KRNNT: Polish Recurrent Neural Network Tagger

Krzysztof Wróbel

Jagiellonian University
Kraków, Poland
krzysztof@wrobel.pro

Abstract

The article presents a state-of-the-art complete part-of-speech tagger for Polish which uses recurrent neural networks. The networks allow accessing the full left and right context of a sentence in comparison to a context window. The tagger uses an external morphological analyzer. In comparison to the best Polish taggers, it does not use word form as a feature for the classifier, there is no separate classifier for unknown words, and predictions are not limited to tags provided by a morphological analyzer. The accuracy is higher — it achieves 28% error reduction and 7% points higher accuracy for unknown words. The tagger also might work faster than others by utilizing GPU. The tagger participated in PolEval POS Tagging competition and won task B and task C.

1. Introduction

Part-of-speech taggers assign part-of-speech tags to each word (token) in a sentence. For inflectional languages, like Polish or Czech tagger, has to recognize values of morphological categories such as gender, number, and case. The categories highly increase the number of tags up to a thousand.

Recurrent neural networks have not been applied in natural language processing for Polish yet, despite they are usually used in state-of-the-art systems for other languages (Huang et al., 2015a; Wang and Nyberg, 2015; Graves and Jaitly, 2014; Graves et al., 2013).

Czech is also a Slavic language. The best Czech taggers achieved the accuracy above 95%. Prague Dependency Treebank (PDT) is used as a training and test data. The main difference between PDT and National Corpus of Polish (NCP), which is the primary source of tagging data for Polish, is that PDT is not well-balanced and contains only articles from newspapers and journals. On the contrary, transcriptions of spoken conversations and user generated content from web forums are included in NCP. Both corpora were manually annotated by 2 annotators and 1 person who was resolving disagreements. (Kobyliński and Kieraś, 2016) obtained scores higher by 0.75 percentage point by training and testing only on newspaper subcorpus.

This work presents part-of-speech tagger for Polish using bidirectional recurrent networks (named KRNNT). Source code and trained models are available at: `https://github.com/kwrobel-nlp/krnnt`.

In the next section training dataset and tagset are described. Also, it summarizes the best publicly available Polish taggers. Section 3. formulates recurrent neural networks and its bidirectional extension. Main section 4. describes all modules of the tagger and the training parameters. Evaluation is described in section 5.. Section 6. presents results of PolEval contest. Last section 7. presents conclusions and future works.

[0]This research was supported in part by PLGrid Infrastructure.

Table 1: National Corpus of Polish statistics.

Paragraphs	18484
Sentences	85663
Tokens	1215513
Unique tokens	143478
Average number of tokens in a sentence	14.19
Unique tags	926
Number of tokens with the same tag	
Average	1312.65
Standard deviation	8389.12
Minimal number	1
Maximal number	223499

2. Polish Tagging

2.1. Dataset

The largest publicly available dataset for Polish is a manually annotated subcorpus of the National Corpus of Polish (NCP) containing above 1 million tokens. The corpus is balanced with respect to genres and subjects — it includes newspaper articles, books, transcriptions of spoken conversations, and user generated content from web forums.

Table 1 presents statistics of NCP. Assuming full tags as labels in a multi-class classification problem, they are very unbalanced.

The NCP tagset consists of 35 grammatical classes, each having a set of grammatical categories. The number of all possible tags (grammatical classes with unique values of grammatical categories) is about 4000, but texts in NCP represent 926 tag variants.

E.g. Polish adjectives have 2 numbers (singular, plural), 7 cases (nominative, genitive, dative, accusative, instrumental, locative, vocative), 5 genders (human masculine, animate masculine, inanimate masculine, feminine, neuter), and 3 degrees (positive, comparative, superlative) — 210 variants in total.

Morphosyntactic tags are represented as a sequence of grammatical class and values of grammatical categories, e.g. adj:sg:nom:m1:pos, where adj is adjective, sg is singular number, nom is nominative case, m1 is masculine gender and pos is positive degree.

2.2. Polish Taggers

The best Polish POS taggers that are publicly available include: Concraft (Waszczuk, 2012), WCRFT (Radziszewski, 2013), OpenNLP (Kobyliński and Kieraś, 2016), WMBT (Radziszewski and Śniatowski, 2011; Radziszewski and Acedański, 2012), Pantera (Acedański, 2010), TaKIPI (Piasecki, 2007). Comparisons of Polish taggers were presented in (Kuta et al., 2012; Pohl and Ziółko, 2013; Kobyliński and Kieraś, 2016)

So far Concraft has been the best Polilsh tagger. It uses an extended version of CRF algorithm (Lafferty et al., 2001) to tackle a high number of labels in an efficient way by restricting space of solutions to the set of tags defined in a morphosyntactic dictionary. For unknown words, morphosyntactic guessing is employed as a separate classifier.

WMBT is a tiered memory-based tagger. Each tier is assigned to a grammatical class or a category. A separate classifier is trained for known and unknown tokens. An algorithm used for each tier is kNN.

WCRFT is similar to WMBT, but kNN algorithm is replaced with CRF. Unknown words are pre-processed by appending potential tags based on analysis of a training data. Therefore, only one classifier is used while tagging.

Pantera uses rule induction algorithm driven by a modified version of Brill's transformation-based learning algorithm (Brill, 1992). It uses two tiers in the process of tagging and operates on parts of labels, i.e. grammatical class and categories.

TaKIPI employs C4.5 decision tree algorithm. About 200 classes of ambiguity were defined and for each one the classifier was trained. The tagger uses also handwritten rules.

Apache OpenNLP library is a free implementation of NLP algorithms. Algorithm for POS tagging employs perceptron. No tiers were used and all tags are on the output of the neural network. The main difference to earlier mentioned taggers is that OpenNLP tagger does not use morphological analyzer having a token as the only input.

3. Recurrent Neural Network

A recurrent neural network is an extension to feedforward neural network, which is able to handle a variable-length sequence inputs. The RNN has a hidden state that is updated at each time step, therefore it has information about the left context of a sequence. The RNN shares parameters across all steps. The default behavior of RNN is to provide output for each step. However, RNNs are difficult to train to capture long-term dependencies, because the gradients tend to vanish or explode.

Long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) was developed to alleviate problems of raw RNNs. The LSTM introduces an additional memory cell and gates. Output, forget and input gates are responsible for modulating the amount of memory cell used to calculate the output of the LSTM and a new value of memory cell.

The output of the LSTM h_t at time t is:

$$h_t = o_t \tanh(c_t),$$

where o_t is output gate and c_t is the memory cell:

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + V_o c_{t-1}),$$

$$c_t = f_t c_{t-1} + i_t \tanh(W_c x_t + U_c h_{t-1}).$$

σ is the logistic sigmoid function. Forget and input gates are computed as follows:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + V_f c_{t-1}),$$

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + V_i c_{t-1}).$$

Gated Recurrent Unit (GRU) is similar to LSTM. It does not have the memory cell, but utilizes update and reset gates. It has fewer parameters than LSTM and therefore it's training is faster.

The output of the GRU is:

$$h_t = (1 - z_t)h_{t-1} + z_t \tanh(W x_t + U(r_t h_{t-1})),$$

where z_t is update gate and r_t is reset gate.

$$z_t = \sigma(W_z x_t + U_z h_{t-1})$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1})$$

A simple extension to the RNN is a bidirectional recurrent neural network (BDRNN). It contains two RNNs, the first working forward, the second working backward. Outputs at each step are merged (i.e. concatenated). Forward RNN remembers the left context and backward RNN remembers the right context. This feature solves a problem of providing full context to a token in POS tagging. Many taggers are using context window to incorporate nearest neighbors of a token, but it is usually of limited size.

4. Polish RNN tagger

4.1. Tokenization and Morphological Analysis

Text preprocessing is performed by external tools. Firstly, a text is segmented into sentences and into tokens using Toki (Radziszewski and Śniatowski, 2011). Secondly, each token in a sentence is analyzed by morphological analyzer SGJP. Maca (Radziszewski and Śniatowski, 2011) integrates both tools and is used in this tagger (Concraft also uses Maca).

4.2. Morphological Guesser

A morphological guesser is a system that predicts potential tags for a word form. In practice, it is applied only for unknown words which are not present in the dictionary. In this work, a morphological guesser as a separate step is omitted — unknown words have no features associated with potential tags. To address this issue, features, based on the word form were added: first and last three letters of a word form. WCRFT and Concraft use a separate classifier that replaces morphological analysis for unknown words.

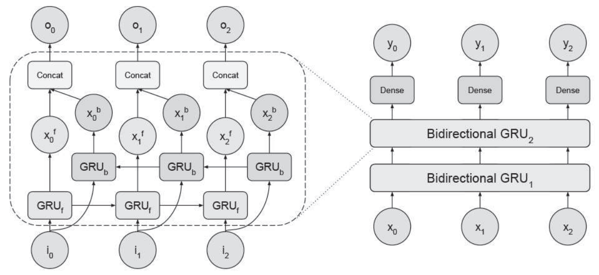

Figure 1: A neural network used in this work. It allows variable-length sequences on input, but the figure shows the network for a sequence of length 3 $\{x_0, x_1, x_2\}$. Arrows represent directions of computation. A bidirectional GRU layer is presented on the left. Operation *Concat* concatenates outputs from two GRU layers into one matrix. Right diagram shows the full network with two bidirectional layers and a dense layer with shared weights on the output.

4.3. Morphological Disambiguator

Morphological disambiguator assigns one tag for each token. The decision is based on features (observations) of tokens. In this work, classification is performed by a recurrent neural network, so there is no need for creating context window which limits the length of a potential dependency. By using a bidirectional recurrent neural network, for each token, a classifier has information about full left and full right context. It creates an advantage in comparison to a window approach used in CRF. Both WCRFT and Concraft use the context of length 2.

4.4. Lemmatization

Lemmatization is a task closely related to morphological disambiguation. Concraft and WCRFT do not tackle this problem. From a dictionary, they choose all lemmas associated with the disambiguated tag. Unknown words are not lemmatized — for the higher scores, token form is set as the lemma.

In this work, a simple extension is provided. Training data is analyzed by counting lemmas for each pair of a token and disambiguated tag. During tagging, the tagger starts with prediction of a tag and then chooses the most frequent lemma for the pair: token and disambiguated tag.

4.5. Network Architecture

The network has two bidirectional GRUs. Dropout (fraction of units to drop during training) is applied to the linear transformation of the recurrent state. Dropout is also applied to the results from the second bidirectional GRU and processed by dense layer with the same shared weights for each step. The network is presented in Fig. 1. All features of words are encoded as one-hot vectors.

4.6. Features

The best Polish taggers use word form as a feature. In Polish, there are more than 3.8 million unique word forms. Treating them as one-hot vector would create too many inputs for a neural network. Generally, this problem of dimensionality reduction is being solved by using word embeddings. However, in this work word embeddings are not used — initial attempts have given worse results. Addition of word embeddings as an additional input to the model prevented learning from gaining above 90% of accuracy.

Table 2: Example of features generation for the word *obrazki* (images).

token	obrazki
tags4	1subst:nom, 2subst:pl:m3, 1subst:acc, 1subst:voc
tags5	pl:nom:m3, nom, pl:acc:m3, acc, pl:voc:m3, voc
shape	l
cases	islower
interps	
qubs	
3letters	P0o, P1b, P2r, S1i, S2k, S3z

Eight sets of features were tested (the number of unique features is given in parentheses):

- **tags4** (388) — each tag is divided into two parts: grammatical class + case or person, and grammatical class + rest of grammatical categories (Acedański, 2010),
- **tags5** (90) — case, and concatenation of number, case and gerund,
- **shape** (76) — collapsed shape of token - upper case letters are represented as *u*, lower case letters as *l*, digits as *d*, other characters as *x* (e.g. *Wrobel2017* gives *ulllllldddd* and after collapsing *uld*),
- **cases** (5) — information whether word form is all lower cased, upper cased, capitalized, or a number,
- **interps** (55) — individual punctuation marks,
- **qubs** (226) — set of all particle adverbs,
- **3letters** (276) — first and last three letters of word form; this feature have source from morphological guesser in Concraft, in which prefixes and suffixes are generated,
- **separator** (2) — information about space before token.

Table 2 presents features generated for the word *obrazki*.

4.7. Output Classes

To reduce the number of outputs WCRFT classifies each grammatical class and category separately while Concraft divides tags into two sets (the same as the feature **tags4**).

In comparison to other Polish taggers, KRNNT has undivided tags on output. A drawback of this approach is that tags not occurring in training data can not be predicted even if the morphological analyzer has information about possible correct tags.

4.8. Training

10% of training data is used as a validation set for early stopping. Early stopping criterion is checked after process-

Table 3: Results of taggers in 10-fold cross-validation scheme using NCP.

Tagger	Acc_{lower}	Acc_{upper}	Acc^{K}_{lower}	Acc^{U}_{lower}
OpenNLP	87.24%		88.02%	62.05%
Pantera	88.95%		91.22%	15.19%
WMBT	90.33%		91.26%	60.25%
WCRFT	90.76%		91.92%	53.18%
Concraft	91.19%	91.53%	92.07%	60.64%
KRNNT	**93.72%**	94.05%	**94.43%**	**69.03%**

ing every 10,000 sentences with patience 10. The maximal number of epochs is 150. A loss (objective) function is categorical loss entropy. The last layer has softmax activation function.

Training is performed using Nadam optimizer, because it was proven to be effective for recurrent neural networks (Dozat, 2016; Sutskever et al., 2013). It is a combination of two algorithms: RMSProp and Nesterov momentum.

Training was performed on GPU NVIDIA Tesla K40 XL and took about 3 hours (on NCP).

5. Evaluation

Many experiments testing different sets of features and neural network architectures were conducted (over 40,000 hours on GPU). Taggers were assessed in terms of accuracy and speed of tagging. National Corpus of Polish with 10-fold cross-validation was chosen as a training corpus. Sentences from one paragraph are always in the same fold. Sentences incorrectly segmented by Maca (3.41% of NCP) are skipped during training (for simplicity).

Evaluation is performed with the whole pipeline including segmentation, morphological analysis and morphological disambiguation as proposed in (Radziszewski and Acedański, 2012).

The main metric is accuracy lower bound (Acc_{lower}). It penalizes all segmentation errors and is calculated as a percentage of all tokens that match tagger segmentation with correct tag. Additional metric accuracy upper bound (Acc_{upper}) treats segmentation errors as correctly tagged. It shows potential accuracy for a perfect tokenizer.

Two additional metrics are also provided: accuracy lower bound for known (Acc^{K}_{lower}) and unknown words (Acc^{U}_{lower}).

Comparison of results for each tagger is presented in table 3. Scores for OpenNLP, Pantera, WMBT, and WCRFT originate from (Kobyliński and Kieraś, 2016). Evaluations for all taggers were performed on NCP and with 10-fold cross-validation scheme.

KRNNT significantly surpasses scores of other taggers, the error is reduced by 28% in comparison to Concraft. The accuracy of tagging unknown words is 7 percentage points higher than in OpenNLP. Simple voting strategy over 10 models trained on the same data, but with different random initialization, increase accuracy lower bound up to 94.30% (tested without cross-validation).

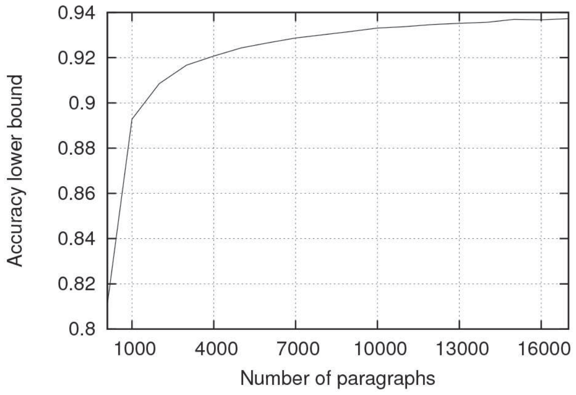

Figure 2: Accuracy lower bound in function of a number of training paragraphs.

Table 4: Time of tagging NCP measured in seconds. In the case of KRNNT, a percentage of tagging time when GPU is waiting for data is given in parentheses. KRNNT is also tested for corpus sorted by sentence (SS) length.

Tagger	1 core	2 cores	4 cores
Concraft	376	199	140
KRNNT GPU	556 (60%)	297 (50%)	223 (40%)
KRNNT GPU SS	423 (74%)	231 (64%)	141 (44%)

For 0.55% tokens, the predicted tag was not in the set returned by the morphological analyzer. Despite that, KRNNT correctly assigns tags in 78.89%.

NCP has 2.81% unknown words according to Maca.

(Kobyliński and Kieraś, 2016) developed ensemble of the first 5 taggers from table 3. The best voting scheme achieves the accuracy lower bound slightly above 92%.

Figure 2 shows accuracy lower bound related to a number of paragraphs, that were used to train the tagger. More training data is needed to determine whether the classifier is saturated.

Table 4 shows tagging time in seconds of whole NCP including the start of a tagger. NCP was manually distributed for separate processes of Concraft because Concraft does not utilize more cores. KRNNT executes longer than Concraft. The analysis showed that the most time-consuming is to generate the features, execute Maca and parse its output. Distribution of these tasks to other cores decreases tagging time. Values in parentheses give the percentage of total time spent by GPU waiting for data. Implementation of the tagger in a statically typed language should improve performance.

Processing sentences sorted by a number of words improves tagging time by 22%–37% because computations on GPU are performed in batches and all sentences need to be padded to the longest sentence in the batch. For sorted sentences, padding is minimal.

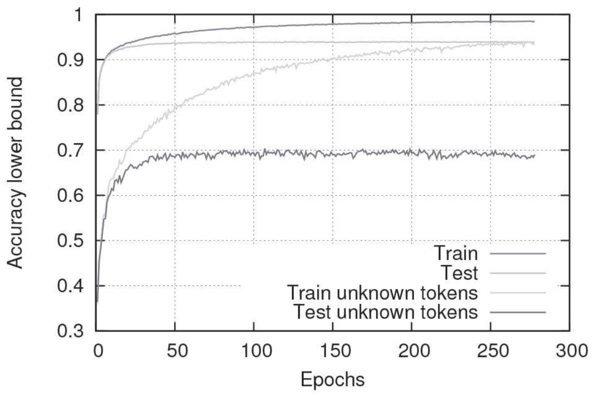

Figure 3: Accuracy lower bound in function of a number of epochs.

Figure 3 shows accuracy lower bound for training and test data related to a number of epochs. The neural network can not memorize the training data (98.3%). Most likely this is caused by an insufficient representation of the input (word form is not a feature). The accuracy for test data is not raising after about 100 epochs and the model is not overfitting.

Manual analysis of 100 errors of the tagger showed that 6% relate to errors in manual annotation of NCP, e.g. *to były ostatnie słowa, jakie wypowiedział* (these were the last words that he had spoken) - the word *jakie* (that) is manually annotated as nominative, KRNNT tags it as accusative, which is correct. 9% of errors could only be avoided with the analysis of the whole paragraph, e.g. *Na baliku [...] bawiło się około 100 dzieci. Znaleźli się wśród nich* (About 100 children played at the ball. There were also among them) - the gender of word *nich* (them) is dependent on reference to *dzieci* (children). Dependencies longer than 5 words occur in 11% of errors. Including semantics and valency information could potentially reduce errors by 15%, e.g. *Władze miasta [...] szukają inwestora* (City authorities are looking for an investor) - the verb *szukać* (look for) takes objects with genitive case in this context (KRNNT assigns accusative case).

6. PolEval: POS Tagging

PolEval is a Polish version of SemEval — a contest for natural language processing tools. KRNNT participated in a task of morphosyntactic tagging (Kobyliński and Ogrodniczuk, 2017). It involves 3 subtasks: morphosyntactic disambiguation and guessing (subtask A), lemmatization (subtask B), and POS tagging (subtask C). Subtasks A and B are tested on gold segmented data, therefore systems do not need to perform tokenization and morphological analysis. Subtask C is tested on raw text and requires whole text processing pipeline. The training data is NCP. For subtasks A and B, the model was trained without morphological reanalysis. Therefore segmentation errors do not occur. Organizers prepared different testing corpus, they annotated over 1626 sentences for subtasks A and B and 1675 sentences for subtask C. Average number of tokens in sentences is around 16.5 — more than in NCP.

Table 5: Results of best performing systems of PolEval subtask A. Accuracy is calculated separately and jointly for known and unknown words.

Tagger	Acc^K	Acc^U	Acc
Toygger	**95.24%**	**65.47%**	**94.63%**
KRNNT	94.49%	61.18%	93.80%
NeuroParser	94.21%	64.94%	93.61%

Table 6: Results of PolEval subtask B. Accuracy of lemmatization is calculated separately and jointly for known and unknown words.

Tagger	Acc^K	Acc^U	Acc
KRNNT	**98.19%**	80.86%	**97.84%**
NeuroParser	97.37%	**84.62%**	97.11%

The best submission for task A (table 5) was also prepared using bidirectional neural network. The main difference to KRNNT is a utilization of word embeddings and reduced output to separate grammatical classes and categories. KRNNT was placed second in the ranking.

Despite simple lemmatization module in KRNNT, it won subtask B (table 6). NeuroParser has better accuracy in lemmatization of unknown words by 4 percentage points, so there is a room for improvement.

Subtask C was also won by KRNNT (table 7). However, lemmatization performed by NeuroParser was also better. Third place is taken by MorphoDiTaPL (Walentynowicz, 2017) — the framework achieving state-of-the-art results in Czech.

7. Conclusion

This work presented Polish morphosyntactic tagger KRNNT. It achieves better accuracy than other publicly available taggers.

PolEval showed that better results could be achieved using bidirectional neural networks.

Lemmatization for unknown words may be improved by a separate neural network using a sequence to sequence architecture (Cho et al., 2014). Bigger dictionary of named entities should boost the results.

Despite that tags have some structure, in this work they are treated separately. Therefore the system can not gen-

Table 7: Results of PolEval subtask C. $POSAcc$ is accuracy lower bound for morphosyntactic tagging, $LemAcc$ is the accuracy of lemmatization and $OverallAcc$ is the average of $POSAcc$ and $LemAcc$.

	Tagging		
Tagger	$POSAcc$	$LemAcc$	$OverallAcc$
KRNNT	**92.98**	96.91	**94.95**
NeuroParser	91.59	**97.00**	94.29
MorphoDiTaPL	89.67	95.78	92.73

eralize well, e.g. "verb must be in every sentence" instead of "one of X tags describing verb must be in sentence". Tags should be partitioned on output, or more fine-grained outputs should be added.

Researchers should also focus on word embeddings for morphologically rich languages. Including them in a tagger should improve accuracy. What is more, representing tags as word embeddings might be beneficial because they can represent dependencies among them (Goldberg, 2016).

RNNs make local decisions for each token, incorporating CRF or hidden Markov models as the last layer will assign labels after seeing the word sequence (Huang et al., 2015b).

Including information from the whole paragraph is essential for some ambiguities in tagging.

8. References

Acedański, Szymon, 2010. A morphosyntactic brill tagger for inflectional languages. In *International Conference on Natural Language Processing*. Springer.

Brill, Eric, 1992. A simple rule-based part of speech tagger. In *Proceedings of the workshop on Speech and Natural Language*. Association for Computational Linguistics.

Cho, Kyunghyun, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio, 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078.

Dozat, Timothy, 2016. Incorporating nesterov momentum into adam.

Goldberg, Yoav, 2016. A primer on neural network models for natural language processing. *J. Artif. Intell. Res.(JAIR)*, 57:345–420.

Graves, Alex and Navdeep Jaitly, 2014. Towards end-to-end speech recognition with recurrent neural networks. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*.

Graves, Alex, Abdel-rahman Mohamed, and Geoffrey Hinton, 2013. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE.

Hochreiter, Sepp and Jürgen Schmidhuber, 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Huang, Zhiheng, Wei Xu, and Kai Yu, 2015a. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991.

Huang, Zhiheng, Wei Xu, and Kai Yu, 2015b. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Kobyliński, Łukasz and Witold Kieraś, 2016. Part of speech tagging for Polish: State of the art and future perspectives. In *Proceedings of the 17th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing 2016)*. Konya, Turkey.

Kobyliński, Łukasz and Maciej Ogrodniczuk, 2017. Results of the PolEval 2017 Competition: Part-of-Speech Tagging Shared Task. In *Proceedings of 8th Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics*. Poznań, Poland: Wydawnictwo Poznańskie i Fundacja Uniwersytetu im. A. Mickiewicza.

Kuta, Marcin, Paweł Chrzaszcz, and Jacek Kitowski, 2012. A case study of algorithms for morphosyntactic tagging of polish language. *Computing and Informatics*, 26(6):627–647.

Lafferty, John, Andrew McCallum, and Fernando CN Pereira, 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

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

Pohl, Aleksander and Bartosz Ziółko, 2013. Using part of speech n-grams for improving automatic speech recognition of polish. In *Machine Learning and Data Mining in Pattern Recognition*. Springer Berlin Heidelberg, pages 492–504.

Radziszewski, Adam, 2013. A tiered crf tagger for polish. In *Intelligent tools for building a scientific information platform*. Springer, pages 215–230.

Radziszewski, Adam and Szymon Acedański, 2012. Taggers gonna tag: an argument against evaluating disambiguation capacities of morphosyntactic taggers. In *International Conference on Text, Speech and Dialogue*. Springer.

Radziszewski, Adam and Tomasz Śniatowski, 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, Adam and Tomasz Śniatowski, 2011. A memory-based tagger for polish. In *Proceedings of the 5th Language & Technology Conference, Poznań*.

Sutskever, Ilya, James Martens, George Dahl, and Geoffrey Hinton, 2013. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*.

Walentynowicz, Wiktor, 2017. MorphoDiTa-based tagger for polish language. CLARIN-PL digital repository.

Wang, Di and Eric Nyberg, 2015. A long short-term memory model for answer sentence selection in question answering. In *ACL (2)*.

Waszczuk, Jakub, 2012. Harnessing the crf complexity with domain-specific constraints. the case of morphosyntactic tagging of a highly inflected language. In *COLING*.