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

Aleksander Smywiński-Pohl[1,2,3(✉)] and Bartosz Ziółko[2,3]

[1] Faculty of Management and Social Communication,
Jagiellonian University, Kraków, Poland
apohllo@o2.pl
[2] Faculty of Computer Science, Electronics and Telecommunication,
AGH University of Science and Technology, Kraków, Poland
[3] Techmo, Kraków, Poland
http://www.techmo.pl

Abstract. In this paper (This is a revised and extended version of the article *A Comparison of Polish Taggers in the Application for Automatic Speech Recognition* that appeared in the *Proceedings of Language and Tools Conference*, Poznan, 2013.) 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 Adama* 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

Z. Vetulani et al. (Eds.): LTC 2013, LNAI 9561, pp. 68–81, 2016.
DOI: 10.1007/978-3-319-43808-5_6

many forms (e.g. *Adam, Adama, Adamowi, Adamem, Adamie, Adamowie,...* are all forms of *Adam*).

These two facts have important implications when building a language model for an ASR system for Polish [29]. The first one makes the generally accepted methods improving language models, namely class-based n-grams [4] 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-sparseness 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 asses 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) [24], 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 [17], Pantera [1], WCRFT [18] and Concraft [24]. These are the most up-to-date, publicly available systems enlisted on the "Computational Linguistics in Poland"[2] web-page (in the section "Language Tools and Resources for Polish") which were developed specifically for Polish. We do not include in the comparison TaKIPI [12] as well as TnT [2] for which there is a Polish model available. Regarding TaKIPI the reason is that it was bound to a specific tagset which is no longer supported, especially in the primary Polish corpus, that is National Corpus of Polish (NCP) [15]. As a result it is no longer developed and its performance is reported [1,17,18,24] to be inferior to all the presented taggers. The reason for TnT is that it uses second-order Markov models, which are also reported to be inferior to all the presented techniques (with respect to tagging of 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

[1] We use the terms *part-of-speech* and *grammatical class* interchangeably in this document, due to the way they are used in the literature regarding Polish tagsets and taggers.

[2] http://clip.ipipan.waw.pl/LRT.

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[3] (Wrocław Memory-Based Tagger) [17] 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 [5], which is a set of Natural Language Processing (NLP) tools employing MBL methods for various language-related problems. Although TiMBL comes with a specific tool designed for tagging, WMBT only uses its general MBL capabilities.

The installation of WMBT is not straightforward and requires manual installation of several other libraries: Maca [17], Corpus2, Morfeusz [26], WCCL [19] and TiMBL. The first library is used for splitting the analyzed text into paragraphs and segments. It also works as a proxy to the morphological analyzer. Corpus2 provides an efficient access to corpora (NCP in particular). Morfeusz is a quite popular library for morphological analysis of Polish words and is used in all the other taggers. WCCL provides a formalism for expressing and transforming various lexical and morphosyntactic features, such as case agreement. It is also used by WCRFT. TiMBL is the already mentioned library providing MBL tools and algorithms.

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). They also have many dependencies so the overall process is pretty tiresome. The most problematic is the requirement for the TiMBL Python wrapper, which is no longer supported by the developers of that library. Compilation and installation errors are not uncommon. What is more, running WMBL on a plain text requires a separate call to Maca, for the input preprocessing. As a final note we should observe, that the tagger is no longer developed by the team, since it was replaced by WCRFT.

The general idea behind MBL-based tagging [6] is as follows: during the training phase, the word occurrences are transformed into feature-vectors which are, together with the correct value of the morphsyntactic label, 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 to 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 [22]. 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

[3] http://nlp.pwr.wroc.pl/redmine/projects/wmbt.

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 features to convert a word occurrence into a feature-vector: 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. All these features are used on all tiers.

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[4] ("Polskiej Akademii Nauk Tager Ekstrahujący Reguły Automatycznie", which means in English "Automatic Rule Extraction Based Tagger of the Polish Academy of Sciences") [1] is distributed under General Public License (GPL) and is based on the idea of Brill tagger [3]. In the past (2013) the installation procedure was straightforward, since the tagger was available as a package for many Linux distributions (Ubuntu, Fedora and OpenSuse). But these packages are no longer available for the most up-to-date distributions, so it has to be complied by the user. Fortunately it also does not require any external resources, so the procedure is rather straight-forward. We should also note, that the code of the tagger is available at `code.google.com` – a service that is no longer in operation. Thus we might conclude that the system is no longer developed.

The mode of operation of the tagger is similar to the idea used in the Brill tagger, i.e. during the learning phase the tagger processes the learning material 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 label statistic is used as an initial model.

[4] http://code.google.com/p/pantera-tagger/.

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. The set 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 were disambiguated in several passes covering only one selected grammatical category. The LHS of the rules might cover lexical features, such as prefix and suffix of the word. And the last but not the least, the implementation was simplified and parallelized.

The generalization of the transformation rules was introduced firstly in order to capture complex conditions that could be useful in Polish (e.g. for capturing case or gender agreement, which can not be expressed by the rules devised in Brill tagger), and secondly in order to allow for assigning the whole morphosyntactic label at once as well as only a part of the label e.g. a value of particular grammatical category or the grammatical class.

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 most simplified tagset and the rules are recorded. Then a more feature-rich tagset is used and 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 % point.

2.3 WCRFT

WCRFT[5] (Wrocław Conditional Random Field Tagger) [18] 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) [9,21] linear-chain classifier.

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 as 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) [16], 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,

[5] http://nlp.pwr.wroc.pl/redmine/projects/wcrft/.

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 counter-domain. 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 problem of Polish morphosyntactic tagging fails due to practical time and memory constraints.

This is the reason why WCRFT uses tiered approach towards tagging: by following the same label selection scheme as 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 lays 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 provided for the whole sentence at once (with respect to the processed tire).

Regarding the features that were used as the input for the model, WCRFT uses: 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 primary 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[6] (Constrained Conditional Random Fields Tagger) [24] 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

[6] http://hackage.haskell.org/package/concraft.

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 have implemented two evaluation scenarios. In both cases the morphosyntactic LM was incorporated into the results of speech recognition according to the following equation:

$$P(h_i) = P(h_i)_{wLM}^{\alpha} * P(h_i)_{mLM}^{\beta} * P(h_i)_{AM}^{1-\alpha-\beta} , \tag{1}$$

where:

- $P(h_i)$ – probability of the i-th recognition hypothesis,
- wLM – word LM,
- mLM – morphosyntactic LM,
- AM – acoustic model (AM),
- α, β – weights of the respective LMs.

Thus the probabilities of a given hypothesis according to the different models were combined using a set of weights optimized on a *tuning* corpus.

The primary difference between the scenarios was the fact, that the first scenario lacked the word LM (i.e. the α parameter was set to 0). The other differences regarded different systems used to build the AM and different corpora used to evaluate the results.

In both cases in the first step a morphosyntactic n-gram model of Polish was built with the help of SRI LM package [20]. One-million subcorpus of NCP [15] was used to compute the counts of the specific tags combinations. The probability of a given set of morphosyntactic tags, given the set of previous morphosyntactic tags was estimated as:

$$P(h_i)_{mLM} = \prod_{w_j \in h_i} P(V(w_j)|V(w_{j-N+1})...V(w_{j-1})) \,, \tag{2}$$

where:

– $V(w_j)$ – the set of morphosyntactic tags attached to word w_j,
– N – n-gram order,
– $w_{j-N+1}...w_{j-1}$ – N–1 words preceding the word w_j.

To reduce the data sparsity the set of morphosyntactic tags attached to each word was filtered only to include grammatical class and values of gender, number and case (if applicable for a given class). Then the model was refined using Witten-Bell (WB) discounting [25]. We have used WB discounting, although the Kneser-Ney [8] 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 excluded the application of Kneser-Ney discounting.

3.1 HTK + mLM

The following steps depended on the evaluation scenario. In the first scenario the n-best list of speech recognition acoustic hypotheses was produced by HTK [27]. Each of the compared taggers was then used to convert the sequence of words into sequence of tags. These tags were filtered in order to keep only the tags present in the mLM. Then SRI LM was used to assign the probabilities to each sequence of tags according to the mLM (we have used 3-order LM in this scenario). Then the tuning corpus was used to compute the β weight implementing a grid search strategy. That parameter was defined independently for each of the taggers. In the last step the recognition hypotheses of each speech signal in the testing set were re-scored according the Eq. 1. The procedure is depicted on Fig. 1.

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 31 samples of one young female professional speaker. These were recordings without noise, made for a film

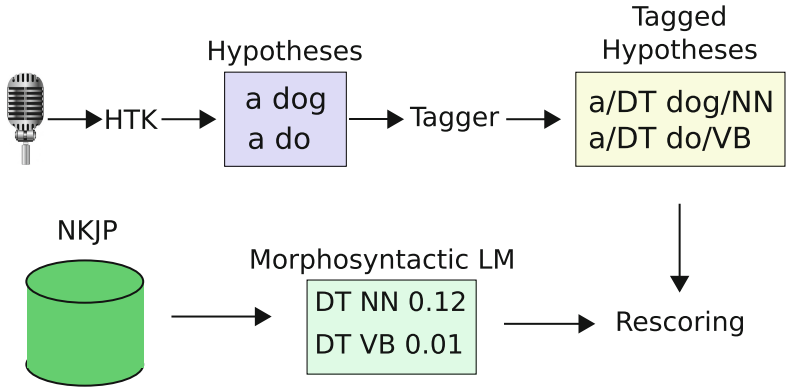

Fig. 1. Rescoring of the hypotheses according to the morphosyntactic LM.

about speech technologies from prepared and checked sentences. The third corpus (C3) consisted of 281 short sentences and commands recorded during various tests of speech and speaker recognition systems at AGH University of Science and Technology 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 LUNA corpus [10] which is a corpus of telephone conversations from a call center of Warsaw public transport information. 192 samples of various female voices (C4) and 228 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.

3.2 Kaldi + mLM + wLM

In the second scenario we have used a more recent automatic speech recognition system called Kaldi [14]. The AM was a triphone HMM Gaussian mixture model [7]. It was trained on a large dataset of recordings collected by AGH [28] and Techmo and selection of recordings from the Global Phone acoustic database [23]. The processing pipeline also included a word-level trigram LM (wLM), trained on a subset (containing approx. 600 million tokens) of NCP. That model was directly combined with the AM, since both models were expressed as weighted finite state transducers [11]. The weight of the wLM was determined in separate set of experiments. As a result the output of the system was similar to that of HTK, with the exception, that also the wLM was taken into account in the probability assigned to the hypotheses.

The evaluation was performed solely on the Global Phone corpus, and included a subset of speakers (4 women and 4 men). The tuning set included

(randomly selected) 10 % of all recording, totaling in 249 entries, while the testing set included 2240 entries. In each case the recordings of the speaker that was tested were excluded from the training set, thus the system was tested in a speaker-independent fashion.

The mLM was used to rescore the hypotheses in the same manner as in the previous scenario, with the exception, that the α parameter was not 0 and the order of the LM was 5. Yet the β parameter was optimized after the α parameter was determined, thus it followed the same grid-search strategy. Moreover the probability of hypotheses produced by Kaldi already included the wLM component. The reason for that was the implemented beam-search strategy, that takes into account both AM and wLM during the search in the hypotheses space.

3.3 Tagging Speed

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 "Ala.", 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 14.04 LTS.

4 Results

Table 1 includes the comparison of the taggers in terms of performance in the first scenario. The best performing tagger is Concraft, reaching 25.2 % points (pp.) WERR on average, while the worst is WMBT with 23.2 pp. WERR. The difference between the best and the worst results is not large, but statistically significant. Performing a paired Student 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 results of the comparison when both the word-level and morphosyntactic LMs were applied. Here both the absolute results and their differences are much smaller (2 orders of magnitude in case of WERR)[7]. Moreover we have not observed any statistically significant differences between the

[7] We have not included the results for WMBT since it was impossible to obtain its results when these tests were performed. Moreover its behaviour was the worst in all the other tests, so we have not expected to see any improvement.

Table 1. Comparison of the performance of the taggers in terms of Word Error Rate Reduction [WERR] when only morphosyntactic LM was employed.

Corpus	Weight	WMBT	Pantera	WCRFT	Concraft
C2	0.04	24.7	**28.6**	28.4	26.8
C3	0.39	27.7	28.9	30.7	**31.0**
C4	0.28	25.2	25.4	26.4	**29.0**
C5	0.29	**14.5**	12.5	13.6	13.7
Avg.		23.1	23.1	24.4	**25.2**

taggers. The probable reason for that result is the fact, that the wLM already included important interdependencies between the grammatical classes and categories between the words and the observed improvement was so small. The other explanation could be based on the fact, that the α parameter was optimized independently of the β parameter, thus we have reached a local maximum. Yet a scenario when both parameters are optimized at once is much more implementationally demanding and was out of scope of this paper.

Table 2. Comparison of the performance of the taggers in terms of WERR when both word and morphosyntactic LMs were employed

Tagger	WERR
Pantera	0.28
WCRFT	**0.36**
Concraft	0.31

Table 3 includes the comparison of the speed of the taggers. The WCRFT tagger has the best loading time – below one fifth of a second, while WMBT has the worst loading time exceeding 10 s. 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 3.5 s) 2 times shorter than the next fastest tagger namely WCRFT. Here WMBT is the worst once again with the tagging time exceeding 200 s. 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.

The table also includes the information of the version of taggers. Only in one case (Concraft), the version was given explicitly. In the other cases we provided the Git or SVN revisions of the particular versions that were used in the test. When comparing with our previous test [13] we might observe that only two taggers are actively developed (WCRFT and Concraft). Both of them made

Table 3. Comparison of the speed of the taggers.

Tagger	Version	Load [ms]	Tagging [ms]
WMBT	`0d67980`	`10560`	`186650`
Pantera	`r156`	`2778`	**`3564`**
WCRFT	`5fba260`	**`194`**	`8202`
Concraft	`0.7.4`	`9207`	`10793`

significant improvements in the tagging time (in case of WCRFT it was 4-fold). We should also note, that the loading time of Concraft can be reduced to 0, since the system implements a client-server architecture and the model might be preloaded before the tagging is performed.

5 Conclusion

The general results of the taggers comparison are as follows: both of the actively maintained taggers, i.e. WCRFT and Concraft offer similar results both in terms of accuracy of the tagging and the tagging speed. Concraft installation is simple, assuming you have Haskell and Cabal installed. WCRFT installation is more demanding, since it requires more dependencies to be installed by the user. Pantera offers the highest speed of tagging, but WCRFT is caching up (2 times longer tagging time at present). WMBL does not offer any improvement neither in tagging accuracy nor in speed – this is probably the primary reason it is no longer developed.

Regarding the application of morphosyntactic LMs in ASR: a sole mLM offers significant recognition improvement, compared to the output produced by HTK. Yet if a word-level LM is involved, the improvement is negligible and probably is not worth the extended recognition time.

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

References

1. Acedański, S.: A morphosyntactic brill tagger for inflectional languages. In: Loftsson, H., Rögnvaldsson, E., Helgadóttir, S. (eds.) IceTAL 2010. LNCS, vol. 6233, pp. 3–14. Springer, Heidelberg (2010)
2. Brants, T.: TnT: a statistical part-of-speech tagger. In: Proceedings of the Sixth Conference on Applied Natural Language Processing, pp. 224–231. Association for Computational Linguistics (2000)
3. Brill, E.: A simple rule-based part of speech tagger. In: Proceedings of the Workshop on Speech and Natural Language, pp. 112–116. Association for Computational Linguistics (1992)

4. Brown, P.F., Desouza, P.V., Mercer, R.L., Pietra, V.J.D., Lai, J.C.: Class-based n-gram models of natural language. Comput. Linguist. **18**(4), 467–479 (1992)
5. Daelemans, W., Zavrel, J., van der Sloot, K., van den Bosch, A.: TiMBL: Tilburg Memory-Based Learner (2010)
6. Daelemans, W., Van den Bosch, A.: Memory-Based Language Processing. Cambridge University Press, New York (2005)
7. Gauvain, J.L., Lee, C.H.: Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains. IEEE Trans. Speech Audio Process. **2**(2), 291–298 (1994)
8. Kneser, R., Ney, H.: Improved backing-off for m-gram language modeling. In: International Conference on Acoustics, Speech, and Signal Processing, vol. 1, pp. 181–184. IEEE (1995)
9. Lafferty, J., McCallum, A., Pereira, F.C.: Conditional random fields: probabilistic models for segmenting and labeling sequence data (2001)
10. Marciniak, M.: Anotowany korpus dialogów telefonicznych. Akademicka Oficyna Wydawnicza EXIT, Warszawa (2011)
11. Mohri, M., Pereira, F., Riley, M.: Weighted finite-state transducers in speech recognition. Comput. Speech Lang. **16**(1), 69–88 (2002)
12. Piasecki, M.: Polish tagger TaKIPI: Rule based construction and optimisation. Task Q. **11**(1–2), 151–167 (2007)
13. Pohl, A., Ziółko, B.: A comparison of polish taggers in the application for automatic speech recognition. In: Proceedings of the 6th Language & Technology Conference, pp. 294–298 (2013)
14. Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlíček, P., Qian, Y., Schwarz, P., et al.: The kaldi speech recognition toolkit. In: Proceedings of Automatic Speech Recognition and Understanding (2011)
15. Przepiórkowski, A., Bańko, M., Górski, R.L., Lewandowska-Tomaszczyk, B.: Narodowy Korpus Języka Polskiego. Wydawnictwo Naukowe PWN, Warsaw (2012)
16. Rabiner, L.R.: A tutorial on hidden Markov models and selected applications in speech recognition. Proc. IEEE **77**(2), 257–286 (1989)
17. Radziszewski, A., Śniatowski, T.: A memory-based tagger for Polish. In: Proceedings of the 5th Language & Technology Conference, Poznań, pp. 29–36 (2011)
18. Radziszewski, A.: A tiered CRF tagger for Polish. In: Bembenik, R., Skonieczny, Ł., Rybiński, H., Kryszkiewicz, M., Niezgódka, M. (eds.) Intelligent Tools for Building a Scientific Information Platform. SCI, vol. 467, pp. 215–230. Springer, Heidelberg (2013)
19. Radziszewski, A., Wardyński, A., Śniatowski, T.: WCCL: a morpho-syntactic feature toolkit. In: Habernal, I., Matoušek, V. (eds.) TSD 2011. LNCS, vol. 6836, pp. 434–441. Springer, Heidelberg (2011)
20. Stolcke, A.: SRILM-an extensible language modeling toolkit. In: Proceedings of the International Conference on Spoken Language Processing, vol. 2, pp. 901–904 (2002)
21. Sutton, C., McCallum, A.: An introduction to conditional random fields for relational learning. In: Introduction to Statistical Relational Learning, pp. 93–128 (2006)
22. Tufis, D.: Tiered tagging and combined language models classifiers. In: Matoušek, V., Mautner, P., Ocelíková, J., Sojka, P. (eds.) TSD 1999. LNCS (LNAI), vol. 1692, pp. 28–33. Springer, Heidelberg (1999)

23. Vu, N.T., Kraus, F., Schultz, T.: Multilingual a-stabil: a new confidence score for multilingual unsupervised training. In: 2010 IEEE Spoken Language Technology Workshop (SLT), pp. 183–188. IEEE (2010)
24. Waszczuk, J.: Harnessing the CRF complexity with domain-specific constraints. The case of morphosyntactic tagging of a highly inflected language. In: Kay, M., Boitet, C. (eds.) Proceedings of COLING, pp. 2789–2804 (2012)
25. Witten, I., Bell, T.: The zero-frequency problem: estimating the probabilities of novel events in adaptive text compression. IEEE Trans. Inf. Theory **37**(4), 1085–1094 (1991)
26. Woliński, M.: Morfeusz—a practical tool for the morphological analysis of Polish. In: Kłopotek, M.A., Wierzchoń, S.T., Trojanowski, K. (eds.) Intelligent Information Processing and Web Mining. Advances in Soft Computing, vol. 35, pp. 511–520. Springer, Heidelberg (2003)
27. Young, S., Evermann, G., Gales, M., Hain, T., Kershaw, D., Moore, G., Odell, J., Ollason, D., Povey, D., Valtchev, V., Woodland, P.: HTK Book. Cambridge University Engineering Department, UK (2005)
28. Żelasko, P., Ziółko, B., Jadczyk, T., Skurzok, D.: AGH corpus of Polish speech. In: Language Resources and Evaluation, pp. 1–17 (2015)
29. Ziółko, B., Ziółko, M.: Przetwarzanie mowy. Wydawnictwo AGH, Kraków (2011)