A Comparison of Polish Taggers in the Application for Automatic Speech Recognition

Aleksander Pohl, Bartosz Ziółko

AGH University of Science and Technology
Faculty of Computer Science, Electronics and Telecommunications, dsp.agh.edu.pl
30 Mickiewicza Street, Kraków, Poland, techno.pl
aleksander.pohl@uj.edu.pl, bziolko@agh.edu.pl

Abstract

In this paper we investigate the performance of Polish taggers in the context of automatic speech recognition (ASR). We use a morphosyntactic language model to improve speech recognition in an ASR system and seek the best Polish tagger for our needs. Polish is an inflectional language and an n-gram model using morphosyntactic features, which reduces data sparsity seems to be a good choice. We investigate the difference between the morphosyntactic taggers in that context. We compare the results of tagging with respect to the reduction of word error rate as well as speed of tagging. As it turns out at present the taggers using conditional random fields (CRF) models perform the best in the context of ASR. A broader audience might be also interested in the other discussed features of the taggers such as easiness of installation and usage, which are usually not covered in the papers describing such systems.

Keywords: morphosyntactic tagger, Polish, automatic speech recognition, language model

1. Introduction

Unlike English, which is a positional language, Polish has a rich morphology, with many morphosyntactic features. This boils down to the observation that many syntactic features that in English are encoded in the relative position of words, in Polish are encoded in the suffix of the word. For instance the expressions dom Adam and Adama dom (Adam’s house) although not equally probable, express the same relation between these words. What is more the number of tokens in Polish and other inflectional languages is larger than in English, since words have many forms (e.g. Adam, Adama, Adamowi, Adamem, Adamicie, Adamowicie, ... are all forms of Adam).

These two facts have important implications when building a language model for an ASR system for Polish (Ziółko and Ziółko, 2011). The first one makes the generally accepted methods improving language models, namely class-based n-grams (Brown, 1992) less useful, since they are based only on the positions of words. The second means that when building word-based language model for Polish, the size of the corpus has to be substantially larger than for English, in order to overcome the data-sparness problem.

In this research we investigate the differences in the performance of taggers in the application for ASR. We want to find out which of the available taggers is the best in terms of tagging quality and speed. Since there are many taggers designed specifically for Polish we are not going to develop our own solution. As a result we assess the primary features of the taggers such as accuracy and speed, but we have also an opportunity to compare their secondary features, such as the easiness of installation and their licenses.

Even though there are results showing which of the implemented taggers performs the best on the reference corpus (Concraft) (Waszczuk, 2012), we want to find out if the differences in accuracy are preserved in a setting which is substantially different from the original one. This is caused by the large number of ungrammatical sentences that are present in the output of an acoustic module as well as restriction on the number of employed grammatical categories (part-of-speech\(^1\) (POS), number, gender and case).

2. Taggers

The comparison of the taggers is restricted to the following systems: WMBT (Radziszewski, 2011), Pantera (Acedański, 2010), WCRFT (Radziszewski, 2013) and Concraft (Waszczuk, 2012). These are the most up-to-date\(^2\), publicly available systems enlisted on the “Computational Linguistics in Poland”\(^3\) web-page (in the section “Language Tools and Resources for Polish”), which were developed specifically for Polish.

The comparison has the following structure. First we present a short description of the technique used by the tagger, together with the information about its license. Second we describe the issues (if any) connected with the installation and usage of the tagger. Then we present the general overview of the technique implemented in the tagger. We conclude the presentation with a more detailed description of the adaptations employed to solve specific Polish tagging issues.

2.1 WMBT

WMBT\(^4\) (Radziszewski, 2011) is a tagger that utilizes the Memory Based Learning (MBL) technique and is distributed under a Lesser General Public License (LGPL). It uses the TiMBL library (Daelemans et al.,

\(^1\)We use the terms part-of-speech and grammatical class interchangeably in this document, due to the way there are used in the literature regarding Polish tagsets and taggers.

\(^2\)We do not include in the comparison TaKIPi (Piatecki, 2007) tagger as well as TnT tagger (Brants, 2000) which are reported to be inferior to all the presented techniques (with respect to tagging of Polish).

\(^3\)http://olip.ipipan.waw.pl/LRT

\(^4\)http://nlp.pwr.wroc.pl/redmine/projects/wmbt

294
and although it comes with a specific tool designed for tagging, WMBT uses only its general MBL capabilities.

The installation of WMBT is not straightforward and requires manual installation of several other libraries: Maca (Radziszewski and Śniadowski 2011), Corpus2, Morfeusz (Woźniak 2006), WCCL (Radziszewski et al. 2011) and TiMBL. The first library is used for splitting the analyzed text into paragraphs and segments ans as a proxy to the morphological analyzer. Corpus2 provides an efficient access to corpora — National Corpus of Polish (NCP) (Przepiórkowski, 2012) in particular. Morfeusz is a popular library for morphological analysis of Polish and is used in all other taggers. WCCL provides a formalism for expressing and transforming various lexical and morphosyntactic features, such as case agreement.

The installation requires manual downloading of some of the tools, since not all of them are provided as packages for popular operating systems (e.g. Ubuntu). As a result the installation is a bit cumbersome. Regarding usage: running WMBL on a plain text requires a separate call to Maca, for input preprocessing.

The general idea behind MBL-based tagging (Daelemans and van den Bosch, 2005) is as follows: during the training phase, the word occurrences are transformed into feature-vectors which are, together with the correct value of the morphosyntactic label, directly stored in the memory of the tagger, i.e. they are simply recorded. During the disambiguation phase words are also transformed into feature-vectors, the tagger consults its memory and finds the vectors which are the most similar (w.r.t a selected metric) to the vector in question and selects the best label using voting among the k-most similar examples.

WMBL uses MBL together with tiered tagging (Tufis, 1999). This is due to the fact, that Polish morphosyntactic labels are positional, i.e. the values of various morphosyntactic categories applicable for a given grammatical class are concatenated and form a complex label. As such the number of possible and also the empirically observed distinct labels is large (more than 4 thousand and 1 thousand respectively). To overcome the data-sparseness problem WMBT disambiguates the input using a sequence of tiers, each using a separate model capturing the features of only one grammatical category (e.g. case) or the grammatical class. It should be noted that due to the sequential nature of the tiers, the error made by a preceding tier cannot be corrected by the following one and in such cases the tagger selects one arbitrary label.

WMBL uses the following word-features: values of the grammatical class, number, gender and case of the surrounding words; lowercased orthographical forms of the surrounding words, if they were popular enough (among 500 most popular words in the training corpus) and binary features indicating if there is a possible agreement in number, gender or case between the word in question and the surrounding words.

During the disambiguation the labels that are compatible with the word in question are supplied by the morphological analyzer. Then at each tier a separate memory is used to retrieve the most similar vectors. The winning grammatical category value (e.g. nominative case) is selected and all the labels provided by the previous tier that are not compatible with the selected value are removed. If that step would yield the label set empty, no action is taken, with assumption that the remaining ambiguity might be removed by the subsequent tiers. If the ambiguity remains until the end of the procedure, one of the remaining labels is arbitrarily selected.

2.2 Pantera

Pantera (Acedański, 2010) is distributed under General Public License (GPL) and is based on the idea of Brill tagger (Brill, 1992). The installation procedure is straightforward, since the tagger is available as a package for many Linux distributions (Ubuntu, Fedora and OpenSuse). It also does not require any external resources, so it can be used directly after installation. What is more it produces output in many formats (e.g. XCES).

The Brill tagger works as follows: during the learning phase it processes the training data using its current knowledge and then, by comparing the results with the reference corpus it induces rules that are used to fix the observed errors. At each iteration the rule that has the largest good to bad modifications margin is selected, the text is tagged once again and the procedure is repeated. A unigram model is used as a starting knowledge.

The modifications implemented in Pantera mainly account for the characteristic features of inflectional languages. The original Brill tagger had very small set of templates used as transformations. It was extended in Pantera and in particular the transformation rules were split into a test (LHS) and an action (RHS) part, allowing for more flexible rule construction. The morphosyntactic labels are disambiguated in several passes covering only one grammatical category at time. The LHS of the rules cover also lexical features, such as prefix and suffix of the word. And the last but not the least, the implementation was simplified and parallelized.

The multipass tagging works as follows – during the learning phase the tagset is converted to a set of tagsets, each covering smaller number of grammatical categories. The training is started with the simplest tagset and the rules are recorded. Then a more feature-rich tagset is used and a new set of rules is discovered. The procedure is repeated until it reaches the original tagset. Then during the tagging the rules recorded at each step are applied separately and the values of already determined grammatical categories are not changed.

The last interesting extension covered the lexical features. The LHS of the rules may check if the word contains particular letter, starts or ends with particular letter sequence and so on. The authors of the tagger reported that the lexical rules improved the tagging accuracy by 1.5 percentage point.

2.3 WCRFT

WCRFT (Radziszewski, 2013) can be treated as a development of the WMBT tagger, since they share the tiered approach. The primary difference is the classifier used to select the labels on each tire – in WCRFT the decision is made using Conditional Random Field (CRF)

http://code.google.com/p/pantera-tagger/
http://nlp.pwr.wroc.pl/redmine/projects/wcrft/

The tagger is distributed under the LGPL license. The installation procedure is similar to WMBL, i.e., it uses similar external libraries (Maca, Corpus2, Morfeusz, etc.) and in many cases this requires manual installation of second-order dependencies. On the other hand the tagging process was simplified, e.g., Maca does not have to be called in a separate step.

Conditional Random Fields is a mathematical model used to estimate the conditional probability of a hidden states assuming given set or sequence of observations. In general they are similar to Hidden Markov Models (HMM) (Rabiner, 1989), with the primary difference being the fact that CRF is a conditional model while HMM is a generative model. In the context of NLP CRF is gaining popularity, since unlike HMM it allows to directly represent distant and forward relations, which are quite common in languages as well it works well with dependencies between the input features, which are also very common.

The design of a CRF for NLP tasks boils down to a selection of a number of characteristic functions which indicate if a given feature holds for the observation in question. The values of these functions with respect to the individual tokens are linearly combined using a fixed set of weights. The weights are determined during the training of the model.

Although the model requires that the features are binary, it is usually easier to model at least some of the features as having multiple values. Since this is a very popular scenario, CRF introduces the notion of function-templates which can be formulated using multi-valued features but are transformed into functions with a binary count/Boolean. As a side effect a large number of functions might be generated. Since the training time is quadratic with respect to the number of possible labels (more than one thousand in Polish), the straightforward application of CRF to the tagging of Polish fails due to practical time and memory constraints.

This is the reason why WCRFT uses tiered approach towards tagging: following WMBT, the set of available label values within each tier is significantly reduced and the CRF model may be practically employed. In fact the primary difference between WMBT and WCRFT lies in the label selection method (k-nearest neighbors in the case of WMBT and highest conditional probability in the case of WCRFT) and the fact that in WMBT the classifiers works token by token, while in WCRFT all labels are determined for the whole sentence at once (with respect to the processed tire).

WCRFT uses the following features: word form of the token, possible values of gender, number and case, agreement between the token and the next token, agreement between three subsequent tokens and capitalization of the word. These features are used to define secondary features, which are dependent on the index relative to the analyzed token and in some cases are used to test two or three subsequent tokens.

2.4 Concraft

Concraft\(^1\) (Waszczyk, 2012) is another tagger utilizing the model of Conditional Random Fields. It is distributed under the BSD two-clause license. The tagger is written in Haskell and comes as a module, that can be downloaded and installed via the Cabal package management tool. Assuming that the Haskell system (including the Cabal manager) is properly installed and configured the installation procedure is very simple and amounts to issuing one command. The tagger is supplemented with a model trained over the NCP corpus which has to be separately downloaded.

On the other hand the documentation of the system is rather minimalistic and amounts to a Readme file. It does not cover any command line options and since the default output of Concraft is a plain text (using very simple tabulation scheme) we have to assume that this system is unable to produce XML as an output. In order to work properly Concraft also requires Maca and Corpus2 tools to perform the segmentation of the input text.

Concraft uses Constraint Conditional Random Fields in order to achieve two goals: the primary, i.e., the disambiguation of morphosyntactic labels and the secondary, i.e., the inference of most probable labels for the unknown words (which are used in constraining the search space during the disambiguation).

It employs second-order linear chain CRF to model the interdependence between the words, their morphosyntactic labels and the previous labels. Since the set of distinct labels contains more than 1000 entries, the model is further simplified by introducing layers: each layer may contain different grammatical categories. As a result the number of distinct labels is reduced. It should be noted however that the layers are not tiers, i.e., they are used in parallel, which allows to model their interdependence. In the development of the model for Polish two layers were used: part-of-speech, case and person in the first layer, and other categories in the second layer.

In order to provide probable labels for the unknown words (i.e. reducing more than 1000 possible labels to a number which is closer to the average 4 labels for the known words) a first-order CRF is used. The feature set covers: lowercase prefixes and suffixes of length 1 and 2, a boolean value indicating if the word is known and a packed shape of the word capturing lower/upper case letters, digits and other symbols. These features together with the label of the previous word are used to estimate the probabilities of the labels, then a fixed number of the most probable labels (10) is provided to the disambiguation phase.

Regarding the features that are used during disambiguation Concraft is very minimalistic. It contains only the lowercase forms of the previous, the current and the next token. In the case of unknown words it also contains lowercase prefixes and suffixes of the word of lengths 1, 2 and 3 and packed shape of the word, together with the information of the first letter case.

3. Evaluation

In order to evaluate the taggers' performance in the context of ASR we followed the following scheme. In the

\(^1\)http://hackage.haskell.org/package/concraft
first step a morphosyntactic n-gram model of Polish was created. The model was refined using Witten-Bell discounting in order to overcome the data sparsity problem (Witten, 1991). In the second step the n-best list of speech recognition acoustic hypotheses received from HTK (Young, 1996) was substituted with the n-best list of sequences of the morphosyntactic tags, corresponding to the words in the hypotheses, provided by each tagger separately. In the next step the probability with respect to the n-gram model of each of the hypotheses was computed. Then a part of the corpus (a tuning set) was used to compute the weight of the language module for each of the taggers. In the last step the recognition hypotheses of each speech signal in the testing set were re-scored according to their combined acoustic model (AM) - language model (LM) probability and the final word-error-rate reduction (WERR) was measured.

The morphosyntactic n-gram model of Polish was computed using the manually tagged 1-million-subcorpus of NCP. The computed model incorporated only selected morphosyntactic features of the words, namely: part-of-speech and value of gender, number and case (if applicable) since we found out that such a reduced feature-set performs much better than the full\(^*\).

We used Witten-Bell discounting, although the Kneser-Ney (Kneser and Ney, 1995) method is reported to perform the best in the case of language modeling for ASR. The reason for that was the relatively small number of distinct labels, namely 734, which excludes the application of Kneser-Ney discounting.

To evaluate the impact of the taggers on ASR we used several speech corpora. The first one (C1), which was used as a tuning set, included 108 sentences spoken by one male voice, without any added noise, but spoken in an office with working computers. It covered political speeches and spoken fragments of song lyrics. The second corpus (C2) consists of 23 samples of one young female professional speaker. These are recordings without noise, made for a film about speech technologies from prepared and checked sentences. The third corpus (C3) consisted of 221 short sentences and commands recorded during various tests of speech and speaker recognition systems at one of the Polish universities with addition of recordings from meetings of the Department Council. This corpus was collected to combine many various voices (one speaker say no more than 6 sentences, often just one or two) and recording devices, often with a natural random noise due to bad acoustic conditions (reverberation of room, voices of other people in a corridor, cars from outside etc.) We used also recordings of Polish LUNA corpus (Marciniak, 2010) which is a corpus of telephone conversations from a call center of Warsaw public transport information. 192 samples of various female voices (C4) and 226 of male voices (C5) were used. These are informal sentences with many questions. The corpus is full of grammar mistakes, very common in natural conversations. The testing corpus consisted of the C2, C3, C4 and C5 corpora.

We also evaluated the speed of the taggers, since this feature is quite important in the case of on-line ASR. We measured separately the start-up time and the processing time. The start-up time was measured as the time required to tag one sentence “Als,” while the processing time was measured for a set of acoustic hypotheses including 900 entries. The loading time was averaged over 5 runs, while the processing time over 10 runs. In the following reports the loading time is subtracted from the processing time.

In all cases the tests were carried out in hot-boot setting, i.e. the linguistic models employed in the tagging were used on the same computer in previous experiments. As a result all files read by a tagger were cached in the operational memory. The computer used to perform the tests had an Intel i7-3537U CPU clocked at 2.0 GHz with 2 cores and 4 hardware threads, 8 GB of RAM and a 256GB SSD drive. The operating system was 64-bit Ubuntu 13.04.

4. Results

Table 1 includes the comparison of the taggers in terms of performance in the context of ASR. The best performing tagger is Concraft, reaching 25.25 percentage points (pp.) WERR on average, while the worst is WMBT with 23.06 pp. WERR. The difference between the best and the words results is not large, but statistically significant. Performing a paired t-test with \( p < 0.05 \) shows that the Concraft tagger is better from both the Pantera and WMBT taggers, but not from the WCRFT tagger. It should be observed that the best performing taggers (Concraft and WCRFT) use the same technique (CRF) and the same training corpus, however their results are slightly different.

Table 2 includes the comparison of the speed of the taggers. The WCRFT tagger has the best loading time – below one third of a second, while WMBT has the worst loading time exceeding 10 seconds. It should be stressed that all taggers were trained on the same corpus (1-million-subcorpus of NCP), so these differences are caused only by the internal representation of the knowledge used by the taggers and the implementation of the loading procedure. When it comes to the tagging time Pantera is definitely the winner, with the tagging time (around 4 seconds) 4 times shorter than the next fastest tagger namely Concraft. Here WMBT is the worst once again with the tagging time exceeding 200 seconds. It is apparent that the speed of the taggers varies significantly and should be strongly considered when choosing the optimal solution for a given settings.

5. Conclusion

The results show that although there is no significant difference between the best performing taggers (Concraft, WCRFT) in terms of accuracy, Concraft is a better choice since its tagging time is much shorter. The results are fairly consistent with those obtained for in-vitro tagging experiments following 10-fold cross validation and show that CRF models are the best choice for tagging implementation for Polish. If someone is mostly concerned with speed, Pantera is definitely the best choice. On the other hand MBL did not prove to be a good choice both in terms of accuracy and speed.

Concraft has also some other advantages, namely the easiness of installation and usage and BSD license, which is very permissive. This is why we will use this tagger in the forthcoming research.

---

\( ^* \)This statement requires experimental justification, but it is out of scope of this paper.
<table>
<thead>
<tr>
<th>Corpus</th>
<th>Tagger</th>
<th>WERR</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>WMBT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2</td>
<td>24,748</td>
<td>0.04</td>
<td></td>
</tr>
<tr>
<td>C3</td>
<td>27,737</td>
<td>0.39</td>
<td></td>
</tr>
<tr>
<td>C4</td>
<td>25,169</td>
<td>0.28</td>
<td></td>
</tr>
<tr>
<td>C5</td>
<td>14,543</td>
<td>0.29</td>
<td></td>
</tr>
<tr>
<td>Avg.</td>
<td>23,06</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Pantera</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2</td>
<td>28,627</td>
<td>0.04</td>
<td></td>
</tr>
<tr>
<td>C3</td>
<td>28,9</td>
<td>0.39</td>
<td></td>
</tr>
<tr>
<td>C4</td>
<td>25,379</td>
<td>0.28</td>
<td></td>
</tr>
<tr>
<td>C5</td>
<td>12,505</td>
<td>0.29</td>
<td></td>
</tr>
<tr>
<td>Avg.</td>
<td>23,13</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>WCRFT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2</td>
<td>28,381</td>
<td>0.04</td>
<td></td>
</tr>
<tr>
<td>C3</td>
<td>30,676</td>
<td>0.39</td>
<td></td>
</tr>
<tr>
<td>C4</td>
<td>26,431</td>
<td>0.28</td>
<td></td>
</tr>
<tr>
<td>C5</td>
<td>13,608</td>
<td>0.29</td>
<td></td>
</tr>
<tr>
<td>Avg.</td>
<td>24,43</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Concraft</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2</td>
<td>26,798</td>
<td>0.04</td>
<td></td>
</tr>
<tr>
<td>C3</td>
<td>30,97</td>
<td>0.39</td>
<td></td>
</tr>
<tr>
<td>C4</td>
<td>28,973</td>
<td>0.28</td>
<td></td>
</tr>
<tr>
<td>C5</td>
<td>13,743</td>
<td>0.29</td>
<td></td>
</tr>
<tr>
<td>Avg.</td>
<td>25,23</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1. Comparison of the performance of the taggers.

<table>
<thead>
<tr>
<th>Load [ms]</th>
<th>Tagging [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>WMBT</td>
<td>10560</td>
</tr>
<tr>
<td>Pantera</td>
<td>2713</td>
</tr>
<tr>
<td>WCRFT</td>
<td>288</td>
</tr>
<tr>
<td>Concraft</td>
<td>9171</td>
</tr>
</tbody>
</table>

Table 2. Comparison of the speed of the taggers.

Acknowledgments
This work was supported by LIDER/37/69/L-3/11/-NCBR/2012 grant.

References


Daelemans, W., Zavrel, J., van der Sloot, K., & van den Bosch, A. (2010). TiMBL: Tilburg Memory-Based Learner.


298