A Memory-Based Tagger for Polish

Adam Radziszewski, Tomasz Śniatowski

Institute of Informatics
Wrocław University of Technology
Wybrzeże Wyspiańskiego 27,
Wrocław, Poland

Abstract

We present and evaluate a tiered memory-based tagger for Polish. The tagger is tested on two manually annotated corpora and achieves state-of-the-art results. We also show that the tagger used in a simple voting ensemble allows to achieve a significant increase in accuracy as compared to previous experiements in voting.

1. Introduction

Part-of-speech (POS) tagging is a common and well-researched Natural Language Processing (NLP) task. It is the process of assigning morpho-syntactic tags to words and word-like units (*tokens*) in text. In languages with rich morphology the tags usually include significantly more information than just parts-of-speech, e.g. nouns may be specified for values of number, gender and case, adverbs may be specified for degree. In such a setting, the tags are often called morpho-syntactic description (MSD) tags and the task is referred to as morpho-syntactic tagging. The set of possible tags, principles of forming tags from atomic symbols and guidelines on practical tag usage are collectively referred to as a *tagset*.

The aim of this paper is to present a new tagging algorithm and evaluate it on two Polish corpora.

2. Tagging Polish

In the case of Polish, there are two large corpora containing a part with manual morpho-syntactic annotations, 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 and its tagset as IPIC),

2. the recently made available National Corpus of Polish (version 1.0) contains a 1 215 513-token manually annotated part (the part and its tagset we will call NCP).

The IPIC and NCP tagsets are similar. 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, there are over 30 grammatical classes. Each class is assigned a set of *attributes* (grammatical categories) whose values must be provided (e.g. nouns are specified for number, gender and case). An exception to this rule is the concept of *optional attributes* whose values may be omitted.

A standard measure for tagger evaluation is *accuracy*, the percentage of tokens the tested tagger assigned the correct (reference) tag. This measure requires no ambiguity in both the tagger output and the reference tagging. In the case of IPIC, the reference tagging assigns multiple correct tags to some tokens. What is more, depending on the tagger used, one may expect to obtain an ambiguous output for similar cases. Several measures have been proposed to deal with such ambiguities (Acedański and Przepiórkowski, 2010). We will use two: *strong correctness* and *weak correctness*. Strong correctness is the percentage of tokens where the tagger assigned exactly the reference set of tags, while weak correctness counts a token as correctly tagged if the tagger output intersects with the reference tagging.

There are two popular, ready-made taggers designed for Polish: TaKIPI and Pantera. TaKIPI (Piasecki, 2007) 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. Pantera (Acedański, 2010) is also based on automatic rule induction. Unlike TaKIPI, almost no language-dependent information is required. The rule induction is driven by a modified version of Brill's transformation-based learning algorithm (Brill, 1992). Both taggers employ variations of the *tiered tagging* model (Tufiş, 1999), that is the disambiguation procedure is divided into tiers (layers), corresponding to subsets of tagset attributes. Pantera was reported to achieve slightly better results as tested on IPIC (Śniatowski and Piasecki, 2011). Another advantage of Pantera is that it is configurable, while TaKIPI is hardcoded to the IPIC tagset.

3. Memory-Based Tagging

Memory-Based Learning (MBL) is a Machine Learning technique, where the training phase consists in storing the training examples without abstraction or restructuring. During classification the input is compared to the stored examples. k most similar examples are retrieved and (possibly weighted) majority voting is used to choose the class

The work was co-funded by the European Union Innovative Economy Programme project NEKST, No POIG.01.01.02-14-013/09.

label. The method may be parametrised with (Daelemans and van den Bosch, 2005):

1. the number of neighbours (k),
2. a similarity metric between feature values,
3. a scheme of feature weighting,
4. a scheme of weighting neighbours as a function of their distance (this influences the final voting).

MBL has been successfully applied to a number of NLP tasks (Daelemans and van den Bosch, 2005). In the case of morpho-syntactic tagging, the problem must be first formulated as a classification task. This may be achieved by representing the context of each token as a fixed-width *feature vector*. The features usually include parts-of-speech of neighbouring tokens, their orthographic forms, sometimes also fixed-width affixes of the word forms (Daelemans and van den Bosch, 2005).

There exists a popular open-source framework for MBL called TiMBL (Daelemans et al., 2010a) with a package for memory-based tagging — MBT (Daelemans et al., 2010b). MBT is able to train a working tagger using an annotated corpus. The corpus must contain a sequence of tokens and their corresponding MSD tags. Optionally, external features may also be included. During training a lexicon of frequent word forms is generated. The lexicon is a mapping of word forms into ordered sets of encountered MSD tags (ordered by their conditional frequencies). Rare forms are treated separately to account for unknown words when tagging. The default set of features includes form suffixes to facilitate guessing of correct tag.

A drawback of MBT when applied to inflectional languages is that it treats the feature values atomically, hence it cannot reason using the attribute values inferred from tags. This, however, can be altered by introducing additional features directly to the input before running MBT. We tested this possibility on the manually disambiguated part of IPIC. The experiments (based on standard 10-fold cross-validation technique and employing the feature set discussed later) resulted in 86,9% weak correctness, which is far from being satisfactory. This was our motivation to construct a memory-based tagger for inflectional languages that would benefit from the positional character of the tagset.

4. Proposed algorithm

Our idea was to employ tiered tagging in tandem with MBL. In contrast to the approaches summarised in the previous section, we consider as many layers (tiers) as there are attributes defined in the tagset.

Our assumption is that the input has already been subjected to morphological analysis (initial tagging), i.e. each token is assigned a set of possible tags (similar assumptions hold for the underlying algorithms of Pantera and TaKIPI). In this setting, the remaining task is that of contextual disambiguation, i.e. successive ruling out of contextually inappropriate tags until exactly one tag remains.

The algorithm is parametrised with a set of *features*, i.e. functions transforming the context of a token into a string or symbolic value. For instance, a very basic feature set may consist of the word form of the token being disambiguated, its predecessor and its successor. A simple extension is to add grammatical class values and some attribute values for those tokens. The algorithm allows to specify a different feature set for each attribute. Note that, depending on the stage of disambiguation, some attributes may be ambiguous (returned as sets of values).

The outcome of the training phase is a separate list of training examples (a case base) for each of the attributes. The procedure is given as Algorithm 1. A training example consists of a sequence of feature values and the class label, being the correct value of the attribute (taken from the reference tagging). In cases where the reference tagging contains more than one tag per token (as happens in IPIC), one tag is arbitrarily chosen (consisting of values that come first in the tagset definition).

Algorithm 1 Generating training examples

```
gather $F$ most frequent word forms from $corpus$
for $sent \in corpus$ do
  for $a \in [class, attr_1, \ldots, attr_k]$ do
    for $tok \in sent$ do
      if $tok$ is ambiguous w.r.t. $a$ then
        $f\_values \leftarrow [f(tok, sent)$ for $f \in feats(a)]$
        $class\_label \leftarrow$ correct value of $a$ for $tok$
        gen. example $(a, feat\_values, class\_label)$
        rem. tags from $tok$ with incorr. value of $a$
      end if
    end for
  end for
end for
```

The performance phase (regular disambiguation) is shown in Algorithm 2. An array of MBL classifiers is used to disambiguate subsequent attributes. If at any layer a classifier's decision leads to discarding of all the remaining tags, the decision is cancelled, possibly leaving an ambiguity pending. Note that the same ambiguity may be resolved later when disambiguating a different attribute. If an ambiguity remains after all layers have been run, an arbitrary decision is made.

5. Implementation

We implemented the described algorithm as a configurable tagger for positional tagsets, named WMBT[1]. WMBT was implemented in Python using the following C++ libraries along with their Python wrappers:

- TiMBL (Daelemans et al., 2010a) and its Python API[2] for the implementation of the MBL classifier;
- WCCL (Radziszewski et al., 2011), a toolkit for morpho-syntactic features,

[1] Wrocław Memory-Based Tagger. The tagger has been made available under GNU GPL at http://nlp.pwr.wroc.pl/redmine/projects/wmbt/wiki/

[2] http://ilk.uvt.nl/~sander/software/python-timbl.html

Algorithm 2 Tagging a sentence

```
for $a \in [class, attr_1, \ldots, attr_k]$ do
  for $tok \in sent$ do
    if $tok$ is ambiguous w.r.t. $a$ then
      $f\_values \leftarrow [f(tok, sent)$ for $f \in feats(a)]$
      $class\_label \leftarrow$ classify $(a, f\_values)$
      if $class\_label \in$ possible values of $a$ for $tok$ then
        rem. tags from $tok$ where value($a$) $\neq$ $class\_label$
      end if
    end if
  end for
end for
for $tok \in sent$ do
  force “tagset-first” tag if multiple left
end for
```

- Corpus2 library (Radziszewski and Śniatowski, 2011) for efficient corpus I/O, tagset and tag manipulations.

The user may provide different tagger configurations, affecting TiMBL classifier parameters, the employed tagset, the ordering of tagset attributes and the set of features assigned to each attribute. The features are described in the WCCL language, which allows for high expressive power.

Our set of features is largely inspired by the experiments in chunking of Polish (Radziszewski et al., 2011). The features include:

- values of grammatical class, number, gender and case of the tokens surrounding the position being disambiguated — using a fixed window (-3, -2, -2, 0, 1, 2),

- orthographical forms (lower-case) of the tokens in the window or empty symbols if the corresponding form was not on the list of 500 most frequent forms (gathered from the training data)

- true/false features denoting whether there is a possibility of a grammatical agreement holding for the values of number, gender and case of tokens in a range; we defined five such features, corresponding to the following token ranges containing the position being disambiguated: (-1, 0), (0, 1), (-2, -1, 0), (-1, 0, 1), (0, 1, 2).

The above set of features was used for all the attributes. Additionally, the value of the attribute in question for the token being disambiguated was used added as the last feature for all the attributes besides the grammatical class, number, gender and case (their values are already included). This was to give the classifier an insight into the set of possible values to choose from.

Note that this feature set is by no means complete. WCCL allows quite complex expressions which could be used e.g. to test for long-distance dependencies.

We used the following TiMBL parameters: Modified Value Difference metric, Information Gain feature weighting (default), $k = 11$ with inverse linear voting weights. Note that those values have been chosen mostly arbitrarily with a very small number of initial experiments, all done on IPIC (this could lead to slight overtraining on IPIC, while the experiments on NCP are unbiased). Systematic experiments with parameter values are planned as further work.

6. Evaluation

The evaluation was carried out using the same methodology as described in (Śniatowski and Piasecki, 2011): ten random splits of the data into training (9/10) and test (1/10) parts were made and all the experiments were run on those data sets. The reported figures are average values across ten runs. Paired t-test with 95% confidence was used to test statistical significance of improvement.

TaKIPI and Pantera were used with default settings[3].

The results of WMBT on the IPIC in comparison with TaKIPI and Pantera are presented in Table 1. Both weak (WC) and strong correctness (SC) is slightly better than that of Pantera (the difference is statistically significant). Note that SC is an unfavourable measure for both WMBT and Pantera, as they are bound to select one tag, while in the IPIC some tokens are assigned multiple tags.

Tagger	WC	WC dev.	SC	SC dev.
TaKIPI	92.93%	0.09%	90.27%	0.12%
Pantera	92.98%	0.07%	90.57%	0.12%
WMBT	93.16%	0.09%	90.66%	0.10%

Table 1: Tagger performance on the IPI PAN corpus. WC stands for weak correctness while SC stands for strong correctness. Standard deviation is also given.

The results for NCP are reported in Table 2. As the corpus as well as the two taggers assume one tag per token, only disambiguation accuracy is given. The WMBT performance is on par with that of Pantera (the difference is statistically insignificant).

Tagger	Accuracy	std dev.
Pantera	92.95%	0.06%
WMBT	92.98%	0.10%

Table 2: Disambiguation accuracy and standard deviation as measured on the NCP corpus.

Note that the differences in performance figures between WMBT and Pantera are not big, while both taggers allow adjustment of parameter values. Thus, any claim on which tagger is better in general would be premature.

A practical disadvantage of WMBT is its low tagging speed. We recorded 372 tokens per second performance on an Intel Core i7 machine (while testing on the NCP). Initial experiments suggest that the bottleneck lies within

[3] Pantera was trained and run as described in the on-line user guide (Acedański, 2011). To facilitate fair comparison, TaKIPI was used with special treatment of noun-gerund ambiguities turned off, as in (Śniatowski and Piasecki, 2011).

the TiMBL classifier. It is likely that some improvement could be obtained by removing less significant attributes from the model. It is worth noting that training is much faster: we recorded as much as 2280 tokens per second on the NCP.

7. As part of an ensemble tagger

Building upon the work described in (Śniatowski and Piasecki, 2011) we investigated how adding our new tagger to a simple voting ensemble can affect the results. The previous work describes combining Pantera with TaKIPI and RFTagger (Schmid and Laws, 2008) and achieves weak correctness of roughly 94% when tested on the IPI PAN corpus. (Śniatowski, 2011) adds a fourth tagger, a prototype maximum entropy tagger of Polish described in (Mastalerz, 2011), but does not register a significant improvement over the original ensemble of three.

Our new results for both weak and strong correctness of new ensembles made possible by introducing WMBT are presented in Table 3. A a five-tagger ensemble achieves a statistically significant increase in both evaluation metrics over the previously-described voting tagger. Somewhat surprisingly, a three-tagger ensemble consisting of the three best taggers (Pantera, TaKIPI and WMBT) performed no better than the original voting tagger.

Tagger	Weak correctness	Std. dev.
Pantera	92.98%	0.07%
TaKIPI	92.93%	0.09%
RFTagger	89.78%	0.05%
3-way Voting	93.96%	0.06%
WMBT	93.16%	0.09%
maxent	89.70%	0.12%
best-3 3-way	93.96%	0.06%
5-way Voting	94.32%	0.07%

Table 3: Voting with WMBT: IPI PAN corpus

We have also tested the voting tagger on the NCP 1.0 corpus using three taggers: Pantera, WMBT and RFTagger. We only used a subset of all taggers available due to difficulties in adapting TaKIPI to a different tagset and the low performance of the entropy tagger. The results are shown in Table 4 and, as expected, the voting tagger achieves a significant increase in accuracy over the best individual tagger.

Tagger	Accuracy	Std. dev.
Pantera	92.95%	0.06%
RFTagger	89.09%	0.08%
WMBT	92.98%	0.10%
Voting	94.11%	0.07%

Table 4: Voting with WMBT: NCP corpus

8. Conclusions

We presented a simple approach to morpho-syntactic tagging suitable for languages with positional tagsets. The approach combines tiered tagging with Memory-Based Learning. The proposed algorithm was implemented in WMBT, a tagger for Polish. WMBT is a simple configurable Python module that employs several publicly available libraries. WMBT achieves state-of-the-art results on two different Polish corpora. What is more, the inclusion of WMBT into a simple tagger ensemble offers a significant improvement over a previously tested ensemble.

The work described here is, to the best of our knowledge, the first attempt at applying Memory-Based Learning techniques to tagging Polish. Our choice of classifier parameters and feature sets is not backed up by any systematic research, thus there should be some room for improvement. Also, the proposed algorithm (as well as our implementation) allows for selecting the order of attribute disambiguation; we did not investigate this possibility, which could make a difference.

9. References

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

Acedański, Szymon, 2011. Pantera tagger user's guide. Available at http://code.google.com/p/pantera-tagger/wiki/UserGuide (accessed 3.08.2011).

Acedański, Szymon and Adam Przepiórkowski, 2010. Towards the adequate evaluation of morphosyntactic taggers. In *Proceedings of COLING 2010*.

Brill, Eric, 1992. A simple rule-based part of speech tagger. In *Proceedings of the third conference on Applied natural language processing*. H92-1022.pdf.

Daelemans, Walter and Antal van den Bosch, 2005. *Memory-Based Language Processing*. Cambridge University Press.

Daelemans, Walter, Jakub Zavrel, and Antal Van den Bosch Ko van der Sloot, 2010a. TiMBL: Tilburg Memory Based Learner, version 6.3, reference guide. Technical Report 10-01, ILK.

Daelemans, Walter, Jakub Zavrel, Antal Van den Bosch, and Ko van der Sloot, 2010b. MBT: Memory-Based Tagger, version 3.2. Technical Report 10-04, ILK.

Mastalerz, Radomir, 2011. *Tager maksimum entropii dla języka polskiego*. Master's thesis, Uniwersytet Warszawski, Wydział Matematyki, Informatyki i Mechaniki.

Piasecki, Maciej, 2007. Polish tagger TaKIPI: Rule based construction and optimisation. *Task Quarterly*, 11(1–2):151–167.

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

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

Radziszewski, Adam and Tomasz Śniatowski, 2011. Maca — a configurable tool to integrate Polish morphological data. In *Proceedings of FreeRBMT11*.

Radziszewski, Adam, Adam Wardyński, and Tomasz Śniatowski, 2011. WCCL: A morpho-syntactic feature toolkit. To appear in: Proceedings of the Balto-Slavonic Natural Language Processing Workshop.

Schmid, H. and F. Laws, 2008. Estimation of conditional probabilities with decision trees and an application to fine-grained POS tagging. In *Proceedings of COLING 2008*, volume 1. Association for Computational Linguistics.

Śniatowski, Tomasz, 2011. *Metody analizy tekstu w języku polskim na poziomie wyrazowym*. Master's thesis, Politechnika Wrocławska, Wydział Informatyki i Zarządzania.

Śniatowski, Tomasz and Maciej Piasecki, 2011. Combining Polish Morphosyntactic Taggers. In *Proceedings of the 2011 International Joint Conference on Security and Intelligent Information Systems*. Springer Berlin / Heidelberg.

Tufiş, Dan, 1999. Tiered tagging and combined language models classifiers. In Václav Matousek, Pavel Mautner, Jana Ocelíková, and Petr Sojka (eds.), *Text, Speech and Dialogue*, volume 1692 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, pages 843–843.