\mathrm{T}i_1, \ldots, \mathrm{T}i_k$,

$w(C)$ – weight assigned to a tagger C which should take into account amount of information contributed by the tagger C. In our experiments $w(C) = |C|!$.

$freq(C)$ – the number of occurrences of a combination $(tag_{i_1}, \ldots, tag_{i_k})$ in the tuning set,

cutoff – parameter of algorithm, default value set to 2 in the experiments,

Majority – function of majority voting among tags suggested by base taggers with random determination of ties.

---

tagged by a baseline tagger were merged to form its tuning corpus, all in all we got 6 tuning corpora.

The tuning corpora and training corpus are identical if only tokens are taken into account and tags are ignored. The tuning corpora served as training sets for complex taggers.

For experiments with baseline taggers the SGI Altix 3700 SMP supercomputer, located at the ACC Cyfronet AGH-UST, and equipped with 128 1.5 GHz Intel Itanium 2 processors, 256 GB RAM, was utilized. Collective methods were tested under

	Training 90 %	Test 10 %	Full 100 %
tokens	592 729	65 927	658 656
word segments	496 907	55 139	552 046
sentences	36 601	4 211	40 812
different tokens	87 097	19 557	92 872
Simple tagset			
tagset size	30	30	30
ambiguous tokens, %	26.15	26.19	26.16
mean token ambiguity	1.44	1.43	1.44
Complex tagset			
tagset size	1 191	724	1 243
ambiguous tokens, %	47.76	47.65	47.74
mean token ambiguity	3.12	3.12	3.12

Table 3. Main parameters of the m-FDCP corpus

a personal computer with 2.0 GHz Intel Pentium M 760 processor, 1 GB RAM. Java heap size was set to 128 MB.

5 EXPERIMENTAL RESULTS AND IMPLEMENTATION ISSUES

All tables and data below refer to the test set containing 65 927 tokens (cf. Table 3).

5.1 Baseline Taggers

We used 6 flexible implementations of main tagging algorithms as baseline taggers, which can be adopted to virtually any language. Table 4 presents accuracy of baseline taggers and is an extension of results given in [16, 17] for 4 and 5 taggers.

5.2 Voting Taggers

Each voting method has been investigated in three setups:

1. as composed from all 6 base taggers,
2. composed from 5 best baseline taggers, (all components except the MBT tagger for the simple tagset and all components except the SVM tagger for the complex tagset),
3. composed from 4 best baseline taggers (all components except the MBT and TrTg taggers for the simple tagset and all components except the SVM and TrTg taggers for the complex tagset).

If the simple tagset is considered, the presented setup stays in contradiction with Table 4, as accuracy values for selection the best component taggers came

	TnT	MXP	fnTBL	MBT	SVM	TrTg
	Simple tagset					
All tokens	96.20	96.30	96.51	95.74	96.74	95.73
Known tokens	96.98	97.01	97.51	97.10	97.52	96.82
Unknown tokens	88.65	89.43	86.89	82.60	89.14	85.19
Ambiguous tokens	89.50	91.09	91.36	89.94	91.36	88.94
Word segments	95.46	95.57	95.83	94.91	96.10	94.89
Word segments with known tags	96.94	96.79	97.55	97.08	97.60	96.73
Word segments with unknown tags	0.00	28.21	3.85	0.00	0.00	0.00
Unknown word segments	88.65	89.43	86.90	82.60	89.15	85.20
Sentences	61.48	62.15	63.71	58.54	65.16	58.35
	Complex tagset					
All tokens	86.33	85.00	86.79	82.31	73.54	82.16
Known tokens	88.97	87.53	89.76	85.75	81.12	85.98
Unknown tokens	60.86	60.55	58.09	49.06	0.23	45.18
Ambiguous tokens	78.66	78.71	80.34	72.48	63.44	72.89
Word segments	83.66	82.07	84.21	78.85	68.36	78.66
Word segments with known tags	90.73	87.44	90.27	86.58	80.69	86.86
Word segments with unknown tags	0.00	29.84	30.50	0.66	0.00	0.93
Unknown word segments	60.86	60.55	58.09	49.05	0.23	45.18
Sentences	28.95	26.88	29.87	22.51	12.04	21.25

Table 4. Accuracy of baseline taggers trained on 90 % of the m-FDCP corpus, with respect to different categories [%]

from evaluation on the tuning corpora rather than test set (to avoid learn on a test set effect). To keep clarity in the shown results (Table 5, see [17] for preliminary results), the accuracy of all voting methods has been presented with '4 best' setup only, except the Majority method, which has been reported with all 3 setups.

Theoretical accuracy, defined as percentage of tokens for which at least one base tagger proposed a correct tag, determines the upper limit of accuracy that simple weighted voting methods may attain. The limit of theoretical accuracy may be, however, exceeded (at least in theory) by combined voting methods. The values of theoretical accuracy stay in accordance with Table 6, which presents votes distribution with the Majority method.

Not every combined tagger outperforms the best single tagger, what is the case of the Majority tagger composed of all 6 base taggers for the complex tagset. Decreasing the number of components may increase accuracy of a voting method, a phenomenon distinctly observed for Majority. Our experiments showed also that substituting TrTg with SVM (that is with the tagger inferior to TrTg if limited to the complex tagset), leaving other components unchanged, may lead to improved performance of combined taggers.

An attempt to phrase the possibility of tagging improvement by combining several taggers is the complementarity measure (Table 7, being an extension of results

Method	Simple tagset		Complex tagset	
	Accuracy	Δ_{Err}	Accuracy	Δ_{Err}
Theoretical	98.58		94.00	
Majority	96.86 ± 0.01	3.7	86.14 ± 0.04	−4.9
5-Best Majority	96.87 ± 0.01	4.0	86.87 ± 0.03	0.6
4-Best Majority	96.89 ± 0.02	4.6	87.19 ± 0.04	3.0
TotalPrecision	96.96	6.8	88.03	9.4
TagPrecision	96.91	5.2	87.41	4.7
PrecisionRecall	96.95	6.4	87.21	3.2
LexAccuracy	96.97	7.1	87.89	8.3
LexPrecision	96.82	2.5	87.15	2.7
LexPrecisionRecall	96.82	2.5	86.99	1.5
TagPair	97.02	8.6	88.22	10.8
LexPair	96.84 ± 0.01	3.1	86.99 ± 0.03	1.5
WPDV	97.02	8.6	88.20	10.7

Table 5. Accuracy (with its standard deviation) and error rate reduction, Δ_{Err} (in relation to the best base tagger) of voting taggers [%]

from [17]). The higher the complementarity, the bigger hope the combined system performs better than its components alone. It remains, however, unclear how many and which taggers should be selected on the basis of the complementarity measure to attain the best results.

Voting scheme	Simple tagset	Complex tagset
All taggers agree and are correct	92.30	67.04
A majority is correct	4.22	17.73
Correct tag present but tied	0.69	2.92
A minority is correct	1.38	6.31
The taggers vary but all are wrong	0.26	4.17
All taggers agree but are wrong	1.16	1.83

Table 6. Voting schemes of the Majority tagger [% of the test set, for which given voting scheme occurred]

Among simple weighted voting taggers the highest accuracy is achieved by the LexAccuracy tagger for the simple tagset and by the TotalPrecision tagger for the complex tagset.

The TagPair and WPDV complex voting taggers outperformed simple voting taggers reaching peak accuracy values: for the simple tagset 97.02 % accuracy and 8.6 % error rate reduction (WPDV), for the complex tagset 88.22 % accuracy and 10.8 % error rate reduction (TagPair). At this stage it cannot be stated in a definite manner which tagger is the best, as accuracy differences between two taggers were modest. In turn, arrangement with 5 or 6 base taggers revealed more prominent supremacy of WPDV.

B \ A	TnT	MXP	fnTBL	MBT	SVM	TrTg
TnT	–	35.72	31.29	39.21	25.74	27.28
MXP	37.32	–	36.72	46.58	30.30	44.16
fnTBL	36.84	40.35	–	41.03	25.65	44.62
MBT	31.80	38.55	28.03	–	23.79	39.04
SVM	36.16	38.55	30.47	41.60	–	45.01
TrTg	18.22	35.60	32.25	38.89	28.07	–

a) simple tagset

B \ A	TnT	MXP	fnTBL	MBT	SVM	TrTg
TnT	–	39.13	32.28	39.25	55.39	37.22
MXP	33.20	–	33.25	43.19	56.42	42.27
fnTBL	34.56	41.22	–	41.07	54.05	42.39
MBT	21.39	33.01	21.09	–	39.96	28.66
SVM	13.62	23.12	7.94	10.17	–	10.73
TrTg	18.04	31.32	22.17	28.03	39.80	–

b) complex tagset

Table 7. Complementarity, *comp*$(B|A)$, of taggers trained on 90 % of the m-FDCP corpus [%]

The LexPair, another member of complex voting taggers family, did not turn out useful with its low accuracy (the lowest accuracy for the simple tagset) and complicated architecture.

5.2.1 Time Complexity

To make matters more complicated, WPDV demands a lot of tagging time, which scales badly (i.e. exponentially) with the growing number of base taggers.

Table 8 shows registered time of training and tagging with WPDV tagger (exactly user time in `time` command terminology). The real time was 10 % longer. With the change of the number of base taggers from 4 to 6 time of tagging augmented from 12 to 90 minutes. One solution to alleviate the problem may be reimplementation of the tagger in a more effective language than Java.

	Simple tagset		Complex tagset	
no. of base taggers	Train	Tagging	Train	Tagging
4 best taggers	5.9	5.1	8.9	$7 \cdot 10^3$
5 best taggers	7.0	17.9	10.8	$23 \cdot 10^3$
6 taggers	8.3	87.8	12.5	$53 \cdot 10^3$

Table 8. Training and tagging time with the WPDV tagger [s]

5.3 Ties and Fallbacks Issues

When dealing with complex methods we had to cope with two main issues: ties and data sparseness.

The Majority method had no training phase and suffered only from ties, which were resolved in a random manner. This introduced some bias to results, which was taken into account by running the method 50 times and providing mean accuracy and its standard deviation.

The TotalPrecision tagger is the most regular method with no data sparseness problem (the only extra parameters needed during tagging are accuracy values of baseline taggers) and almost no ties (accuracy values almost always take different real number, what makes tie pattern extremely hard to encounter).

As TotalPrecision is highly regular and at the same time achieves high accuracy, we decided that this is the method the remaining simple voting taggers fallback to, both in case of data sparseness and ties. Irregular cases accounted for 9.5 %–13.9 % of tokens for lexical methods, and were negligible for TagPrecision and PrecisionRecall methods (for detailed characteristic concerning fallbacks of simple weighted method see Table 9).

Method	Simple tagset	Complex tagset
TagPrecision	0	43 $(42 + 1)$
PrecisionRecall	0	49 $(42 + 7)$
LexAccuracy	6 287 $(6\,178 + 109)$	7 211 $(6\,178 + 1\,033)$
LexPrecision	6 606 $(6\,571 + 35)$	9 145 $(8\,875 + 270)$
LexPrecisionRecall	6 583 $(6\,571 + 12)$	9 001 $(8\,875 + 126)$

Table 9. Number of fallbacks of weighted voting methods to the TotalPrecision method [total number of fallbacks (fallbacks due to data sparseness + fallbacks due to data ties)]

The problems of data sparseness and ties of LexPair and TagPair were tackled in the following way. Data sparseness was overcome by computing required probabilities according to smoothed formulae, accessible from less data (fallback to smoothed probabilities). If simplified formulae were still not enough, final fallbacks to the Majority tagger were undertaken. All the ties (both at main, smoothed probability, and at Majority level) were resolved in a random manner.

The TagPair tagger, the most regular complex voting method, manifests very few ties and probability smoothing determines almost all cases of data sparseness without need of additional fallbacks to the Majority tagger.

The LexPair tagger demands precise data about simple tokens, what is paid with the largest amount of fallbacks among all investigated taggers, accounting from 11 % (simple tagset) to 20 % (complex tagset) of tagged tokens. Moreover, the overwhelming number of fallbacks had to be resolved ultimately by calls to the Majority tagger, as less thorough data were still unavailable. Due to the high number of ties (esp. for the complex tagset) and random bias introduced thereby, the accuracy of LexPair was reported as a mean of 50 subsequent runs of the tagger.

The component taggers of WPDV operate on conditional probabilities, $P(tag_{\mathrm{T}} \mid tag_{i_1}, \ldots, tag_{i_N})$, of a tag, tag_{T}, provided an N-tuple occurred, which are sensitive to data sparseness problem. To avoid numerous chains of fallbacks the cutoff strategy was admitted, taking over 5 % of cases for the complex tagset. No additional fallbacks to Majority and practically no ties occurred.

The interesting feature of LexPair, TagPair and WPDV methods is the ability of assigning a tag different from the tags proposed by the taggers $\mathrm{T}1, \ldots, \mathrm{T}k$. It is not only a theoretical possibility – in one third of cases (LexPair) or nearly 100 % of cases (TagPair, WPDV) such external tags were proposed and in several hundred cases indeed won.

More detailed insight into the number of ties and data sparseness cases for complex voting taggers is given in Table 10.

Pattern	Simple tagset	Complex tagset
external tag proposed	54 411	65 311
external tag wins	0	319
ties	0	7
probability smoothing fallbacks	18	1 096
majority fallbacks	0	42

a) TagPair

Pattern	Simple tagset	Complex tagset
external tag proposed	15 736	24 754
external tag wins	107	599
ties	300	1 785
probability smoothing fallbacks	7 234	12 927
majority fallbacks	7 070	10 998

b) LexPair

Pattern	Simple tagset	Complex tagset
external tag proposed	65 927	65 378
external tag wins	1	407
ties	0	2
cutoff fallbacks	102	3 532
majority fallbacks	0	0

c) WPDV

Table 10. Characteristics of complex voting taggers, [number of cases]

5.4 Split Methods

To give full view of possibilities of various tagging algorithms we adjoin results of thematic split and attribute split models [17, p. 910] in Table 11.

The split methods disturb the distribution of tags over tokens in order to get new, smoother distributions, which would be easier predictable by a tagger. This is achieved by dividing a training corpus into smaller, thematic parts (thematic split approach) or by dividing a complex tagset into a few simple, pairwise disjoint tagsets (attribute split approach).

The split methods are discussed separately in [17].

Tagger	Thematic split	18 %/2 % uniform	Attribute split
	Simple tagset		
TnT	95.49	94.82	–
MXP	94.99	94.46	–
fnTBL	95.16	94.64	–
MBT	94.57	93.80	–
SVM	95.75	94.93	–
TrTg	94.86	94.09	–
	Complex tagset		
TnT	83.28	83.26	81.73
MXP	80.54	79.22	82.55
fnTBL	81.39	80.92	81.73
MBT	78.84	78.03	81.00
SVM	68.26	67.69	84.06
TrTg	78.91	78.59	80.65

Table 11. Accuracy of the thematic split model – average accuracy (arithmetic mean) of all thematic parts, accuracy of taggers trained on the uniform set, accuracy of the attribute split model [%]

6 CONCLUSIONS

We conducted a series of experiments, requiring preparation of test, tuning and training corpora.

The SVM tagger achieves the highest, state-of-the-art accuracy among the baseline methods for the simple tagset, and fnTBL yields both the highest overall accuracy and sentence accuracy for the complex tagset.

Values of theoretical accuracy show that there is still room for improvement of collective methods.

The LexAccuracy and TotalPrecision taggers achieve the highest accuracy (for the simple and complex tagset) among simple weighted methods. The above results indicate that it is important to examine the lexical methods on other languages. The lexical methods should perform even better for English, as then they operate on weakly inflecting word forms and should gather more reliable estimations of accuracy on a word form.

The TagPair and WPDV taggers reach the highest accuracy among all collective taggers. WPDV shows characteristics dissimilar from other baseline and complex taggers – its training time is short while tagging is long. Low tagging speed touching 12 words/sec puts WPDV at a disadvantage in case of applications demanding online processing of large text parts.

Gained error rate reduction equal to 10.83 % is similar to the results communicated by Brill and Wu [4] (10.4 %) or Mihalcea [20] (11.6 %), although lower than van Halteren's 19.1 % [13]. Our work refers to Polish, while the cited authors performed experiments for English. Tagging Polish and Slavic languages is more difficult because of large tagsets used. The above results show that error rate reductions achieved with voting taggers are similar, irrespective of size of a tagset.

Depending on the voting method, the growth of the number of component baseline taggers may lead to increase or decrease of the accuracy. There is no one overall dependance, valid for all voting methods.

Acknowledgments

This research is supported by the AGH-UST grant No. 11.11.120.865. ACC CYFRONET AGH is acknowledged for the computing time.

REFERENCES

[1] Abney, S.—Schapire, R.—Singer, Y.: Boosting Applied to Tagging and PP Attachment. Proceedings of the 1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, pp. 38–45, University of Maryland, College Park, USA, June 21–22, 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. Proceedings of the 3rd Workshop on Very Large Corpora, Association for Computational Linguistics, pp. 1–13, Massachusetts, USA 1995.

[4] Brill, E.—Wu, J.: Classifier Combination for Improved Lexical Disambiguation. Proc. of the 17th Int. Conf. on Computational Linguistics, Montreal, Canada, 1998, pp. 191–195.

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

[6] Collins, M.: Discriminative Training Methods for Hidden Markov Models: Theory and Experiments With Perceptron Algorithms. Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing, Vol. 10, pp. 1–8, University of Pennsylvania, Philadelphia, USA, July 6–7, 2002.

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

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

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

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

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

[12] van Halteren, H.: Weighted Probability Distribution Voting: An introduction. Computational Linguistics in the Netherlands, 1999.

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

[14] van Halteren, H.—Zavrel, J.—Daelemans, W.: Improving Accuracy in Word Class Tagging Through the Combination of Machine Learning Systems. Computational Linguistics, Vol. 27, No. 2, pp. 199–229, June 2001.

[15] Kuta, M.—Chrzaszcz, P—Kitowski, J.: A Case Study of Algorithms for Morphosyntactic Tagging of Polish Language. Computing and Informatics, Vol. 26, 2007, No. 6, pp. 627–647.

[16] Kuta, M.—Chrzaszcz, P.—Kitowski, J.: Increasing Quality of the Corpus of Frequency Dictionary of Contemporary Polish for Morphosyntactic Tagging of the Polish Language. Computing and Informatics, Vol. 28, 2009, No. 3, pp. 319–338.

[17] Kuta, M.—Wrzeszcz, M.—Chrzaszcz, P.–Kitowski, J.: Accuracy of Baseline and Complex Methods Applied to Morphosyntactic Tagging of Polish. Proceedings of the 8th International Conference on Computational Science (ICCS 2008), Cracow, Poland, Part I, LNCS, Vol. 5101, Springer, pp. 903–912, 2008.

[18] Lafferty, J.—McCallum, A.—Pereira, F.: Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. Proceedings of the 18th International Conference on Machine Learning (ICML-2001), pp. 282–289, San Francisco, USA.

[19] Leech, G.—Garside, R.—Bryant, M.: CLAWS4: The Tagging of the British National Corpus. Proceedings of the 15th International Conference on Computational Linguistics (COLING 94), pp. 622–628, Kyoto, Japan 1994.

[20] Mihalcea, R.: Performance Analysis of a Part of Speech Tagging Task. Proceedings of the 4th International Conference on Computational Linguistics and Intelligent Text Processing, pp. 158–166, Mexico City, Mexico 2003.

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

[22] Przepiórkowski, A.: The IPI PAN Corpus: Preliminary Version (in Polish). IPI PAN, Warsaw, Poland 2004.

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

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

[25] Roth, D.—Zelenko, D.: Part of Speech Tagging Using a Network of Linear Separators. Proceedings of the Joint 17th International Conference on Computational Linguistics and 36th Annual Meeting of the Association for Computational Linguistics (ACL/COLING-98), Vol. 2, pp. 1136–1142, Montréal, Quebec, Canada 1998.

[26] Schmid, H.: Probabilistic Part-of-Speech Tagging Using Decision Trees. 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. Proceedings of the 15th 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, Germany 2002.

[29] Toutanova, K.—Klein, D.—Manning, C. D.—Singer, Y.: Feature-Rich Part-of-Speech Tagging With a Cyclic Dependency Network. Proceedings of the Human Language Technology Conference, North American Chapter of the Association for Computational Linguistics, pp. 252–259, Edmonton, Canada, May 27–June 1, 2003.

[30] Voutilainen, A.: A Syntax-Based Part of Speech Analyser. 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] Modified corpus of Frequency Dictionary of Contemporary Polish, `http://nlp.icsr.agh.edu.pl`.

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, machine learning techniques, ontology usage, question answering systems and knowledge engineering. He is the author or co-author of 10 scientific papers.

Wojciech Wójcik is currently an undergraduate student of computer science at the AGH University of Science and Technology in Kraków (Poland). His research interests are in database techniques, and natural language processing. He is the co-author of 2 papers.

Michał Wrzeszcz is an undergraduate student of computer science at the AGH University of Science and Technology in Kraków (Poland). His research interests are in artificial intelligence, distributed computing, and natural language processing. He is the co-author of 4 papers.

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 Dr. Sc. in 1991 in computer science at 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. Made Full Professor in 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, EU IST K-WfGrid, EU int.eu.grid and in EU Gredia. Polish expert in EU Program Committee "e-Infrastructures" (EU Unit F3 "Research Infrastructures") and Director of PL-Grid Consortium (National Grid Initiative, NGI). Member of the Interfaculty Commission of Technical Sciences of the Polish Academy of Arts and Sciences (PAU) and of the Computational Science Section of the Polish Academy of Sciences (PAN), Committee on Informatics.