A tiered CRF tagger for Polish

Adam Radziszewski

Institute of Informatics, Wrocław University of Technology

Abstract In this paper we present a new approach to morphosyntactic tagging of Polish by bringing together Conditional Random Fields and tiered tagging. Our proposal also allows to take advantage of a rich set of morphological features, which resort to an external morphological analyser. The proposed algorithm is implemented as a tagger for Polish. Evaluation of the tagger shows significant improvement in tagging accuracy on two state-of-the-art taggers, namely PANTERA and WMBT.

Key words: morphosyntactic tagging, conditional random fields, Polish, tiered tagging

1 Introduction

Conditional Random Fields (CRF) is a relatively new mathematical model that may be employed to solve sequence labelling problems (Lafferty et al. 2001). The model has been successfully applied to various Natural Language Processing (NLP) tasks, including Part-of-Speech tagging of English. The list of successful applications also includes processing of Polish, e.g., concept-tagging of spoken language corpora (Lehnen et al. 2009), named entity recognition (Marcińczuk and Janicki 2012) and NP chunking (Radziszewski and Pawlaczek 2012).

There have been attempts at morphosyntactic tagging of Polish using CRF. Kuta (2010) reports on using the CRF++ toolkit (Kudo 2005) on Polish data. However, the results are given only for a set-up, where the morphosyntactic tags are limited to grammatical class (roughly speaking, Part-of-Speech). In this paper we argue that it is very difficult to use a single CRF model to perform morphosyntactic tagging of Polish using the full tagset. We propose a simple solution to overcome this difficulty: a tiered CRF tagger. The proposed algorithm is largely inspired by the WMBT memory-based tagger for Polish (Radziszewski and Śniatowski 2011), including its later enhancements (Radziszewski 2012). We perform evaluation of the algorithm and show that it performs significantly better than WMBT, but also PANTERA, another state-of-the-art Polish morphosyntactic tagger (Acedański 2010).

Progess made in morphosyntactic tagging is important, since the error rate at this level contributes to lower quality of next processing stages, e.g. parsing (Hajič et al. 2001). This was also shown for Polish: a CRF-based NP chunker exhibits 1.7 times higher error rate on automatically tagged data than when using reference manual tagging (Radziszewski and Pawlaczek 2012).

In the next section of this paper we introduce CRF and discuss the difficulties in using the model for morphosyntactic tagging of Polish. What follows is a short introduction to the realms of tagging Polish: available corpora, their tagsets, available taggers, common practices and assumptions. In the next section we propose our method of tagging Polish that combines CRF with tiered tagging. The method consists of training and testing algorithms, as well as a feature set designed for Polish. Afterwards we present evaluation, which shows that the tagger implementing our method outperforms two available Polish taggers. The paper is summarised with conclusions.

2 Conditional Random Fields for tagging

CRFs are a family of statistical models similar to Hidden Markov Models (HMM). The fundamental difference is as follows: HMM is a *generative* model, that is, models the joint probability distribution of observations and tags, while CRF is a *conditional* model, since it models the conditional probability distribution of tags given observation sequence, i.e. $P(s_1 \dots s_K | w_1 \dots w_K)$. This allows to avoid undesired assumptions. Most importantly, the probability of transition between tags is not forced to depend on the current observation only (Sutton and McCallum 2011). This enables reasoning based on wide contexts, which seems especially important in the case of NLP tasks, where one faces long-distance syntactic dependencies. What is more, CRF work well with many features, possibly mutually-dependent (Lafferty et al. 2001). This property is particularly welcome when dealing with large and structured tagsets: one may naturally introduce features corresponding to values of various grammatical categories inferred from tags.

Linear-chain CRF is the most popular class of CRFs suitable for tagging that may be descibed using the equation (1). The first sum corresponds to subsequent tokens in text (observations and their tags), the second one deals with features f_j and their weights λ_j. The model assumes that features are characteristic functions. The value of 1 denotes that the predicate represented by a feature is satisfied for i-th position of the labelled token sequence (Lafferty et al. 2001). An example feature function could test if the token at i is a noun, having a word form of `to` and the preceding token is a preposition. CRF training consists in estimation of weight values. A high weight value means that strong evidence has been found to support the relation between observations and tags as expressed by the feature. Z is a normalising function as defined in (2); T^K denotes a set of possible tag sequences of length K assuming tagset T.

$$P(s_1 \dots s_K | w_1 \dots w_K) = \frac{1}{Z(w_1 \dots w_K)} \exp \sum_{i=1}^{K} \sum_{j=1}^{J} \lambda_j f_j(i, s_i, s_{i-1}, (w_1 \dots w_K)) \tag{1}$$

$$Z(w_1 \dots w_K) = \sum_{(s'_1 \dots s'_K) \in T^K} \exp \sum_{i=1}^{K} \sum_{j=1}^{J} \lambda_j f_j(i, s'_i, s'_{i-1}, (w_1 \dots w_K)) \quad (2)$$

Tagging with a trained CRF consists in finding a tag sequence that maximises the conditional probability. Z is independent from the sought sequence, hence the problem may be reduced to maximising the sum of $\lambda_j f_j(i, s_i, s_{i-1}, (w_1 \dots w_K))$ expressions. This may be achieved using dynamic programming methods (Wallach 2004). Training of a CRF tagger requires finding weight values that maximise the probability (1), which may be implemented using maximum likelihood or other estimation methods (Lafferty et al. 2001).

The assumption that features are characteristic functions brings about practical difficulties. If a feature function is to test word forms and/or tags for particular values, those values must be closed over with the function. For each such value configuration, a separate function must be provided in advance. In practice, *feature templates* are used to solve this problem. Feature templates are essentially functions with parameters, whose closures are used as feature functions. There are two types of feature templates commonly used: *unigram templates* (3a), defining functions of the tag and word form occupying the current position and *bigram templates* (3b), also dependent on the tag at the previous position. During training, all the encountered tags and word forms are gathered, and then, the templates are expanded to all their possible instances (Kudo 2005).

$$f_{t_A,v_A}(i, s_i, s_{i-1}, (w_1 \dots w_K)) = \begin{cases} 1, & s_i = t_A \wedge w_i = v_A \\ 0, & \text{else} \end{cases} \quad (3a)$$

$$f_{t_A,t_B,v_A}(i, s_i, s_{i-1}, (w_1 \dots w_K)) = \begin{cases} 1, & s_i = t_A \wedge s_{i-1} = t_B \wedge w_i = v_A \\ 0, & \text{else} \end{cases} \quad (3b)$$

The above formulation assumes that the features operate on word forms and tags. One may similarly define feature templates referring to any transformation of word forms or tags, e.g. returning suffixes of word forms or extracting attribute values from positional tags.

Lafferty et al. (2001) apply CRF to Part-of-Speech tagging of English, reporting 94.5% accuracy using a basic feature set and 95.7% when also using simple transformations of word forms.

Unfortunately, a straightforward application of this model to full morphosyntactic tagging of Polish is computationally unfeasible. This is due to the fact that the tagsets used for Polish contain over 1000 different tags appearing in actual texts (Przepiórkowski 2005), while the time complexity of linear-chain CRF training is quadratic in the size of the label set (that is, the number of different tags appearing in the training data). Strictly speaking, the complexity

is $O(T^2 \cdot J \cdot D^2)$, where T is the number of different tags, J is the number of features defined and D is the number of tokens in the training data (Cohn 2007).

These considerations were confirmed during our initial experiments: attempts at training the CRF++ tagger (Kudo 2005) with the morphosyntactic annotation from the National Corpus of Polish (Przepiórkowski et al. 2010) had to be quickly terminated due to the exhaustion of RAM on our server (24 GB) before the first iteration had been completed. This is also the most likely reason why Kuta (2010) gives 'n/a' instead of actual CRF++ performance figure in tables presenting accuracy values for full morphosyntactic tagging of Polish.

3 Tagging Polish

3.1 Corpora and tagsets

There are two large Polish corpora containing a part with manual morphosyntactic annotation, each with its own tagset:

1. the IPI PAN Corpus (Przepiórkowski 2004, Przepiórkowski and Woliński 2003) contains a 884 273-token manually annotated part (we will refer to this part as IPIC),
2. the National Corpus of Polish (version 1.0; Przepiórkowski et al. 2010) contains a 1 215 513-token manually annotated part (henceforth NCP).

NCP is the largest publicly available Polish corpus suitable for tagger evaluation. Its annotation was performed according to high quality standards, involving the 2+1 model (Przepiórkowski and Murzynowski 2009). On the other hand, about 30% of the IPIC corpus is not publicly available, while the free part was compiled in the 1960s. NCP, on the contrary, represents contemporary Polish (Przepiórkowski et al. 2010). For these reasons we decided to limit our experiments to NCP.

The NCP tagset is a conservative modification of the IPIC tagset (Przepiórkowski 2009). The basic assumption is that the *grammatical classes* (generalisation of the usual part-of-speech notion) are distinguished primarily on the grounds of inflection (Przepiórkowski and Woliński 2003). In consequence, over 30 grammatical classes are defined. Each class is assigned a set of *attributes* (grammatical categories) whose values must be provided. For instance, nouns are specified for number, gender and case, infinitives are specified for aspect. For example, the `subst:sg:nom:m2` tag denotes an animate masculine (`m2`) noun (substantive) in a nominative singular form. A small subset of attributes are deemed *optional* and their values may be omitted.

The size of tagset is frequently quoted as a major source of difficulty of morpho-syntactic tagging (Vidová-Hladká 2000, Hajič et al. 2001, Piasecki and Godlewski 2006). On the other hand, if a principled tagset design may facilitate manual annotation (Przepiórkowski and Woliński 2003), the same may be hoped for automatic tagging.

3.2 Taggers

We are aware of three publicly available, ready-made taggers designed for Polish: TaKIPI, PANTERA and WMBT.

TaKIPI (Piasecki and Godlewski 2006) is based on a small set of hand-written rules and a substantial number of decision tree classifiers that acquire tagging rules automatically. The rule acquisition process is partially driven by hand-written expressions that constitute building blocks for the rules. TaKIPI employs *tiered tagging* (Tufiş 1999), that is parts of the tags, corresponding to subsets of attributes, are dealt with sequentially. TaKIPI defines three tiers, corresponding to grammatical class, number and gender (together), and then case. Unfortunately, the tagger is tied to the IPIC tagset and cannot be tested on NCP.

PANTERA (Acedański 2010) is also based on automatic rule induction. Unlike TaKIPI, almost no language-dependent information is required (besides the training corpus itself). The rule induction is driven by a modified version of Brill's transformation-based learning algorithm (Brill 1992). There are two major modifications that allow to obtain better results for inflectional languages. The main enhancements is that of tiered tagging (two tiers are used). The other one is generalisation of rule patterns to operate on attribute level instead of whole tags.

WMBT (Radziszewski and Śniatowski 2011) is a memory-based tiered tagger. The tagger uses as many tiers as there are attributes in the tagset (plus one for the grammatical class). The algorithm iterates over tiers; tagging of one tier involves classification of subsequent tokens with a k-Nearest Neighbour classifier. The classification process benefits from a rich feature set, including values of particular attribute (grammatical class, number, gender and case for tokens surrounding the token being tagged), but also tests for morphological agreement on number, gender and case.

Later, WMBT was enhanced to better deal with words uknown to the morphological analyser (Radziszewski 2012). Two modifications have been proposed:

- training separate classifiers for known and unkown words,
- morphological reanalysis of training data.

3.3 Reanalysis of training data

Morphological reanalysis of training data is a procedure that allows for a better usage of the available resources, i.e., the available reference corpus and available morphological analyser. The procedure has been proposed as a means for improving WMBT accuracy (Radziszewski 2012).

Taggers that divide the tagging process into morphological analysis and disambiguation usually assume (TaKIPI, PANTERA and WMBT do) that the training data, besides reference tagging, contains results of morphological analysis. That is to say, the reference corpus assigns each token a set of possible tags, out of which one is *highlighted as contextually appropriate*. This information is

used for training. Best results are to be expected if this "reference morphological analysis" matches the behaviour of the analyser used during tagging. This concerns the same forms being unrecognised, but also the same sets of tags attached to recognised forms.

A quick work-around could be to use exactly the same version of the analyser as used for corpus annotation. This solution, however, has two serious shortcomings. First, this forces attachment to a particular version of the analyser, which might be outdated, deprecated or even not available — assuming that the information on the exact analyser version employed for annotation is available at all. What is more, one may be willing to extend the analyser dictionary in hope of improving tagging accuracy, which should be encouraged.

Reanalysis of training data is a simple alternative that allows to update the morphological annotation of the reference corpus with the data taken from the version of the morphological analyser that will be used when tagging. The procedure may be sketched as follows. As NCP and IPIC also assign lemmas to tokens, here we will use the term *morphosyntactic interpretation* to denote a tuple consisting of a tag and a lemma.

1. The training data is turned to plain text.
2. Plain text is fed through the morphological analyser and sentence splitter (for best results it should be exactly the same configuration as used when tagging).
3. The analyser output is synchronised with the original training data in the following manner:
   - Tokenisation is taken from the original training data; should any segmentation change occur, the tokens subjected to it are taken from the original data intact (for simplicity).
   - The remaining tokens (vast majority) are compared; if the *highlighted* interpretation also appears in the reanalysed token, the reanalysed token is taken and the correct interpretation is marked as *highlighted*.
   - If the *highlighted* interpretation is not present there, it means that the tagger would not be able to recover it, hence it is an unknown word. This token is marked as such: we retain only the *highlighted* interpretation and add a non-highlighted interpretation consisting of the "unknown" tag (lemma is set as token's orthographic form, lower-cased). This is to let the tagger see it the same way as when tagging plain text.

The above procedure employs a trick to mark the forms recognised as 'unknown words'. Note that forms that were unknown to the analyser used during manual annotation are marked in a very similar fashion in NCP. An example is presented in Fig. 1. The boxed tags and lemmas correspond to the interpretations highlighted as contextually appropriate in the corpus. Note the first interpretation of ofiarowuje, which is marked as 'unknown word' — it consists of the unknown word tag (`ign`) and artificial lemma None. The above procedure uses word forms as lemmas for unknown words to increase the chance of getting the lemmatisation right, although in this paper we are only concerned with tagging accuracy, while lemmatisation is not assessed.

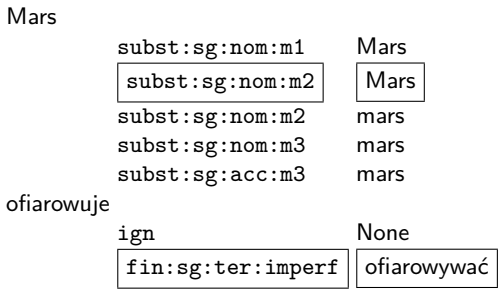

Figure 1. NCP annotation of two tokens, the second being an unkown word.

The purpose of this trick is to give the tagger a valuable information that may be exploited during training: if a particular word form is not present in the actually used morphological dictionary, this form will crop up as an unknown word when tagging with the trained model. Even if, according to the reference corpus, this word form is assigned a set of possible interpretations, this is practically an unknown word and the training algorithm is likely to benefit from this information. Conversely, if a word form is unknown according to the reference corpus, but the new version of the analyser is able to recognise it correctly, the above procedure will update the reference corpus to account for this.

Reanalysis of training data also handles a third scenario: when a word form is known according to both the reference corpus and the morphological analyser employed, but the sets of possible tags attached in both sources do differ. Such situations are likely causes for tagging errors, as they let the same sentence be represented differently when training and when using the trained model to disambiguate. This problem is a practical one. For instance, the reference morphological analysis in NCP version 1.0 is apparently not fully consistent. For instance, different instances of the word form TAK (*yes*, *so*, *thus*) are assigned different sets of possible tags. There are 12 different sets appearing throughout the whole corpus. Figure 2 presents three of them. Such inconsistencies are likely to negatively impact the quality of the trained model.

4 Proposed algorithms for tagger training and performance

The proposed method is based on the improved WMBT tagger. Our main modification is to substitute k-nearest neighbour classification of each sentence token with employing a trained CRF to predict the tagging of the whole sentence at a time. Also, as we perform no classification at token level, the distinction into known and unknown words is used only for pre-processing (appending the list of typical interpretations to unknown words).

```
tak
    adv             tak
    qub             tak
    subst:pl:gen:f taka
tak
    adv:pos         tak
    qub             tak
    subst:pl:gen:f taka
tak
    adv:pos         tak
    qub             tak
    subst:pl:gen:f taka
    conj            tak
```

Figure 2. Three variants of reference morphological analysis of the word form tak in NCP 1.0.

4.1 Training

The training procedure as sketched below assumes that morphological analysis has already been applied. The input is therefore tokens assigned sets of possible morphosyntactic interpretations. We operate according to *tiers* (layers): the first one is responsible for selecting the correct value of the grammatical class, the subsequent correspond to attributes as defined in the tagset (grammatical categories, e.g. number, case). The action performed at each tier is to generate training examples and reproduce the behaviour of normal tagger performance. The latter is executed as partial disambiguation after generating the training examples: the set of interpretations attached to a token at the moment is limited to those that have a particular value of a given tagset attribute (or the grammatical class in case of the first tier). This particular value is taken from the tag highlighted as contextually appropriate. This way we simulate tagger performance under the assumption that a correct decision was made.

The algorithm is parametrised with a set of *features*. For clarity, we assume here that each of these functions transforms the context of a token into a symbolic value, not necessarily binary-valued. The transformation of these features into characteristic functions according to templates defined is the job of the whole procedure described here simply as 'train CRF with training data for tier a and save'. Note that this is also how the actual implementation works: the CRF++ toolkit, which we use, assumes training data is a list of feature vectors (and sentence delimiters) and during training feature templates are used to expand the features into characteristic functions (Kudo 2005).

The training procedure is given as Algorithm 1. The procedure starts with collecting tags that are typical for unkown words. This list is then used to aid tagging of unkown words: each token marked in the training data as unknown word (thanks to the reanalysis stage, Sec. 3.3) is extended with artificial interpretations, each based on a tag from the frequency list. The interpretations added consist of a tag from the list and a lemma equal to the word form (lower-

cased). This step is referred to as 'add tags from the frequency list to *token*'. The next stage consists in generating training examples. Each example consists of a sequence of feature values and the class label, being the correct value of the attribute (taken from the reference tagging). Next, partial disambiguation is performed: the interpretations are removed where the attribute corresponding to the current tier is assigned value other than the one inferred from the manual tagging.

Algorithm 1 Training of the WCRFT algorithm.

```
gather frequency list of tags assigned to unknown words in training corpus
remove tags appearing less than U times from the list
for sentence ∈ corpus do
  for token ∈ sentence do
    if token is an unknown word then
      add tags from the frequency list to token
    end if
  end for
  for a ∈ [class, attr_1, ..., attr_k] do
    for token ∈ sentence do
      f_values ← [f(token, sentence) for f ∈ features(a)]
      decision ← correct value of a for token
      store training example (f_values, decision) for tier a
      remove interpretations from token with incorrect value of a
    end for
    store sentence delimiter for tier a
  end for
end for
for a ∈ [class, attr_1, ..., attr_k] do
  train CRF with training data for tier a and save
end for
```

4.2 Performance

Tagging is done according to Algorithm 2. An array of trained CRF models is used to disambiguate subsequent attributes. As CRF model classifies a whole sentence at a time, first a sentence representation is gathered, being essentially a list of feature vectors (a vector for each sentence token). Note that, as in WMBT, if some classifier decision would require choosing an attribute value that is not present in the list of possible interpretations attached to a token (including those added to unknown words), this decision is not made, leaving the ambiguity pending. The ambiguity may be resolved later when dealing with subsequent attributes. If any ambiguity remains at the end, an arbitrary decision is made.

Algorithm 2 Disambiguation of a single sentence with the WCRFT algorithm.

```
for token ∈ sentence do
  if token is an unknown word then
    add tags from the frequency list to token
  end if
end for
for a ∈ [class, attr_1, ..., attr_k] do
  sent_repr ← []
  for token ∈ sentence do
    f_values ← [f(token, sentence) for f ∈ features(a)]
    append f_values to sent_repr
  end for
  label_list ← classify sent_repr with CRF trained for tier a
  for (token, label) ∈ zip(sentence, label_list) do
    if label ∈ possible values of a for token then
      remove interpretations from token where value(a) ≠ label
    end if
  end for
end for
for token ∈ sentence do
  force "tagset-first" tag if multiple left
end for
```

5 Implementation and feature set

Our implementation is a direct modification of the WMBT tagger. This way we could re-use parts of the existing code, also benefitting from the same tools:

- WCCL[1] (Radziszewski et al. 2011), a toolkit and formalism for morpho-syntactic feature generation,
- Corpus2[2] library (Radziszewski and Śniatowski 2011) for efficient corpus I/O, tagset and tag manipulations.

Both toolkits are distributed with Python wrappers, enabling rapid development of language processing applications.

As the underlying classifier we chose the CRF++ toolkit (Kudo 2005), which also comes with Python wrappers. Thanks to all the above, the implementation was quite straightforward.

The proposed algorithm has been implemented as a configurable tagger for positional tagsets, named WCRFT[3].

[1] Available at `http://nlp.pwr.wroc.pl/redmine/projects/joskipi/wiki/`.

[2] Available at `http://nlp.pwr.wroc.pl/redmine/projects/corpus2/wiki/`.

[3] *Wrocław CRF Tagger* to follow the naming scheme of its predecessor, WMBT (*Wrocław Memory-Based Tagger*). The code has been released under GNU LGPL 3.0 and may be obtained from `http://nlp.pwr.wroc.pl/redmine/projects/wcrft/wiki`.

A WCRFT configuration specifies tagset to use, features defined using WCCL expressions and CRF++ feature templates for each attribute. As mentioned above, features are understood here as functions transforming the context of a token into symbolic values, not necessarily binary. These 'high-level' features are later expanded into characteristic functions for CRF via *feature templates*. First we will describe the features and then we will turn to employed feature templates.

Both the feature set and the templates are taken exactly the same as those used for NP chunking (Radziszewski and Pawlaczek 2012), which in turn are very similar to the original WMBT definitions. Exactly the same feature set is used for all tiers. The features are as follows:

1. word form of the current token,
2. possible values of the grammatical class of the token,
3. possible values of grammatical number,
4. possible values of gender,
5. possible values of grammatical case,
6. a predicate checking if there holds a grammatical agreement of the current and the next token with respect to number, gender and case,
7. a similar predicate that checks the agreement of the previous, current and the next tokens (-1, 0, 1),
8. if the current token's orthographic form starts with an upper-case letter,
9. if it starts with lower-case letter.

Below we list the feature templates employed. We use the notation (p, f) to denote the f-th feature (according to the above enumeration) evaluated at the p-th position, relatively to the current token. Each template generates a number of characteristic functions closed over the domain of the f-th feature as well as the domain of the attribute a corresponding to the current tier. For instance, $(0, 1)$ refers to the first feature from the above list at the position 0, that is the word form of the current token. The expanded template generates tests of the form: 'if the word form of the current token equals w and the value of the current tier attribute is v', where for each word form w appearing in the training data and each value v of the current attribute a separate characteristic function is generated. Similarly, $(-1, 5)$ refers to the value of the grammatical case (feature no. 5) of the token preceding the one currently tagged (hence $p = -1$). The notation $(p_1, f_1)/(p_2, f_2)$ is a template that produces all the conjunctions of the two tests encountered in the data, one for the template (p_1, f_1) and the other one for $(p2, f_2)$.

1. Word forms: $(p, 0)$ for $p \in \{-2, -1, 0, 1, 2\}$
2. Word form bigrams: $(-1, 0)/(0, 0)$ and $(0, 0)/(1, 0)$
3. Grammatical class: $(p, 1)$ for $p \in \{-2, -1, 0, 1, 2\}$
4. Class bigrams: $(-2, 1)/(-1, 1)$, $(-1, 1)/(0, 1)$, $(0, 1)/(1, 1)$, $(1, 1)/(2, 1)$
5. Class trigrams: $(p - 1, 1)/(p, 1)/(p + 1, 1)$ for $p \in \{-2, 0, 1\}$
6. Case: $(p, 2)$ for $p \in \{-2, -1, 0, 1, 2\}$
7. Gender: $(p, 3)$ for $p \in \{-2, -1, 0, 1, 2\}$

8. Number: $(p, 4)$ for $p \in \{-2, -1, 0, 1, 2\}$
9. Agreement: $(-1, 5)$, $(0, 5)$, $(-1, 6)$, $(0, 6)$, $(1, 6)$,
10. Orthographic: $(0, 7)/(0, 8)$

Besides the above templates we used one that employed bigrams of attribute values: 'if the current token's word form is w, the value of the attribute for the current token is v_1 and the value of the attribute for the previous token is v_2'. In the CRF++ terminology such templates are called *bigram templates*, as opposed to unigram templates listed above (hence they test only the value of the attribute related to the current token). Note that we use here the term *attribute* to refer to the class label of the sequence classification tasks, which in the case of the tagger presented here is the grammatical class or a tagset attribute, depending on the tier. We do so to avoid confusion between class labels in general and grammatical classes of the tokens in particular.

6 Evaluation

Radziszewski and Acedański (2012) argue that morphosyntactic taggers should be evaluated as whole systems, that is:

1. The testing material should be turned to plain text.
2. A tagger should perform tokenisation and all the necessary steps (in case of the taggers evaluated here, these include morphological analysis).
3. Tagger output should be compared to the reference annotation and tokenisation of the test material.
4. Differences in tokenisation should be penalised, that is, each token from the reference corpus that is not explicitly present in the tagger output should be treated as tagger's error. This is to promote effort at getting the tokenisation right instead of tweaking with evaluation procedure to match particular taggers' behaviour.

This testing procedure corresponds to a statistic called *accuracy lower bound*. The statistic is based on the assumption that we count the number of reference tokens that are present in tagger output intact (they undergo no segmentations changes) **and** are correctly tagged (the tagger assigned the same tag as in the reference corpus). Accuracy lower bound is the number of such tokens divided by the total number of tokens in the reference corpus, cf. (4).

$$Acc_{lower} = \frac{|\{i : tag(i) = ref(match(i)), i \in match\}|}{N} \tag{4}$$

The above formalisation introduces the mapping $match : N \to N$. The mapping assigns tokens output by the tagger to tokens in the reference corpus. The mapping is only defined for those tokens from the reference corpus that have a lexically corresponding token in tagger output, therefore $i \in match$ denotes that the i-th token of the reference corpus undergoes no segmentation changes when re-tagged (as observed when trying to match tagger output to the reference corpus). $tag(i)$ is used to indicate the i-th tagging of the i-th token (take i-th token

in tagger output and return its tag); $ref(i)$ denotes the tag assigned to the i-th token in the reference corpus.

Following the suggestion of Radziszewski and Acedański (2012), we also provide the values of an additional statistic called *accuracy upper bound*, reflecting a hypothetical upper limit of tagging accuracy. The upper bound is the virtual tagging accuracy under the (false) assumption that each token from the reference corpus that was not explicitly present in tagger output due to unexpected tokenisation would be correctly tagged. This statistic may be formalised as (5). The underlying idea is that some of the actual changes in tokenisation are practically more important, while others are less; the lower and upper bounds define a range where the actual tagging accuracy lies, regardless of which changes in tokenisation we decide to penalise. Note that the accuracy lower bound is the measure recommended for tagger comparison.

$$Acc_{upper} = \frac{|\{i : tag(i) = ref(match(i)), i \in match\}| + |\{i : 0 < i \leq N \wedge i \notin match\}|}{N} \tag{5}$$

Note that both statistics operate on the level of tokens, which include words, but also punctuation, sequences of digits etc. The accuracy figures are counted including all the tokens without any distinction between correctly tagging a word and a punctuation mark or another string.

Our evaluation has been performed using the data from NCP 1.0 (1 215 513 tokens) and 10-fold cross-validation. Strictly speaking, we used exactly the same data (including the same division into training and test folds, the same version of the morphological analyser) as used for the experiments reported by Radziszewski and Acedański (2012), hence the results are fully comparable.

We report values of accuracy lower and upper bound (Acc_{lower}, Acc_{upper}), but also, accuracy lower bound measured separately for words known and unknown to the morphological analyser (Acc^{K}_{lower} and Acc^{U}_{lower}, respectively).

Both versions of WMBT, as well as the algorithm proposed here, assume that the input has been tokenised and morphologically analysed. To obtain this structure from plain text we use the MACA software (Radziszewski and Śniatowski 2011) and the configuration named `morfeusz-nkjp-official-guesser`, as suggested in Radziszewski and Śniatowski (2011).

Table 1 shows the values of *accuracy lower bound* and other measures for the following taggers:

PANTERA with default parameter values (threshold value of 6). The tagger was run against plain text to take advantage of its heuristics to solve segmentation ambiguities.

WMBT — the basic algorithm as proposed in Radziszewski and Śniatowski (2011).

WMBT+u — the modified algorithm with handling unknown words. Also, the training data were subjected to morphological reanalysis beforehand.

WCRFT — the algorithm proposed here. Reanalysis of training data was employed as well.

The improvement over PANTERA and WMBT+u is statistically significant (as measured in accuracy lower bound, using paired t-test with 95% confidence). The improvement in accuracy lower bound in comparison with both WMBT variants is mirrored in accuracy upper bound increase, which was expected as the same toolchain was employed for tokenisation and morphological analysis. It may be noticed that the proposed CRF-based algorithm exhibits slightly higher error rate for unknown words than the WMBT+u.

Tagger	Acc_{lower}	Acc_{upper}	Acc^{K}_{lower}	Acc^{U}_{lower}
WMBT	87.50%	87.82%	89.78%	13.57%
PANTERA	88.79%	89.09%	91.08%	14.70%
WMBT+u	89.71%	90.04%	91.20%	41.45%
WCRFT	90.34%	90.67%	91.89%	40.13%

Table 1. Accuracy measures obtained during evaluation on NCP 1.0.

The performance of the tagger is about 1.5 times faster that that of WMBT+u. We recorded 269 tokens per second for WMBT+u and 408 tokens per second for WCRFT on an Intel Core i7 machine. These figures does not include using MACA, although its overhead is minor (the employed MACA configuration results in over 18k tokens output per second).

The training of WCRFT is unfortunately much slower: training ten models (one for each training fold) using three concurrent processes on a server with Intel Xeon 2.67GHz CPU took 7 days.

7 Conclusion

We have shown that application of Conditional Random Fields together with tiered tagging and a rich feature set allows to achieve improvement on state-of-the-art results in tagging Polish. The practical value of the work presented here lies in the implemented tagger, WCRFT. The tagger will be used internally in the SyNaT project, but is also publicly available under GNU LGPL. What is more, a dedicated web-service is being developed at the moment, which will facilitate using the tagger.

A practical disadvantage of the tagger is its relatively low tagging speed. Most likely it could be made considerably faster by merging together tiers corresponding to some attribites. Initial experiments show that the training data for a couple of less important attributes are almost 'empty', that is only a small percentage of tokens assign any value to the attributes. This is due to the characteristics of the tagset: the set of attributes whose value must be given depends on the grammatical class. What is more, many of the ambiguities are already resolved after dealing with grammatical class, number, gender and case (this was the reason why disambiguation in TaKIPI stopped here, remaining infrequent

ambiguities at output). This intuition could be exploited to boost the tagger performance by reducing the number of tiers.

It would also be interesting to test the tagger on data from another Slavic languages, e.g. Croatian or Slovene. We consider adding support for MULTEXT-East tagsets (Erjavec 2012) to Corpus2 library for this purpose.

Acknowledgement

This work was financed by the National Centre for Research and Development (NCBiR) project SP/I/1/77065/10 ("SyNaT").

Bibliography

Acedański, S. (2010). A morphosyntactic brill tagger for inflectional languages. In Loftsson, H., Rögnvaldsson, E., and Helgadóttir, S., editors, *Advances in Natural Language Processing*, volume 6233 of *Lecture Notes in Computer Science*, pages 3–14. Springer Berlin / Heidelberg.

Brill, E. (1992). A simple rule-based part of speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing*, pages 152–155, Morristown, USA. Association for Computational Linguistics.

Cohn, T. (2007). *Scaling conditional random fields for natural language processing*. PhD thesis, Department of Computer Science and Software Engineering, University of Melbourne, Australia.

Erjavec, T. (2012). MULTEXT-East: morphosyntactic resources for Central and Eastern European languages. *Language Resources and Evaluation*, 46(1):131–142.

Hajič, J., Krbec, P., Květoň, P., Oliva, K., and Petkevič, V. (2001). Serial combination of rules and statistics: A case study in Czech tagging. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 268–275. Association for Computational Linguistics.

Kudo, T. (2005). CRF++: Yet another CRF toolkit. User's manual and implementation available at http://crfpp.googlecode.com/svn/trunk/doc/index.html.

Kuta, M. (2010). *Tagging and Corpus based Methods for improving Natural Language Processing of Polish*. PhD thesis, Wydział Elektrotechniki, Automatyki, Informatyki i Elektroniki, Akademia Górniczo-Hutnicza, Kraków.

Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML-2001)*.

Lehnen, P., Hahn, S., Ney, H., and Mykowiecka, A. (2009). Large-scale Polish SLU. In *Interspeech*, pages 2723–2726, Brighton, UK.

Marcińczuk, M. and Janicki, M. (2012). Optimizing CRF-based model for proper name recognition in Polish texts. In Gelbukh, A., editor, *CICLing 2012, Part I*, volume 7181 of *LNCS*, pages 258–269. Springer, Heidelberg.

Piasecki, M. and Godlewski, G. (2006). Effective architecture of the Polish tagger. In *Text, Speech and Dialogue*, volume 4188, pages 213–220, Brno, Czechy. Springer.

Przepiórkowski, A. (2004). *The IPI PAN Corpus: Preliminary version*. Institute of Computer Science, Polish Academy of Sciences, Warsaw.

Przepiórkowski, A. (2005). The IPI PAN Corpus in numbers. In Vetulani, Z., editor, *Proceedings of the 2nd Language & Technology Conference*, Poznań, Poland.

Przepiórkowski, A. (2009). A comparison of two morphosyntactic tagsets of Polish. In Koseska-Toszewa, V., Dimitrova, L., and Roszko, R., editors, *Repre-*

senting Semantics in Digital Lexicography: Proceedings of MONDILEX Fourth Open Workshop, pages 138–144, Warsaw.

Przepiórkowski, A. and Woliński, M. (2003). A flexemic tagset for Polish. In *Proceedings of* Morphological Processing of Slavic Languages*, EACL 2003*.

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

Przepiórkowski, A. and Murzynowski, G. (2009). Manual annotation of the National Corpus of Polish with Anotatornia. In Goźdź-Roszkowski, S., editor, *The proceedings of Practical Applications in Language and Computers PALC 2009*, Frankfurt, Germany. Peter Lang.

Przepiórkowski, A. and Woliński, M. (2003). 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*.

Radziszewski, A. (2012). *Treatment of unknown words in WMBT*. Wrocław University of Technology. http://nlp.pwr.wroc.pl/redmine/projects/wmbt/wiki/Guessing.

Radziszewski, A. and Acedański, S. (2012). Taggers gonna tag: an argument against evaluating disambiguation capacities of morphosyntactic taggers. To be published in proceedings of TSD 2012.

Radziszewski, A. and Pawlaczek, A. (2012). Large-scale experiments with NP chunking of Polish. To be published in proceedings of TSD 2012.

Radziszewski, A. and Śniatowski, T. (2011). Maca — a configurable tool to integrate Polish morphological data. In *Proceedings of the Second International Workshop on Free/Open-Source Rule-Based Machine Translation*.

Radziszewski, A. and Śniatowski, T. (2011). A memory-based tagger for Polish. In *Proceedings of the 5th Language & Technology Conference, Poznań*.

Radziszewski, A., Wardyński, A., and Śniatowski, T. (2011). WCCL: A morphosyntactic feature toolkit. In *Proceedings of the Balto-Slavonic Natural Language Processing Workshop*. Springer.

Sutton, C. and McCallum, A. (2011). An introduction to conditional random fields. In *Foundations and Trends in Machine Learning*.

Tufiş, D. (1999). Tiered tagging and combined language models classifiers. In Matousek, V., Mautner, P., Ocelíková, J., and Sojka, P., editors, *Text, Speech and Dialogue*, volume 1692 of *Lecture Notes in Computer Science*, pages 843–843. Springer Berlin / Heidelberg.

Vidová-Hladká, B. (2000). *Czech Language Tagging*. PhD thesis, Uniwersytet Karola, Wydział Matematyki i Fizyki, Praga.

Wallach, H. M. (2004). Conditional random fields: An introduction. Technical Report MS-CIS-04-21, Department of Computer and Information Science, University of Pennsylvania, USA.