Proceedings of the PolEval 2018 Workshop

Maciej Ogrodniczuk, Łukasz Kobyliński (eds.)

Institute of Computer Science, Polish Academy of Sciences

Warszawa, 2018

Contents

PolEval 2018: Instead of a Preface

Maciej Ogrodniczuk, Łukasz Kobyliński (Institute of Computer Science, Polish Academy of Sciences)

There has been a lot of research activity in Natural Language Processing in Poland in recent years but the community was lacking something competitive and concrete: this is how the idea of PolEval was conceived. It was designed to become a forum for evaluating NLP tools for processing Polish with established evaluation measures, procedures and data sets.

It all started only in 2017, and, as with any such new initiative among the multitude of other options, could be limited to a one-year event or could grow — and we are happy to experience the latter. In 2018 we were very pleased to receive a total of 24 submissions from 14 teams. Some of them are companies which seems to be the current trend, often neglected by the scholars but more and more evident. The big and smaller research labs successfully employ the top researchers, offering them motivating working environments, access to big data and top-notch infrastructure. It can only make us happier that they decide to take part.

The current procedings are the result of a special session organized during the AI & NLP Day 2018 (`https://nlpday.pl`), which took place on October 19th, 2018 at the Institute of Computer Science, Polish Academy of Sciences in Warsaw. The event gathered many PolEval 2018 participants and featured 12 presentations of their submissions. Apart from individual descriptions, 9 in the form of research papers and 2 as short abstracts, the volume presents an overview of the submitted systems in each task summarizing the machine learning methods used, system architectures, features and training data.

Thanks to our sponsors, we were happy to award 4 prizes in each category based on the condition of releasing the source code of the winning systems. We view this as one of additional achievements of our initivative.

We hope you will join us next year for PolEval 2019! Please feel free to share your ideas for improving this competition or willingness to help in organizing your own NLP tasks.

Organizing Committee

Maciej Ogrodniczuk

Institute of Computer Science, Polish Academy of Sciences

Łukasz Kobyliński

Institute of Computer Science, Polish Academy of Sciences
Sages

Aleksander Wawer

Institute of Computer Science, Polish Academy of Sciences
Samsung R&D Center Poland

Grzegorz Wojdyga

Institute of Computer Science, Polish Academy of Sciences

Alina Wróblewska

Institute of Computer Science, Polish Academy of Sciences

Acknowledgements

PolEval 2018 Sponsor

We would like to thank Samsung Electronics Poland for supporting PolEval 2018 which allowed us to provide prizes for the competition winners.

AI & NLP Day Workshop Sponsors

We would like to thank the following organizations for supporting the 2018 edition of AI & NLP Day Workshop.

AI & NLP Day Workshop Organizers

Results of the PolEval 2018 Competition: Dependency Parsing Shared Task

Alina Wróblewska (Institute of Computer Science, Polish Academy of Sciences)

Abstract

This paper summarises the first PolEval 2018 shared task on dependency parsing of Polish. The focus of this task is to develop NLP systems that can analyse Polish sentences and predict their morphosyntactic representations possibly with some semantic extensions. Except for gold-standard tokenisation and sentence segmentation, no other gold-standard annotation is included in test data. The participating systems have to predict labelled dependency trees of the tokenised sentences as well as universal part-of-speech tags, universal morphological features, Polish-specific tags, and lemmata of the individual tokens (subtask 1A). The participants are also encouraged to predict semantic roles of some dependents and labelled enhanced graphs (subtask 1B). Dependency parsing systems are trained and tested on data from Polish Dependency Bank in the UD-like format. Four systems partake in the dependency subtask 1A and three of them in the subtask 1B. The winner of the subtask 1A — COMBO — predicts the labelled dependency trees with LAS F_1-score of 86.11, the Polish-specific tags with F_1-score of 93.44, and the lemmata with F_1-score of 97.27. In the subtask 1B, the most accurate enhanced graphs are predicted by the IMS system (ELAS F_1-score of 81.9) and the semantic roles are most precisely predicted with the COMBO system (SLAS F_1-score of 77.3).

Keywords

Polish, natural language processing, dependency parsing, enhanced graphs, semantic roles, evaluation

1. Introduction

The PolEval series is an annual Polish language processing competition organised by Institute of Computer Science PAS[1] in cooperation with other institutions and companies. The first PolEval 2017 competition featured two shared tasks on part-of-speech tagging (Kobyliński and Ogrodniczuk 2017) and sentiment analysis (Wawer and Ogrodniczuk 2017). The second edition of the competition — PolEval 2018 (Ogrodniczuk and Kobyliński 2018) — offers three tasks on dependency parsing, named entity recognition, and language models.

This paper summarises the first PolEval 2018 shared task on dependency parsing of Polish.[2] The focus of this task is to develop NLP systems that can analyse Polish sentences and predict their morphosyntactic representations possibly with some semantic extensions. Except for gold-standard tokenisation and sentence segmentation, no other gold-standard annotation is included in test data. The participating systems have to predict labelled dependency trees of the tokenised sentences, i.e. they have to predict a syntactic governor for each token and a proper label of the relation between the token and its predicted governor. Furthermore, the systems are required to predict universal part-of-speech tags, universal morphological features, Polish-specific tags, and lemmata of the individual tokens (subtask 1A). The participants are also encouraged to predict semantic roles of some dependents and labelled enhanced graphs, i.e. labelled dependency trees with the morphosyntactically annotated tokens and with the enhanced edges encoding the shared dependents and the shared governors of the coordinated conjuncts (subtask 1B). The participants can take part in one of two subtasks or in both subtasks (see Section 2 for a detailed description of both subtasks).

Dependency parsing systems are trained and tested on data from Polish Dependency Bank (Wróblewska 2014) provided by the organisers of the shared task (see Section 3). As the Universal Dependencies (UD) annotation schema (Nivre et al. 2016) has become the standard of annotating dependency trees in recent years, the provided data sets are in the UD-like format (Wróblewska 2018). Four systems partake in the dependency subtask 1A and three of them in the subtask 1B. The participating systems are briefly characterised in Section 4. The evaluation issues, i.e. the evaluation metrics and the evaluation script, are presented in Section 5. Finally, the results and some conclusions are in Sections 6 and 7, respectively.

[1]`https://ipipan.waw.pl/en/`

[2]`http://poleval.pl/tasks#task1`

2. Task Description

2.1. Task 1A: Morphosyntactic Prediction of Dependency Trees

The participating systems have to predict labelled dependency trees of the tokenised sentences and morphosyntactic analyses of the individual tokens. Except for the ROOT node, each node of a dependency tree corresponds to one sentence token. Each of these nodes depends on exactly one governing node (HEAD) and the relation between this node and its governor is labelled with a dependency type (DEPREL) from the repertoire of the universal dependency labels.[3] The UD dependency types can be extended with the Polish-specific subtypes, e.g. `advmod:arg` (an adverbial subcategorised by a verb) for labelling the function of *lepiej* ('better') governed by *mieć* ('to have') in *Wiem, że możemy mieć lepiej* ('I know that our situation/conditions will improve', lit. 'I know that we can have better'). The morphosyntactic analysis, in turn, consists in predicting universal part-of-speech tags (UPOS), Polish-specific tags (XPOS), universal morphological features (FEATS), and lemmata (LEMMA) of the individual tokens. If participants do not aim at predicting the morphosyntactic analyses of the Polish tokens, their systems are allowed to only predict labelled dependency trees (the morphosyntactic analyses are predicted with the baseline parser in this case). The dependency trees are encoded in the CoNLL-U format.[4]

2.2. Task 1B: Beyond Dependency Tree

The participants are encouraged to predict semantically motivated labelled dependency graphs, i.e. the labelled dependency trees with the enhanced edges and with the semantic roles, which some dependents can be additionally annotated with. The enhanced edges encode the shared dependents of the coordinated elements (e.g. *Piotr wstał i wyszedł.* 'Piotr stood up and left.') and the shared governors of the coordinated elements (e.g. *Lubimy babeczki i ciasteczka.* 'We like cupcakes and cookies.'). The additional semantic roles (e.g. `Experiencer`, `Place`, `Condition`) extend the semantic meaning of indirect objects (`iobj`), oblique nominals (`obl`), adverbial clause modifiers (`advcl`), some adverbial modifiers (`advmod`), some noun modifiers (`nmod`), etc. The semantically motivated enhanced graphs are encoded in the CoNLL-U-like format with the enhanced edges in the 9th column (DEPS) and the semantic roles in the additional 11th column (SEM).

[3] `http://universaldependencies.org/u/dep/index.html`
[4] `http://universaldependencies.org/format.html`

3. Data

The participating systems can be trained on the PDBUD trees (Wróblewska 2018), i.e. the trees from Polish Dependency Bank (Wróblewska 2014) converted to the Universal Dependencies format (Nivre et al. 2016). The updated version of PDBUD is publicly available.[5]

3.1. Data Split

PDBUD is divided into three parts – training (17,770 trees), test (2219 trees) and development (2219 trees) data sets (see Table 1 for more details). The procedure of assigning dependency trees to particular data sets is generally random while maintaining the proportion of sentences from individual sources, i.e. NKJP (Przepiórkowski et al. 2012), CDSCorpus (Wróblewska and Krasnowska-Kieraś 2017), projection-based corpus (Wróblewska and Przepiórkowski 2014) and literature. There is one constraint on the dividing procedure — the trees from *Składnica zależnościowa*[6] (Wróblewska 2012) are not included in the test set. The *Skladnica* trees have been publicly available for some time and we decided to exclude them in the validation process. Since sentences underlying the *Składnica* trees are generally shorter than the remaining sentences, the average number of tokens per sentence is significantly higher in the test set than in two other sets.

Table 1: Statistics of the training (train), test, and development (dev) data sets of PDBUD

		PDBUD	
	train	**test**	**dev**
number of sentences	17770	2219	2219
average number of tokens per sentence	15.4	20.2	15.1
number of non-projective trees	1310	302	172
percent of non-projective trees	7.4	13.6	7.7
number of enhanced graphs	7147	1181	855
percent of enhanced graphs	40.2	53.2	38.5

[5] `http://git.nlp.ipipan.waw.pl/alina/PDBUD`
[6] *Składnica zależnościowa* is the first Polish dependency treebank.

3.2. Test Data Set

Sentence segmentation and tokenisation are not evaluated in the current shared task. We therefore provide the gold-standard tokenised test sentences. The morphosyntactic properties of the test tokens — UPOS, XPOS, FEATS, and LEMMA — are automatically predicted with UDPipe (Straka and Straková 2017) trained on PDBUD training data. UDPipe's predictions should be replaced with the participants' predictions.

3.3. Additional Resources

The following additional resources are allowed to use while training dependency parsing systems for Polish:

— all data collected for the purpose of the CoNLL 2018 UD shared task (Zeman et al. 2018),[7]

— Polish word embeddings.[8]

4. Participating Systems

The overview of four systems participating in the first PolEval 2018 shared task is presented in Table 2. All systems are briefly described in the following subsections.

Table 2: The overview of the systems participating in the first task of the PolEval 2018 competition. Explanations: UPOS: universal part-of-speech tag; XPOS – Polish-specific tag; FEATS – list of universal morphological features; LEMMA – lemma; HEAD – head of the current word; DEPREL – universal dependency relation; DEPS – enhanced dependency graph; SEM – semantic role.

System	Architecture	Predictions			
		HEAD DEPREL	UPOS XPOS FEATS LEMMA	DEPS	SEM
COMBO	neural	yes	yes	yes	yes
IMS	CRF/neural/rules	yes	yes	yes	no
Poleval2k18	neural	yes	yes	no	yes
Drewutnia	neural	yes	no	no	no

[7] http://universaldependencies.org/conll18/data.html

[8] http://dsmodels.nlp.ipipan.waw.pl, http://mozart.ipipan.waw.pl/~axw/models

4.1. COMBO

COMBO[9] (Rybak and Wróblewska 2018) is a neural system that partakes in both subtasks of the Task 1. The COMBO system used in the subtask 1A is a version of the ICS PAS system (Rybak and Wróblewska 2018) participating in the CoNLL 2018 UD shared task (Zeman et al. 2018). COMBO consists of jointly trained tagger, lemmatizer, and dependency parser. They are based on the features extracted by a bidirectional long-short term memory network (biLSTM), which takes the concatenation of external word embeddings and internal character-based word embeddings as input. COMBO uses both fully connected and dilated convolutional neural network architectures.

In the subtask 1B, COMBO predicts enhanced edges and semantic roles (Rybak and Wróblewska 2018b). The enhanced graphs are predicted in the similar way as the dependency trees in the subtask 1A. Instead of the soft-max function, the sigmoid activation function is applied to each row of the adjacency matrix to predict an enhanced graph, i.e. to predict all heads for each token of a sentence. Dependency labels of these arcs are predicted with a fully connected neural network with one hidden layer. The soft-max activation function is used to force the network to predict only one label for each arc. Both parts are jointly optimised using cross-entropy. COMBO predicts the semantic roles using a fully connected neural network with one hidden layer that takes the features extracted by biLSTM as input. The procedure of predicting semantic roles is similar to the prediction of part-of-speech tags.

COMBO is trained on data provided for the purpose of the PolEval 2018 dependency shared task. It also uses pre-trained word embeddings.[10]

4.2. IMS

The IMS team submitted two systems for both subtasks of the Task 1 (Falenska et al. 2018). The system for the subtask 1A is based on the IMS ensemble system (Björkelund et al. 2017), which successfully participated in the CoNLL 2017 UD shared task (Zeman et al. 2017). The IMS system integrates:

— a CRF tagger predicting morphological features, part-of-speech tags, and lemmata,

— a neural tagger predicting supertags which are incorporated into the feature model of a dependency parser (Ouchi et al. 2014),

— an ensemble of multiple graph-based and transition-based parsers applying the blending technique (Sagae and Lavie 2006) for combining parsers' outputs.

[9]`https://github.com/360er0/COMBO`

[10]`http://mozart.ipipan.waw.pl/~axw/models`

The system designed for the subtask 1B processes sentences in two steps: (1) the sentences are parsed with the system provided for the subtask 1A, and (2) the enhanced edges are predicted with a small set of rules. This rule-based system does not predict semantic roles.

The IMS system is trained on PolEval 2018 data. Except for training data, the system uses the pre-trained word embeddings prepared for the CoNLL 2017 UD shared task.[11]

4.3. Poleval2k18

Poleval2k18[12] (Zapotoczny et al. 2017) is a neural system participating in both subtasks of the Task 1. The system takes character-segmented words as input and estimates word embeddings with a feedforward network. The sentence tokens represented with the embeddings are further processed with a bidirectional GRU recurrent neural network. The governor of each token is predicted with an attention mechanism. The final dependency tree is estimated with Chu-Liu-Edmonds algorithm (Chu and Liu 1965, Edmonds 1967). The dependency labels and the semantic roles are predicted with an additional hidden layer followed by the soft-max function. The system architecture is described in details in (Zapotoczny et al. 2017).

4.4. Drewutnia

Drewutnia (Skuczyńska 2018) is a neural system participating in the subtask 1A and predicting labelled dependency trees. The system takes tokenised and part-of-speech tagged sentences in the CoNLL-U format as input and predicts dependency edges with a bidirectional GRU recurrent neural network. In the postprocessing phase, the predicted structures are normalised, i.e. the number of ROOT's dependents is reduced to one in each sentence and the cycles are resolved by attaching all nodes to the dependent of the ROOT node. Drewutnia is trained on PolEval 2018 data.

5. Evaluation

5.1. Evaluation Measures

The measures defined for the purpose of the CoNLL 2018 UD shared task (Zeman et al. 2017), i.e. LAS, MLAS, and BLEX, are also used in the PolEval 2018 shared task

[11] `https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-1989`

[12] `https://github.com/mzapotoczny/dependency-parser`

on dependency parsing of Polish. Additionally, we define two new measures ELAS and SLAS for the evaluation of enhanced graphs and semantic roles. Each metric is the harmonic mean (F_1) of precision P and recall R (see equation below). P and R are differently defined in the particular measures.

$$metric = \frac{2PR}{P+R}$$

LAS (labelled attachment score) evaluates how many words are correctly parsed. Precision P is the ratio of the number of the correct relations to the number of the predicted nodes. Recall R is the ratio of the number of the correct relations to the number of the gold standard nodes.

MLAS (morphology-aware labelled attachment score) is inspired by the CLAS metric (Zeman et al. 2017) and extended with the evaluation of part-of-speech tags and morphological features. Precision P is the ratio of the number of the correct words to the total number of the predicted content words. Recall R is the ratio of the number of the correct words to the total number of the gold-standard content words. The predicted word S is considered to be correct, if the following conditions are met:

— S is assigned the correct HEAD and DEPREL,

— S is assigned the correct UPOS, XPOS, and FEATS.

A word is considered to be a content word, if it fulfils one of the following content functions: `nsubj`, `obj`, `iobj`, `csubj`, `ccomp`, `xcomp`, `obl`, `vocative`, `expl`, `dislocated`, `advcl`, `advmod`, `discourse`, `nmod`, `appos`, `nummod`, `acl`, `amod`, `conj`, `fixed`, `flat`, `compound`, `list`, `parataxis`, `orphan`, `goeswith`, `reparandum`, `root`, and `dep`.

BLEX (bi-lexical dependency score) evaluates dependencies and lexemes, i.e. it combines content-word relations with lemmatisation. Precision P and recall R are defined exactly the same as in MLAS, but the predicted word S is considered to be correct, if the following conditions are met:

— S is assigned the correct HEAD and DEPREL,

— S is assigned the correct LEMMA.

ELAS (enhanced labelled attachment score) is defined for the purpose of evaluating enhanced graphs. ELAS is a strict extension of LAS. Precision P is the ratio of the number of the correct words to the total number of the predicted nodes. Recall R is the ratio of the number of the correct words to the total number of the gold-standard nodes. The predicted word S is considered to be correct, if the following conditions are met:

— S is assigned the correct HEAD and DEPREL,

— S is assigned the correct set of enhanced edges (DEPS).

SLAS (semantic-aware labelled attachment score) is defined for evaluating dependency trees with semantic roles of some dependents. Precision P and recall R are defined exactly the same as in MLAS, but the predicted word S is considered to be correct, if the following conditions are met:

— S is assigned the correct HEAD and DEPREL,

— S is assigned the correct value of SEM, i.e. it is either the correct semantic role or the underscore indicating 'no semantic role'.

5.2. Evaluation Script

The submitted files with predictions should be in the CoNLL-U format with the standard 10 columns (the subtask 1A) or in the CoNLL-U-like format with 11 columns (the subtask 1B). The predicted analyses are evaluated with the script `poleval2018_cykle.py`.[13] It is a modified version of the evaluation script prepared for the CoNLL 2018 UD shared task. The most important modification consists in adding two measures, ELAS and SLAS, for the purpose of evaluating enhanced graphs and semantic roles, respectively. Next, instead of evaluating some selected morphological features and the universal parts of the predicted dependency types, we evaluate all morphological features and the full dependency labels (i.e. universal dependency types possibly extended with Polish-specific subtypes). The final modification is motivated by the fact that some of the participating systems predict ill-formed dependency trees, i.e. structures with cycles, multiple dependents of the ROOT node, etc. We decided not to reject such submissions, but only to score incorrect trees with 0.

[13] `http://poleval.pl/task1/poleval2018_cykle.py`

6. Results

6.1. Task 1A

COMBO is the indisputable winner of the PolEval 2018 subtask 1A (see Table 3). It predicts the Polish tags (XPOS) with F_1-score of 93.44, the Polish lemmata with F_1-score of 97.27, and the dependency trees with LAS F_1-score of 86.11. Apart from external word embeddings it does not use other additional resources (e.g. dictionaries) and external tools (e.g. morphological analysers). The predictions of the second best system – IMS – are also of high quality. This is noteworthy, because it is the only one system developed by a foreign team which participated in the PolEval 2018 competition. The results of the last two systems are below the baseline (UDPipe). The main reason for the worse performance of these systems is that they predict ill-formed dependency trees for some sentences (i.e. structures with cycles or multiple roots). All tokens of these structures are scored 0 with respect to all metrics and thus the overall scores are low.

Table 3: Results of the subtask 1A of the PolEval 2018 competition. The systems are ranked by LAS F_1-scores.

System	UPOS	XPOS	FEATS	LEMMA	UAS	LAS	MLAS	BLEX
1. COMBO	**98.07**	**93.44**	**94.53**	**97.27**	**91.31**	**86.11**	**76.18**	**79.86**
2. IMS	97.38	86.89	90.16	83.54	89.31	83.82	69.27	60.88
3. UDPipe (baseline)	96.81	86.05	88.02	95.61	83.32	78.93	64.33	71.17
4. Poleval2k18	93.62	83.2	84.91	92.47	84.65	77.70	61.21	70.01
5. Drewutnia	n/a	n/a	n/a	n/a	33.98	27.39	n/a	n/a

6.2. Task 1B

Three systems participate in the subtask 1B. COMBO predicts enhanced graphs and semantic roles, IMS predicts only enhanced graphs, and Poleval2k18 predicts only semantic roles. The results are presented in Table 4.

The rule-based IMS system outperforms the neural COMBO system in predicting enhanced graphs, i.e. 81.9 vs. 80.66 (ELAS F_1). Semantic roles are more accurately

Table 4: Results of the subtask 1B of the PolEval 2018 competition. The systems are ranked by ELAS F_1-scores.

System	ELAS	SLAS
1. IMS	**81.90**	n/a
2. COMBO	80.66	**77.30**
3. Poleval2k18	n/a	67.84
4. UDPipe (baseline)	72.49	59.34

predicted by COMBO than by the Poleval2k18 system, i.e. 77.3 vs. 67.84 (SLAS F_1). All systems outperform the baseline (UDPipe).[14]

7. Conclusions

The current edition of the PolEval competition featured the shared task on dependency parsing of Polish, in which the participating teams trained and tested their systems on Polish Dependency Bank in the UD-like format. There were two subtasks in the parsing shared task. One of them was the morphosyntactic prediction of dependency trees, i.e. the prediction of the dependency trees together with the prediction of the universal part-of-speech tags, the Polish-specific tags, the morphological features, and the lemmata of the gold-standard tokens. The second subtask consisted in the prediction of the enhanced graphs and the semantic roles of some dependents. Four systems participated in the PolEval 2018 shared task on dependency parsing of Polish. Three of them were pure neural systems and the IMS system used a neural network for dependency prediction, conditional random fields for morphosyntactic predictions, and rules for predicting enhanced graphs.

The predictions were evaluated with the metrics defined in the CoNLL 2018 UD shared task. In addition to the CoNLL metrics, we defined two new metrics — ELAS and SLAS, for the purpose of evaluating enhanced graphs and semantic roles. The systems' predictions were of high quality. The winner of the subtask 1A — COMBO — predicted the labelled dependency trees of Polish sentences with LAS F_1-score of 86.11. According to our knowledge, it was the first shared task on predicting enhanced graphs. The most accurate enhanced graphs were predicted by the IMS system (ELAS F_1-score of 81.9). The prediction of dependency trees extended with semantic roles

[14]In the baseline prediction, the enhanced edges are replaced with the predicted dependency edges, i.e. the pairs HEAD:DEPREL are in the 9th column (DEPS), and the semantic roles in the 11th column are replaced with single underscore characters (SEM).

of some dependents was also a novelty, as semantic role labelling is currently not supported in Universal Dependencies.

Acknowledgements

The research presented in this paper was founded by the Polish Ministry of Science and Higher Education as part of the investment in the CLARIN-PL research infrastructure.

References

Björkelund A., Falenska A., Yu X. and Kuhn J. (2017). *IMS at the CoNLL 2017 UD Shared Task: CRFs and Perceptrons Meet Neural Networks*. [in:] *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pp. 40–51.

Chu Y. J. and Liu T. H. (1965). *On the Shortest Arborescence of a Directed Graph*. „Science Sinica", 14, pp. 1396–1400.

Edmonds J. (1967). *Optimum Branchings*. „Journal of Research of the National Bureau of Standards", 71B(4), pp. 233–240.

Falenska A., Björkelund A., Yu X. and Kuhn J. (2018). *IMS at the PolEval 2018: Bulky Ensemble Dependency Parser meets 12 Simple Rules for Predicting Enhanced Dependencies in Polish*. [in:] Ogrodniczuk and Kobyliński (2018), pp. 25–39.

Kobyliński Ł. and Ogrodniczuk M. (2017). *Results of the PolEval 2017 Competition: Part-of-Speech Tagging Shared Task*. [in:] Vetulani Z. and Paroubek P. (eds.), *Proceedings of the 8th Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics*, pp. 362–366. Fundacja Uniwersytetu im. Adama Mickiewicza w Poznaniu.

Ogrodniczuk M. and Kobyliński Ł., eds. (2018). *Proceedings of the PolEval 2018 Workshop*, PolEval 2018. Institute of Computer Science, Polish Academy of Sciences.

Nivre J., de Marneffe M., Ginter F., Goldberg Y., Hajič J., Manning C. D., McDonald R. T., Petrov S., Pyysalo S., Silveira N., Tsarfaty R. and Zeman D. (2016). *Universal Dependencies v1: A Multilingual Treebank Collection*. [in:] *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*, pp. 1659–1666. European Language Resource Association.

Ouchi H., Duh K. and Matsumoto Y. (2014). *Improving Dependency Parsers with Supertags*. [in:] *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, (Volume 2: Short Papers)*, pp. 154–158.

Przepiórkowski A., Bańko M., Górski R. L. and Lewandowska-Tomaszczyk B., eds. (2012). *Narodowy Korpus Języka Polskiego*. Wydawnictwo Naukowe PWN, Warsaw.

Rybak P. and Wróblewska A. (2018a). *Semi-Supervised Neural System for Tagging, Parsing and Lematization*. [in:] *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pp. 45–54. Association for Computational Linguistics.

Rybak P. and Wróblewska A. (2018b). *Semi-Supervised Neural System for Tagging, Parsing and Lematization. Addendum*. [in:] Ogrodniczuk and Kobyliński (2018), pp. 43–45.

Sagae K. and Lavie A. (2006). *Parser Combination by Reparsing*. [in:] *Proceedings of the Human Language Technology Conference of the NAACL (Companion Volume: Short Papers)*, pp. 129–132. Association for Computational Linguistics.

Skuczyńska B. (2018). *Drewutnia: Frugal Approach to Dependency Parsing*. [in:] Ogrodniczuk and Kobyliński (2018), pp. 40–47.

Straka M. and Straková J. (2017). *Tokenizing, POS Tagging, Lemmatizing and Parsing UD 2.0 with UDPipe*. [in:] *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pp. 88–99. Association for Computational Linguistics.

Wawer A. and Ogrodniczuk M. (2017). *Results of the PolEval 2017 Competition: Sentiment Analysis Shared Task*. [in:] Vetulani Z. and Paroubek P. (eds.), *Proceedings of the 8th Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics*, pp. 406–409. Fundacja Uniwersytetu im. Adama Mickiewicza w Poznaniu.

Wróblewska A. (2012). *Polish Dependency Bank*. „Linguistic Issues in Language Technology”, 7(1), pp. 1–15.

Wróblewska A. (2014). *Polish Dependency Parser Trained on an Automatically Induced Dependency Bank*. Ph.D. dissertation, Institute of Computer Science, Polish Academy of Sciences, Warsaw.

Wróblewska A. (2018). *Extended and Enhanced Polish Dependency Bank in Universal Dependencies Format*. [in:] *Proceedings of Universal Dependencies Workshop 2018* (UDW 2018).

Wróblewska A. and Krasnowska-Kieraś K. (2017). *Polish Evaluation Dataset for Compositional Distributional Semantics Models*. [in:] *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 784–792. Association for Computational Linguistics.

Wróblewska A. and Przepiórkowski A. (2014). *Towards a Weighted Induction Method of Dependency Annotation*. [in:] Przepiórkowski A. and Ogrodniczuk M. (eds.), *Advances in Natural Language Processing: Proceedings of the 9th International Conference on NLP (PolTAL 2014)*, *Lecture Notes in Artificial Intelligence vol. 8686*, pp. 164–176. Springer International Publishing, Heidelberg.

Zapotoczny M., Rychlikowski P. and Chorowski J. (2017). *On Multilingual Training of Neural Dependency Parsers*. [in:] Ekštein K. and Matoušek V. (eds.), *Text, Speech, and Dialogue. Proceedings of the 20th International Conference (TSD 2017)*, *Lecture Notes in Artificial Intelligence vol. 10415*, pp. 326–334. Springer International Publishing.

Zeman D., Popel M., Straka M., Hajič J., Nivre J., Ginter F., Luotolahti J., Pyysalo S., Petrov S., Potthast M., Tyers F., Badmaeva E., Gökırmak M., Nedoluzhko A., Cinková S., Hajič jr. J., Hlaváčová J., Kettnerová V., Urešová Z., Kanerva J., Ojala S., Missilä A., Manning C., Schuster S., Reddy S., Taji D., Habash N., Leung H., de Marneffe M.-C., Sanguinetti M., Simi M., Kanayama H., de Paiva V., Droganova K., Alonso H. M., Çöltekin Ç., Sulubacak U., Uszkoreit H., Macketanz V., Burchardt A., Harris K., Marheinecke K., Rehm G., Kayadelen T., Attia M., Elkahky A., Yu Z., Pitler E., Lertpradit S., Mandl M., Kirchner J., Alcalde F. H., Strnadová J., Banerjee E., Manurung R., Stella A., Shimada A., Kwak S., Mendonça G., Lando T., Nitisaroj R. and Li J. (2017). *CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. [in:] *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pp. 1–19. Association for Computational Linguistics.

Zeman D., Hajič J., Popel M., Potthast M., Straka M., Ginter F., Nivre J. and Petrov S. (2018). *CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. [in:] *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pp. 1–20. Association for Computational Linguistics.

IMS at the PolEval 2018: A Bulky Ensemble Dependency Parser Meets 12 Simple Rules for Predicting Enhanced Dependencies in Polish

Agnieszka Falenska, Anders Björkelund, Xiang Yu, Jonas Kuhn
(Institute for Natural Language Processing, University of Stuttgart, Germany)

Abstract

This paper presents the IMS contribution to the PolEval 2018 Shared Task.[1] We submitted systems for both of the Subtasks of Task 1. In Subtask (A), which was about dependency parsing, we used our ensemble system from the CoNLL 2017 UD Shared Task. The system first preprocesses the sentences with a CRF POS/morphological tagger and predicts supertags with a neural tagger. Then, it employs multiple instances of three different parsers and merges their outputs by applying blending. The system achieved the second place out of four participating teams. In this paper we show which components of the system were the most responsible for its final performance.

The goal of Subtask 1B was to predict enhanced graphs. Our approach consisted of two steps: parsing the sentences with our ensemble system from Subtask 1A, and applying 12 simple rules to obtain the final dependency graphs. The rules introduce additional enhanced arcs only for tokens with “conj” heads (conjuncts). They do not predict semantic relations at all. The system ranked first out of three participating teams. In this paper we show examples of rules we designed and analyze the relation between the quality of automatically parsed trees and the accuracy of the enhanced graphs.

Keywords

dependency parsing, enhanced dependencies, ensemble parsers

[1]`http://poleval.pl`

1. Introduction

This paper presents the IMS contribution to the PolEval 2018 Shared Task (PolEval18-ST). The Shared Task consisted of three Tasks: (1) Dependency Parsing, (2) Named Entity Recognition, and (3) Language Models. Our team took part only in the Task 1 and submitted systems for both of its Subtasks 1A and 1B.

The goal of the Subtask 1A was predicting morphosyntactic analyses and dependency trees for given sentences. The IMS submission was based on our ensemble system from the CoNLL 2017 UD Shared Task (Zeman et al. 2017). The system (described in detail in (Björkelund et al. 2017) and henceforth referred to as IMS17) relies on established techniques for improving accuracy of dependency parsers. It performs its own preprocessing with a CRF tagger, incorporates supertags into the feature model of a dependency parser (Ouchi et al. 2014), and combines multiple parsers through blending (also known as reparsing; Sagae and Lavie 2006).

The original system only needed few modifications to be applied in the PolEval18-ST setting. First, the organizers provided gold-standard tokenization so we excluded the tokenization modules from the system. Second, one of the metrics used in the PolEval18-ST was BLEX. While the metric takes lemmas into consideration we added a lemmatizer to the preprocessing steps. Finally, IMS17 was designed to run on the TIRA platform (Potthast et al. 2014), where only a limited amount of CPU time was available to parse a multitude of test sets. The maximal number of instances of individual parsers thus had to be limited to ensure that parsing would end within the given time. Since in the PolEval18-ST setting the parsing time was not limited we removed the time constraint from the search procedure of the system. We call the modified version IMS18.

The aim of Subtask 1B was to predict enhanced dependency graphs and additional semantic labels. Our approach consisted of two steps: parsing the sentences to surface dependency trees with our system from Subtask 1A, and applying a rule-based system to extend the trees with enhanced arcs. Since the PolEval18-ST data contains enhanced dependencies only for conjuncts, our set of manually designed rules is small and introduces new relations only for tokens with “conj” heads (it does not predict semantic labels at all).

All components of both submitted systems (including POS tagger, morphological analyzers, and lemmatizer) were trained only on the training treebank. Out of all the additional resources allowed by the organizers we used only the pre-trained word embeddings prepared for the CoNLL 2017 UD Shared Task.[2] We did not employ any

[2]https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-1989

Polish-specific tools as they (or the data their models were trained on) was not among the resources allowed by the organizers.

The remainder of this paper is organized as follows. Section 2 discusses our submission to Subtask 1A and analyzes which components of the system were the most responsible for its final performance. In Section 3 we describe our submission to Subtask 1B, show examples of the designed rules, and analyze the relation between the quality of automatically parsed trees and the accuracy of the enhanced graphs. Our official test set results are shown in Section 4 and Section 5 concludes.

2. Subtask 1A: Morphosyntactic Prediction of Dependency Trees

The focus of Subtask 1A was morphosyntactic prediction and dependency parsing. The training and development data contained information about gold-standard tokenization, universal part-of-speech tags (UPOS), Polish-specific tags (XPOS), universal morphological features (UFeats), lemmas, and dependency trees. The dependency trees were annotated with the Universal Dependencies (UD) (Nivre et al. 2016) according to the guidelines of UD v. 2.[3] To make the Shared Task more accessible to participants, the test data was released with baseline predictions for all preprocessing steps using the baseline UDPipe 1.2 system (Straka et al. 2016).

2.1. System Description

Figure 1 shows an overview of the IMS18 system architecture. The architecture can be divided into two steps: preprocessing and parsing. The system uses its own preprocessing tools, so we did not utilize the baseline UDPipe predictions provided by the ST organizers. All the preprocessing tools annotate the training data via 5-fold jackknifing. The parsing step consists of running multiple instances of three different baseline parsers and combining them into an ensemble system. All the trained models for both of the steps, as well as code developed during this Shared Task will be made available on the first author's page.

Below we give a summary of all the components of the system and describe changes introduced to the IMS17 system needed to adapt it to the PolEval18-ST setting.

Lemmatization is not performed by IMS17. Since BLEX, one of the metrics used in the PolEval18-ST, takes lemmas into consideration, we added a lemmatizer to the

[3] `http://universaldependencies.org/`

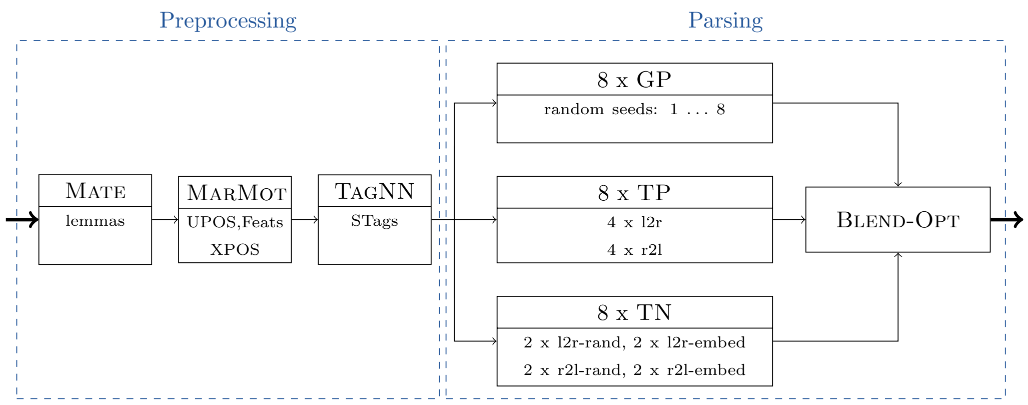


Figure 1: IMS18 system architecture

preprocessing steps. For this purpose we used lemmatizer from the `mate-tools` with default hyperparameters.[4]

Part-of-Speech and Morphological Tagging is performed within IMS17 by MARMOT, a morphological CRF tagger (Müller et al. 2013).[5] UPOS and UFeats are predicted jointly. Since IMS17 did not use XPOS tags, we added an additional CRF tagger predicting only XPOS tags (separately from other preprocessing steps). We used MARMOT with default hyperparameters.

Supertags (Joshi and Bangalore 1994) are labels for tokens which encode syntactic information, e.g., the head direction or the subcategorization frame. IMS17 follows (Ouchi et al. 2014) and extracts supertags from the training treebank. Then, it incorporates them into the feature models of all baseline dependency parsers. Supertags are predicted with an in-house neural-based tagger (TAGNN) (Yu et al. 2017).[6]

Baseline parsers used by IMS17 differ in terms of architecture and employed training methods. The system uses three baseline parsers: (1) The graph-based perceptron parser from `mate-tools` (Bohnet 2010), henceforth referred to as **GP** (the parser has been slightly modified to handle features based on supertags and shuffle training instances between epochs).[7] (2) An in-house transition-based beam-perceptron

[4]`https://code.google.com/archive/p/mate-tools/`

[5]`http://cistern.cis.lmu.de/marmot/`

[6]`https://github.com/EggplantElf/sclem2017-tagger`

[7]Since there are no time constraints in the POLEVAL18-ST (unlike the CoNLL 2017 Shared Task), GP is applied to all sentences, cf. (Björkelund et al. 2017) for details on how some sentences were skipped to save time in the IMS17 system.

parser (Björkelund and Nivre 2015), henceforth referred to as **TP**. (3) An in-house transition-based greedy neural parser (Yu and Vu 2017), henceforth referred to as **TN**. We use the default hyperparameters during training and testing of all the three baseline parsers.

Blending, i.e., combining outputs of multiple different baseline parsers, can lead to improved performance (Sagae and Lavie 2006). IMS17 parses every sentence with each baseline parser and combines all the predicted trees into one graph. It assigns scores to arcs depending on how frequent they are in the predicted trees. Then it uses the Chu-Liu-Edmonds algorithm (Chu and Liu 1965, Edmonds 1967) to find the maximum spanning tree in the combined graph. For every resulting arc it selects the most frequent label across all the labels previously assigned to it.

To enlarge the number of parsers taking part in the final ensemble IMS17 trains multiple instances of each baseline parser using different random seeds: (1) eight GP instances, (2) eight TP instances which differ in the direction of parsing – four parse from left to right (TP-l2r) and four from right to left (TP-r2l), (3) eight TN instances which differ in the direction of parsing and the used word embeddings – four use pre-trained embeddings (TN-l2r-embed, TN-r2l-embed) and four use randomly initialized embeddings (TN-l2r-rand, TN-r2l-rand).

The final component of the IMS17 system (BLEND-OPT) selects the best possible blending setting. It checks all the possible combinations of the above-mentioned instances ($9 \times 5 \times 5 \times 3 \times 3 \times 3 \times 3 = 18,225$ possibilities) and selects the one which achieves the highest LAS score on the development set. The original IMS17 limits the maximal number of instances of individual parsers to ensure that parsing will end within a restricted time. Since in the POLEVAL18-ST setting the parsing time was not limited we removed the time constraint from the search procedure BLEND-OPT.

Finally, since the UD guidelines do not allow multiple root nodes, we re-attach all excessive root dependents in a chain manner, i.e., every root dependent is attached to the previous one.

2.2. Evaluation of the Components of the System

In this section we evaluate all the components of the submitted IMS18 system with the evaluation script provided by the ST organizers. We use the UDPipe 1.2 system (as provided by the ST organizers) as a baseline through all the steps.

Preprocessing and Supertags. We begin with evaluating the preprocessing components of our system on the development data (see Table 1). We find that UDPipe is much better at predicting lemmas than `mate-tools` and it surpasses it by more than 10 points. On the contrary, MARMOT outperforms UDPipe on all the other tagging tasks, with the highest gain of more than two points on the task of predicting morphological features.

Table 1: Preprocessing accuracy (F_1 score) on the development set

	Lemma	**UPOS**	**XPOS**	**UFeats**
UDPipe	94.41	97.24	86.50	88.30
IMS	84.09	97.69	87.00	90.52

To see how the above-mentioned differences influence the parsing accuracy we run the baseline parsers (GP, TP, and TN) in four incremental settings: (1) using UPOS and morphological features predicted by UDPipe, (2) replacing UPOS and morphological features with MARMOT's predictions, (3) adding lemmas, (4) adding supertags. Table 2 shows LAS scores for the three baseline parsers for the consecutive experiments. Replacing UDPipe's UPOS and morphological features with the predictions from MARMOT improves accuracy by 0.42 points on average. The introduction of lemmas improves only the GP parser and leads to minuscule improvements for the other two. The step which influences the final accuracy the most is the addition of supertags. It brings an additional 0.9 points on average (with the biggest gain for TP of 1.54 points).

Table 2: Gains in parsing accuracy (LAS) for by incrementally replacing the UDPipe preprocessing baseline

	UDPipe	**MARMOT**	**+lemma**	**+STags**
GP	83.36	+0.27	+0.30	+1.03
TP (l2r)	81.80	+0.55	+0.01	+1.54
TN (l2r-rand)	82.77	+0.43	+0.03	+0.15
average	82.64	+0.42	+0.11	+0.90

Parsing and Blending. Table 3 shows parsing results on the development set. The relation between baseline parsers (rows 2, 3, and 4) is the same as in (Björkelund et al. 2017): GP is the strongest method, TP ranked second, and TN performs the worst. All the baseline parsers surpass the UDPipe parser (row 1) in terms of the LAS and MLAS measures. Since the measure BLEX uses lemmas and UDPipe is much better in terms of lemmatization, it achieves higher BLEX than the baseline parsers (in fact it achieves the highest BLEX across all the compared methods).

Table 3: Parsing accuracy (F_1 scores) on the development set. The highest value in each column is bold

		LAS	MLAS	BLEX
1	UDPipe	76.58	61.81	**71.39**
2	GP	84.96	71.32	63.04
3	TP (l2r)	83.80	70.14	61.82
4	TN (l2r-rand)	83.39	69.66	61.34
5	BLEND-BL	86.04	72.27	63.83
6	BLEND-OPT	**86.24**	**72.46**	63.98

Rows 5 and 6 show results of two separate blends. BLEND-BL (row 5) is an arbitrarily selected combination of 4+4+4 instances: four GP instances, four TP instances (two TP-l2r and two TP-r2l), and four TN instances (TN-l2r-rand, TN-r2l-rand, TN-l2r-embed, TN-l2r-embed). Comparing rows (2 – 4 with row 5 we see that blending parsers ends with a strong boost over the baselines, which corroborates the findings of (Sagae and Lavie 2006, Björkelund et al. 2017). The blended accuracy surpasses the strongest baseline parser GP by more than one point.

Finally, searching for the optimal combination yields an additional small improvement of 0.2 points. The best combination selected by the search contains: seven instances of GP, three instances of TP (two TP-l2r and one TP-r2l) and all the instances of TN.

3. Subtask 1B: Beyond Dependency Tree

The goal of Subtask 1B was to predict labeled dependency graphs and semantic labels. The dependency graphs used in the ST were UD dependency trees extended with additional enhanced arcs. The arcs encoded shared dependents and shared governors of conjuncts. The semantic labels (e.g. Experiencer, Place, Condition) were used to annotate additional semantic meanings of tokens.

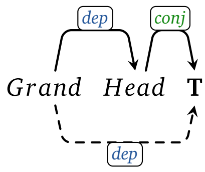
3.1. System Description

Our submission to the Subtask 1B followed (Schuster and Manning 2016, Candito et al. 2017) and carried out rule-based augmentation. The method consisted of two steps. First, we parsed all sentences to obtain surface dependency trees. Since the training data for Subtasks 1A and 1B was the same, we performed parsing with the same BLEND-OPT system as described in Section 2.1. In the second step, we applied 12 simple rules to the predicted trees and augmented them with enhanced relations.

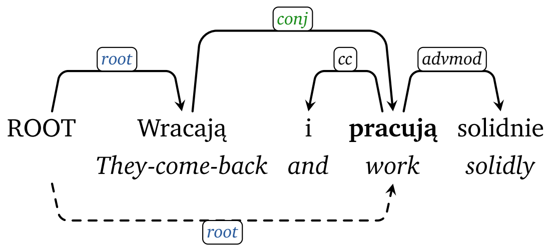
The rules of the system were designed manually and guided by intuition of a Polish native speaker while analyzing gold-standard graphs from the training part of the treebank. As the enhanced relations in the treebank mostly apply to conjuncts, i.e., tokens connected with the relation "conj" to their heads, our rules only apply to such tokens. We define two main rules: *Head*, which predicts additional heads, and *Children*, which adds enhanced children. The remaining 10 out of the 12 rules serve as filtering steps to improve the accuracy of the *Children* rule.

The *Head* Rule introduces enhanced arcs for all the tokens whose head is "conj" and connects them to their grandparents (see Figure 2a). Figure 2b shows an example of a sentence where an enhanced arc was introduced by the *Head* rule. The word *pracują* (eng. they-work) received an additional head ROOT.

When introducing enhanced heads for "conj" tokens, this rule achieves an F-score of 99.40 on the gold-standard trees from the training data.



(a) *Head* rule – adds the grandparent as an additional enhanced head. Applies to all tokens with "conj" heads



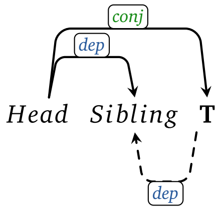
(b) Example sentence (id train-s9826) where the *Head* rule introduces a correct enhanced "root" arc

Figure 2: The *Head* rule

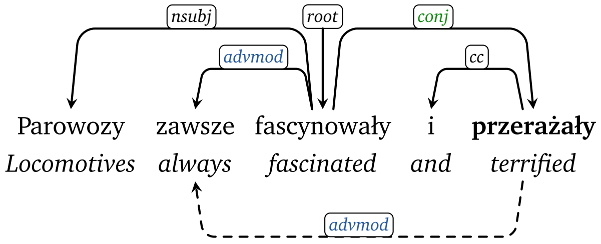
The *Children* Rule adds all the siblings of a "conj" token as its dependents (see Figure 3a). Figure 3b shows an example of a sentence where an enhanced arc was introduced by the *Children* rule. The word *zawsze* (eng. always) is a sibling of the "conj" token *przerażały* (eng. terrified) and therefore got attached to it by an "advmod" arc.

When introducing enhanced children of "conj" tokens this rule alone is too generous. On gold trees from the training data it has a perfect recall, it introduces a lot of incorrect arcs. It achieves a precision of only 21.64, resulting in an an F-score of 35.58.

We tackled this problem by designing 10 additional filtering rules which remove some suspicious arcs. Combined with the 10 filtering rules the *Children* rule achieves an F-score of 73.55 on the gold trees from the training data. Below we give examples of three such rules: *labels*, $advmod_1$, *obj*.



(a) The *Children* rule – adds siblings as enhanced dependents



(b) Example sentence (id `train-s9353`) where the *Children* rule predicts a correct enhanced "advmod" arc

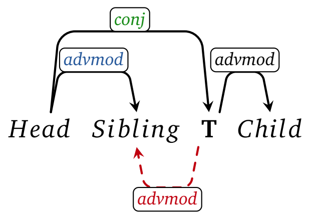
Figure 3: The *Children* rule

The filter *labels* removes all the enhanced arcs with labels that are not among the ten most common ones: case, nsubj, mark, obl, advmod, amod, cop, obj, discourse:comment, advcl.

The filter $advmod_1$ is the first of four filtering rules that remove enhanced arcs with label "advmod". It applies to tokens which have their own "advmod" basic modifiers (see Figure 4a). The intuition is that if the token has its own adverbial modifier then most likely the modifier of its head does not refer to it. Figure 4b shows an example of a sentence where $advmod_1$ correctly removed an arc. Since the word *miauknął* (eng. meowed) has its own adverbial modifier *znowu* (eng. again) the enhanced arc to *obok* (eng. nearby) was removed.

When applied to the training data, this filter removed 105 enhanced arcs with an accuracy of 93%.

The filter *obj* is the only filter which removes arcs with label "obj". It applies when the enhanced "obj" modifier appears before the token in the sentence (see Figure 5a). The intuition is that in Polish "obj" modifiers tend to appear after both of the conjuncts. For example, in sentence *Podziwiali i doceniali ją też uczniowie* (id `train-s4812`; eng. Admired and appreciated her also students) the "obj" modifier *ją* (eng. her) appears after both of *Podziwiali* (eng. admired) and *doceniali* (eng. appreciated) and modifies both of them. In contrast, Figure 5b shows an example of a sentence where the filter



(a) The filter $advmod_1$: T has its own “advmod” dependent

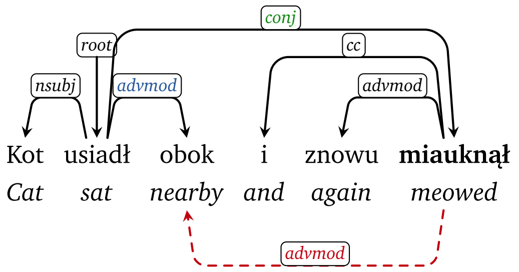
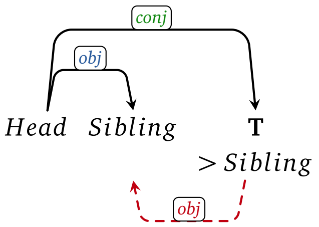

conj
root
cc
nsubj
advmod
advmod
Kot usiadł obok i znowu miauknął
Cat sat nearby and again meowed
advmod
(b) Example sentence (id `train-s6417`) where the filter $advmod_1$ correctly removes an enhanced arc

Figure 4: The filter $advmod_1$

obj correctly removed an arc. The rule $Children$ introduced an arc from the token *śpiewają* (eng. they-sing) to *ręce* (eng. hands). But since the word *ręce* appears before *śpiewają* the arc was removed.

When applied to the training data, this filter removed 854 enhanced arcs with an accuracy of 96%.



(a) The filter obj: sibling with label “obj” appears before T in the sentence

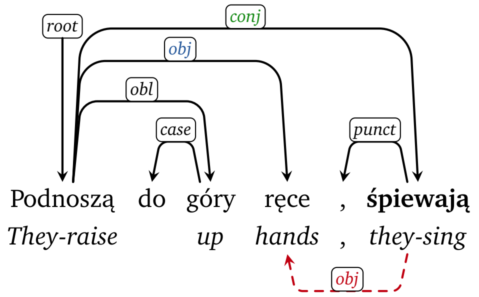

root
conj
obj
obl
case
punct
Podnoszą do góry ręce , śpiewają
They-raise up hands , they-sing
obj
(b) Example sentence (id `train-s12456`) where the filter obj correctly removes an enhanced arc

Figure 5: The filter obj

3.2. Evaluation of the Rules

In this section we evaluate the rules on the development set to test if they generalize well. As a baseline we use the system without any rules, i.e., we run the evaluation script on trees without any enhanced arcs.

We start with oracle experiments and apply the rules to gold-standard trees (see Table 4; Column 2). In this scenario the baseline achieves a very high accuracy of

94.23 ELAS. Adding the *Head* rule gives a big boost of almost 4 points, resulting in an ELAS of 98. As expected, the pure *Children* rule introduces too many incorrect arcs and considerably deteriorates the performance. All the consecutive filters (*labels*, $advmod_1$, *obj*) give small improvements, but together (see Table 4; the final row) they not only recover the drop caused by the *Children* rule but also improve the total accuracy by additional 0.73 points.

Next, we analyze the situation when enhanced arcs are introduced on automatically parsed trees. We apply the rules to outputs of two systems: the strongest parsing baseline GP and the full ensemble system BLEND-OPT. As expected, replacing gold-standard trees with a parser's predictions results in a big drop in performance: baseline accuracy decreases from 94.23 to 80.68 for GP and 81.83 for BLEND-OPT. Apart from the lower starting point, the rules behave similarly to the setting with gold-standard trees: *Head* gives a big boost, *Children* causes a big drop in accuracy, while the 12 rules together perform better than *Head* alone. Finally, comparing the accuracy of GP and BLEND-OPT shows that the parsing accuracy directly translates into enhanced parsing accuracy – BLEND-OPT surpasses GP by 1.28 in terms of LAS (cf. Table 3) and the advantage stays the same in terms of ELAS (1.31 points).

Table 4: Gains in enhanced parsing accuracy (ELAS) on the development set for incremental changes to the set of rules and different input trees

		Gold	**GP**	**BLEND-OPT**
No rules		94.23	80.68	81.83
+	*Head*	98.00	82.94	84.15
+	*Children*	93.19	78.12	79.25
–	*labels*	96.82	81.26	82.45
–	$advmod_1$	96.98	81.45	82.65
–	*obj*	97.37	81.84	83.05
12 rules		98.73	83.28	84.60

4. Test Results

The final results on the test set are shown in Table 5. In Subtask 1A we ranked second in terms of LAS score (83.82) and MLAS score (69.27) and were behind the COMBO team by 2.29 and 6.9 points respectively. We achieved the third best result in terms of BLEX score due to our poor lemmatization accuracy. In Subtask 1B we ranked first with an ELAS score of 81.90. Since we did not predict any semantic labels our SLAS score can be treated as a baseline result of running the evaluation script only on trees.

Table 5: Test results for all the systems participating in Task 1. The highest value in each column is bold

	LAS	**MLAS**	**BLEX**
COMBO	**86.11**	**76.18**	**79.86**
IMS	83.82	69.27	60.88
Poleval2k18	77.70	61.21	70.01
Drewutnia	27.39	18.12	25.24

(a) Subtask 1A: dependency parsing

	ELAS	**SLAS**
IMS	**81.90**	65.98
COMBO	80.66	**77.30**
Poleval2k18	66.73	67.84

(b) Subtask 1B: enhanced parsing

5. Conclusion

We have presented the IMS contribution to the PolEval 2018 Shared Task.

In Subtask 1A we re-used our system from the CoNLL 2017 UD Shared Task. We confirmed our previous findings that strong preprocessing, supertags, and the use of diverse parsers for blending are important factors influencing the parsing accuracy. We extended those findings to the PolEval treebank which was a new test case for the system. The treebank differs from traditional treebanks since it is mostly built from selected sentences containing difficult syntactic constructions, instead of being sampled from some source at random.

In Subtask 1B we extended the bulky ensemble system from Subtask 1A by a set of 12 simple rules predicting enhanced arcs. We showed that a successful rule-based augmentation strongly depends on the employed parsing system. As we have demonstrated, if perfect parsing is assumed (by using gold trees), the simple rules we have developed are able to achieve an extremely high ELAS, leaving little space for further improvements. However, since the rules are not built to handle parsing errors, the parsing accuracy directly translates into performance on predicting the enhanced arcs.

Acknowledgments

This work was supported by the Deutsche Forschungsgemeinschaft (DFG) via the SFB 732, project D8.

References

Björkelund A. and Nivre J. (2015). *Non-deterministic Oracles for Unrestricted Non-projective Transition-based Dependency Parsing*. [in:] *Proceedings of the 14th International Conference on Parsing Technologies*, pp. 76–86. Association for Computational Linguistics.

Björkelund A., Falenska A., Yu X. and Kuhn J. (2017). *IMS at the CoNLL 2017 UD Shared Task: CRFs and Perceptrons Meet Neural Networks*. [in:] *Proceedings of CoNLL 2017 Shared Task*, pp. 40–51.

Bohnet B. (2010). *Top Accuracy and Fast Dependency Parsing is not a Contradiction*. [in:] *Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010)*, pp. 89–97. COLING 2010 Organizing Committee.

Candito M., Guillaume B., Perrier G. and Seddah D. (2017). *Enhanced UD Dependencies with Neutralized Diathesis Alternation*. [in:] *Proceedings of the 4th International Conference on Dependency Linguistics (Depling 2017)*, pp. 42–53. Association for Computational Linguistics.

Chu Y. and Liu T. (1965). *On the Shortest Aborescence of a Directed Graph*. „Science Sinica", 14, pp. 1396–1400.

Edmonds J. (1967). *Optimum Branchings*. „Journal of Research of the National Bureau of Standards", 71(B), pp. 233–240.

Joshi A. K. and Bangalore S. (1994). *Disambiguation of Super Parts of Speech (or Supertags): Almost Parsing*. [in:] *Proceedings of the 15th Conference on Computational Linguistics — Volume 1* (COLING '94), pp. 154–160. Association for Computational Linguistics.

Müller T., Schmid H. and Schütze H. (2013). *Efficient Higher-Order CRFs for Morphological Tagging*. [in:] *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 322–332. Association for Computational Linguistics.

Nivre J., de Marneffe M.-C., Ginter F., Goldberg Y., Hajič J., Manning C., McDonald R., Petrov S., Pyysalo S., Silveira N., Tsarfaty R. and Zeman D. (2016). *Universal Dependencies v1: A multilingual Treebank Collection*. [in:] *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*, pp. 1659–1666. European Language Resources Association.

Ouchi H., Duh K. and Matsumoto Y. (2014). *Improving Dependency Parsers with Supertags*. [in:] *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*, pp. 154–158. Association for Computational Linguistics.

Potthast M., Gollub T., Rangel F., Rosso P., Stamatatos E. and Stein B. (2014). *Improving the Reproducibility of PAN's Shared Tasks: Plagiarism Detection, Author Identification, and Author Profiling*. [in:] Kanoulas E., Lupu M., Clough P., Sanderson M., Hall M., Hanbury A. and Toms E. (eds.), *Information Access Evaluation meets Multilinguality, Multimodality, and Visualization. 5th International Conference of the CLEF Initiative (CLEF 14)*, pp. 268–299, Berlin Heidelberg New York. Springer.

Sagae K. and Lavie A. (2006). *Parser Combination by Reparsing*. [in:] *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pp. 129–132. Association for Computational Linguistics.

Schuster S. and Manning C. D. (2016). *Enhanced English Universal Dependencies: An Improved Representation for Natural Language Understanding Tasks.* [in:] *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*, pp. 23–28. European Language Resources Association.

Straka M., Hajič J. and Straková J. (2016). *UDPipe: Trainable Pipeline for Processing CoNLL-U Files Performing Tokenization, Morphological Analysis, POS Tagging and Parsing*. [in:] *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*, pp. 4290–4297. European Language Resources Association.

Yu X. and Vu N. T. (2017). *Character Composition Model with Convolutional Neural Networks for Dependency Parsing on Morphologically Rich Languages*. [in:] *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 672–678. Association for Computational Linguistics.

Yu X., Falenska A. and Vu N. T. (2017). *A General-purpose Tagger with Convolutional Neural Networks*. „arXiv:1706.01723".

Zeman D., Popel M., Straka M., Hajič J., Nivre J., Ginter F., Luotolahti J., Pyysalo S., Petrov S., Potthast M., Tyers F., Badmaeva E., Gökırmak M., Nedoluzhko A., Cinková S., Hajič jr. J., Hlaváčová J., Kettnerová V., Urešová Z., Kanerva J., Ojala S., Missilä A., Manning C., Schuster S., Reddy S., Taji D., Habash N., Leung H., de Marneffe M.-C., Sanguinetti M., Simi M., Kanayama H., de Paiva V., Droganova K., Alonso H. M., Çöltekin Ç., Sulubacak U., Uszkoreit H., Macketanz V., Burchardt A., Harris K., Marheinecke K., Rehm G., Kayadelen T., Attia M., Elkahky A., Yu Z., Pitler E., Lertpradit S., Mandl M., Kirchner J., Alcalde F. H., Strnadová J., Banerjee E., Manurung R., Stella

A., Shimada A., Kwak S., Mendonça G., Lando T., Nitisaroj R. and Li J. (2017). *CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. [in:] *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pp. 1–19. Association for Computational Linguistics.

Drewutnia: a Frugal Approach to Dependency Parsing

Beata Skuczyńska (University of Warsaw)

Abstract

The paper presents system for dependency parsing of Polish sentences based on bidirectional GRU recurrent neural network. Drewutnia competed in PolEval 2018 1A task and achieved 4th place in ranking.

Keywords

dependency parsing, GRU, Natural Language Processing

1. Introduction

Dependency parsers, along with lemmatizers, part-of-speech taggers, named entity recognizers, etc., are considered to be basic natural language processing tools. Therefore they must be fast and domain-independent. Fields where they might be helpful are, for instance, text summarization, machine translation or question-answering systems.

The aim of dependency parsing task is to obtain the predicate-argument structure of the sentence. Example of a sentence analyzed in such a way is depicted in Figure 1. Tokens are paired, each of them (but one) has assigned parent (governor) and a label (ex. *subj*) that describes the relation in pair. The exception is a root token (labeled with *root*) which doesn't have parent. More theoretical discussion on conception of dependency trees could be found in (Wróblewska 2014).

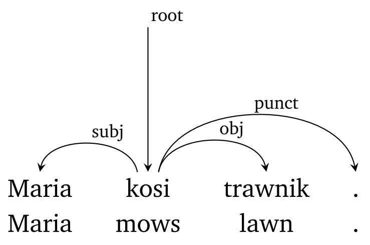


Figure 1: Example of dependency tree for Polish sentence

2. Previous Work

Vast majority of currently used and developed dependency parsers are statistical. There are two main approaches to create such parsers: graph-based (ex. MATE parser; Bohnet 2010) and transition-based (ex. MaltParser; Nivre et al. 2007). The former creates a set of candidate trees, scores them upon the previously trained model and returns the one that achieved the highest result. The latter deterministically parses sentence, building dependency structure based on transitions (shift-reduce actions) predicted by a classifier. Both models are available for Polish language as trained on Polish Dependency Bank[1]. The last published results of their performance (LAS score, see Section 5) are 0.85 for Mate parser and 0.82 for Malt parser (Wróblewska 2018). However, growing popularity of usage of neural networks in natural language processing reached also dependency parsing problem. In this year's edition of CoNLL Shared Task at least half of the submitted systems are making use of various neural architectures (Zeman and Hajič 2018b).

3. Data

Dataset used to train and evaluate presented system is the one provided by PolEval 2018 organizers (Ogrodniczuk and Kobyliński 2018). It is annotated according to Universal Dependencies guideline and kept in CONLL-U format (Zeman and Hajič 2018c). The aim of Drewutnia was only to predict labels and parent of tokens, so the input sentences were preprocessed i.e. splitted, tokenized and tagged with universal part-of-speech tags, Polish-specific tags, morphological features, and lemmas of particular tokens.

[1]http://zil.ipipan.waw.pl/PDB

4. System Description

Drewutnia's pipeline consists of a few steps. The preparation of data is described in subsection 4.1. The neural network design is presented in subsection 4.2 and its output in subsection 4.3. Finally, important postprocessing steps are described in subsection 4.4.

4.1. Preprocessing

The input to the system is a text file in CONLL-U format (Zeman and Hajič 2018c). Each token is transformed to one-hot vector where one indicates the column dedicated for its part-of-speech tag. Each sentence's number of tokens is normalized to an arbitrary value. All experiments and evaluation in this paper are performed with sentences of 50 tokens size, but this is a configurable parameter of system. If the sentence is longer than this value, then it's divided to 50 tokens chunks – if it's shorter then there are added padding rows. Additionally, every vector in sentence as a last value has assigned whether is a sentence's token or padding row.

4.2. Neural Network Architecture

System's architecture is based on bidirectional recurrent GRU (Cho et al. 2014) network with dropout layer. All experiments and evaluation process was conducted with 400 GRU hidden units, although the number of it is configurable parameter of the system. Drewutnia was implemented in Python, using the Keras[2] package with TensorFlow[3] backend. Its schematic depiction is shown in Fig. 2.

As an input serves set of matrixes representing sentences of normalized length with one-hot encoded part-of-speech tags (see 4.1). As an output there are two layer with sigmoid activation. One is for predicting id of token's parents and the other one is for relation label (see 4.3).

4.3. Predicted Output

Both output layers store the information in one-hot encoded vectors. Labels are encoded similarly as input layer of part-of-speech tags – every possible tag has own column and the last column serves as indicator of padding. Parents values are straightforward – they are indicating column, whose index is the index of token's parent in

[2]https://keras.io/

[3]https://www.tensorflow.org/

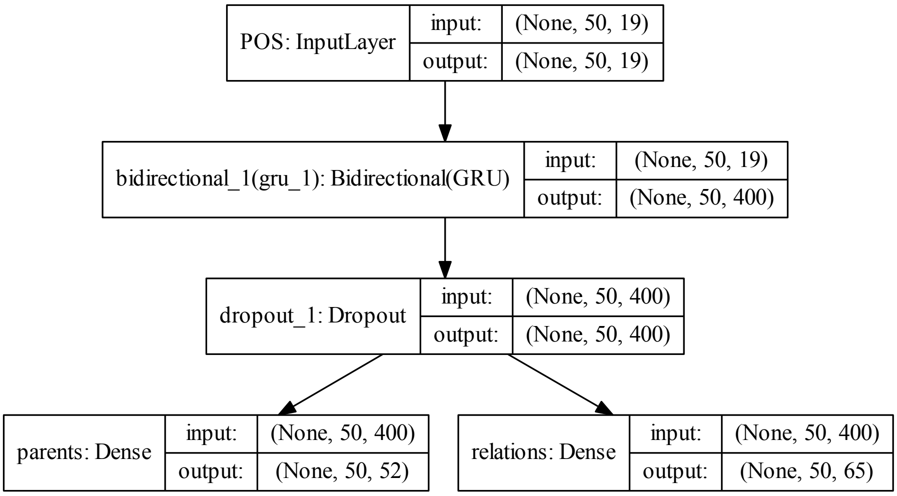


Figure 2: Schematic depiction of Drewutnia's neural network architecture

sentence. Parents output matrix has two extra columns. The first one is reserved for root as its parent is always indicating zero which is out of sentence. The second bonus column is for padding.

4.4. Postprocessing

During this phase predicted output is translated to CONLL-U format and there are fixed incorrectly builded trees. After removing padding and transforming sentences to their original length, there is checked whether they have exactly one root. If has none, then the output matrix of neural network is checked again to see which token had the biggest score in column indicating root. If there is more than one root token, then it is checked which of them had the biggest score in root column. Rest of tokens that had root label are given chosen root word as a parent. Also, the tokens without assigned parent are treated the same as rejected roots.

Another step of postprocessing is removing cycles from predicted tree. It is performed with using simple function over mapped vertices of dependency graph. If the cycle is detected, then parents of all tokens (except root) are set on root.

Table 1: Results of evaluating different configurations of the system

Number of epochs	**Removing cycles in postprocessing**	**LAS F_1 score**
30	NO	48.02
30	YES	29.54
50	NO	31.06
50	YES	17.86

5. Experiments and Evaluation

Evaluated test set contained 2219 sentences (see Section 3). The evaluation was performed with usage of an adapted version of the CoNLL 2018 Shared Task script (Zeman and Hajič 2018a). The main introduced change was not breaking evaluation in case if:

1. sentence was having multiple roots,
2. token's parent was pointing outside of the tree,
3. there was cycle in tree.

Instead, those issues were counted and provided as supplementary evaluation statistics. Additionally, token who was found to be in 2nd situation was excluded from further processing.

In Table 1 are presented results of evaluation. LAS metric counts percentage of the words who were ascribed both good parent id and appropriate relation label. Details could be found in (Zeman and Hajič 2018a).

6. Discussion

Results presented in Table 1 are far below the state-of-the-art systems. Yet, considering that the input for the model were only part-of-speech tags, LAS F_1 score on level of 48.02 seems not a bad achievement. The approach realized in this system was purely exploration and experimental. Most of the contemporary Natural Language Processing tools based on neural networks makes use of word embeddings (see e.g. Liu et al. 2017) for question-answering system or (Guzmán et al. 2017) in machine translation problem). Although they generally improve performance, usually the models are quite space consuming and therefore difficult to adapt to devices with limited memory space. Comparing to that Drewutnia's model was only around 3 MB size.

The described system shows that dependency parsing restrictions is challenging for simple seq-to-seq model. Limiting number of roots to one and preventing showing of cycles in sentences are bounds that are hard to incorporate in neural model. The method of dealing with cycles described in Subsection 4.4 significantly decreases the performance of the system.

7. Conclusions

In this paper was presented experimental frugal approach to dependency parsing of Polish sentences. GRU neural network with only part-of-speech tags on input proved to be insufficient for such a complex task. Drewutnia need to undergo significant improvements in order to become competitive for the state-of-art systems.

References

Bohnet B. (2010). *Very High Accuracy and Fast Dependency Parsing is not a Contradiction*. [in:] *Proceedings of 23rd International Conference on Computational Linguistics (COLING 2010)*, vol. 2, pp. 89–97. Association for Computational Linguistics.

Cho K., van Merrienboer B., Gülçehre Ç., Bougares F., Schwenk H. and Bengio Y. (2014). *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*. „CoRR", abs/1406.1078.

Guzmán F., Joty S., Màrquez L. and Nakov P. (2017). *Machine Translation Evaluation with Neural Networks*. „Computer Speech & Language", 45, pp. 180–200.

Liu X., Shen Y., Duh K. and Gao J. (2017). *Stochastic Answer Networks for Machine Reading Comprehension*. „CoRR", abs/1712.03556.

Nivre J., Hall J., Nilsson J., Chanev A., Eryigit G., Kübler S., Marinov S. and Marsi E. (2007). *MaltParser: A Language-independent System for Data-driven Dependency Parsing*. „Natural Language Engineering", 13(2), pp. 95–135.

Ogrodniczuk M. and Kobyliński Ł. (2018). *PolEval 2018 Site*. `http://poleval.pl/tasks/`. Last accessed 30 Sept 2018.

Wróblewska A. (2014). *Polish Dependency Parser Trained on an Automatically Induced Dependency Bank*. Ph.D. dissertation, Institute of Computer Science, Polish Academy of Sciences.

Wróblewska A. (2018). *PDBparser Page*. `http://zil.ipipan.waw.pl/PDB/PDBparser`. Last accessed 30 Sept 2018.

Zeman D. and Hajič J. (2018a). *CoNLL 2018 Shared Task Evaluation Site*. `http://universaldependencies.org/conll18/evaluation.html`. Last accessed 30 Sept 2018.

Zeman D. and Hajič J., eds. (2018b). *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Association for Computational Linguistics.

Zeman D. and Hajič J. (2018c). *Universal Dependencies CONLL-U Page*. `http://universaldependencies.org/format.html`. Last accessed 30 Sept 2018.

Semi-Supervised Neural System for Tagging, Parsing and Lemmatization. Addendum

Piotr Rybak and Alina Wróblewska (Institute of Computer Science, Polish Academy of Sciences)

This addendum to the paper (Rybak and Wróblewska 2018) is devoted to the winner of the 1A shared task of the PolEval 2018 competition (Ogrodniczuk and Kobyliński 2018). The main paper presents the system COMBO,[1] which successfully participated in the CoNLL 2018 shared task on Multilingual Parsing from Raw Text to Universal Dependencies (Zeman et al. 2018). The current addendum describes some COMBO's extensions: adding mechanisms to predicting enhanced graphs and semantic labels.

The COMBO system used in the task 1A consists of jointly trained tagger, lemmatizer, and dependency parser which are based on the features extracted by the bidirectional long-short term memory network (biLSTM) taking the concatenation of external word embeddings and internal character-based word embeddings as input. The tagger takes the extracted features and predicts universal part-of-speech tags, Polish-specific tags and morphological features using three separate fully connected neural networks with one hidden layer. The lemmatizer uses a dilated convolutional neural network to predict lemmas based on characters of corresponding words and the features extracted by the biLSTM encoder. As a scoring function, the graph-based dependency parser uses simple dot product of the vector representations of a dependent and its governor. These representations are output by two single fully connected layers which take the feature vectors extracted by the biLSTM encoder as input.

In the task 1B, the extended version of COMBO predicts enhanced graphs and semantic roles of some dependents. The enhanced graphs are predicted in the similar way as the dependency trees in the task 1A. First, the adjacency matrix is calculated for an enhanced graph, i.e. the embeddings are computed for the heads and their dependents based on the features extracted by the biLSTM encoder and dot product is used as a scoring function. The only difference is usage of the sigmoid activation

[1] `https://github.com/360er0/COMBO`

function instead of soft-max. The sigmoid activation function allows the network to predict many heads for a given dependent. Next, we calculate another set of head and dependent embeddings. For each pair of a head and its dependent we concatenate their embeddings and use a fully connected neural network with one hidden layer to predict the label of such hypothetical arc. In this case, the soft-max activation function is used to force the network to predict only one label for each arc. Finally, the vector of probabilities of various labels is multiplied by the probability of the given arc taken from the first part of the model. Both parts are jointly optimised using cross-entropy loss.

The semantic label are predicted in a way that resembles the prediction of part-of-speech tags. The system takes the features extracted by the biLSTM encoder as input and predicts semantic roles for some dependents using a fully connected neural networks with one hidden layer.

COMBO is trained on data provided for the purpose of the PolEval dependency shared task (Wróblewska 2018).[2] It also uses the pre-trained word embeddings.[3] COMBO is the winner of the 1A shared task of the PolEval 2018 competition. It achieves the highest F_1-scores in all categories. In the task 1B, COMBO outperforms other systems in predicting semantic labels. In predicting enhanced graphs, it performs only slightly worse than the best IMS system.

Acknowledgements

The research presented in this paper was founded by SONATA 8 grant no 2014/15/D/HS2/03486 from the National Science Centre Poland and by the Polish Ministry of Science and Higher Education as part of the investment in the CLARIN-PL research infrastructure.

References

Ogrodniczuk M. and Kobyliński Ł., eds. (2018). *Proceedings of the PolEval 2018 Workshop*. Institute of Computer Science, Polish Academy of Sciences.

Rybak P. and Wróblewska A. (2018). *Semi-Supervised Neural System for Tagging, Parsing and Lematization*. [in:] *Proceedings of the CoNLL 2018 Shared Task: Mul-*

[2] http://git.nlp.ipipan.waw.pl/alina/PDBUD
[3] http://mozart.ipipan.waw.pl/~axw/models

tilingual Parsing from Raw Text to Universal Dependencies, s. 45–54. Association for Computational Linguistics.

Wróblewska A. (2018). *Extended and Enhanced Polish Dependency Bank in Universal Dependencies Format*. [in:] *Proceedings of Universal Dependencies Workshop 2018* (UDW 2018).

Zeman D., Hajič J., Popel M., Potthast M., Straka M., Ginter F., Nivre J. and Petrov S. (2018). *CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. [in:] *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pp. 1–21. Association for Computational Linguistics.

Results of the PolEval 2018 Shared Task 2: Named Entity Recognition

Aleksander Wawer, Estera Małek (Institute of Computer Science, Polish Academy of Sciences)

1. Introduction

PolEval is a SemEval-inspired evaluation campaign for natural language processing tools for Polish. Submitted tools compete against one another within certain tasks, using available data and are evaluated according to pre-established procedures. This article describes the results of Named Entity Recognition task.

Named entities (NE) are phrases that contain the names of things, such as persons, organizations and locations. Named Entity Recognition (NER) task is to label sequences of words in a text with appropriate named entity markup. Named entity recognition is an important task of information extraction systems.

For example the sentence:

> [PER Ban Ki-moon] przebywał z wizytą w [LOC Gdańsku].

contains two named entities: Ban Ki-moon is a person [PER], Gdańsk is a location [LOC].

Entity lemmatization is not a part of this task.

There has been a lot of work on named entity recognition, especially for English. The Message Understanding Conferences (MUC), CoNLL-2002 and CoNLL-2003 (Tjong Kim Sang and De Meulder 2003) have offered the opportunity to evaluate systems for English on the same data in a competition. They have also produced schemes for entity annotation. The PolEval Named Entity Recognition task aims to bridge this gap and provide a reference data set for the Polish language.

Annotating conventions of both training and testing data follow the National Corpus of Polish (NKJP). For a discussion of named entity annotation in NKJP, please refer to chapter 9 by Przepiórkowski et al. (2012). Please note that NKJP supports nested entity annotation and multiple sub-categories (three sub-types of persons, five sub-types of places, see Figure 9.2, p. 133).

2. Data

2.1. Training Data

As the training data set, we encouraged participants to use the manually annotated 1-million word subcorpus of the National Corpus of Polish. It is available from `http://clip.ipipan.waw.pl/NationalCorpusOfPolish`.

2.2. Evaluation Data

In order to test the submitted systems, we have prepared a corpus with manual named entity annotations. This data set has not been publicly released before the PolEval event. Texts in this corpus originate in the Polish coreference corpus.

The corpus is composed of 1828 documents with different length, extracted randomly from National Corpus of Polish (NKJP).

The annotation process was conducted according to the ontology from National Corpus of Polish manual. The ontology precisely describes most types of Named Entities and assigns them to certain categories. However, there are cases of Named Entities in corpus that are not covered by instructions. The reasons are various – specific Named Entity's context, difficulties in interpretation, lack of one decision about specific Named Entity type annotation in ontology (possible two ways of annotation), multiple nested entities, etc.

In those cases annotators' strategy was to choose the best possible annotation for each Named Entity by analogy to the ontology instructions. It was also helpful to check how certain Named Entity is annotated in 1-million word subcorpus of the National Corpus of Polish (described in Section 2.1) or check if certain Named Entity (or a similar one) appears in examples in the annotation manual.

Below we include some examples of Named Entities that were difficult to annotate.

Uncommon nicknames:

> Janusz i **Nieznajomy** znają dokładne daty (...)
> A teraz śpiewała to **Urszula** na "Przystanku Woodstock"

Uncommon way of writing of a particular Named Entity:

> **#PosełAleksanderKwasniewski**

Metaphorical use of a Named Entity:

> Każdy ma swoje **Kilimandżaro**.

Named Entity that does not refer directly to any of possible category:

> Otrzymał tytuł Filantropa Roku 2001 oraz statuetkę **Ferdynanda**. Co wytrwalsi mogli jeszcze bezpośrednio po balu okrążać, skacząc na jednej nodze, fontannę przy ratuszu z **Jackiem**.

Named Entities in languages different than Polish or English:

> Mistrzem Argentyny (...) został klub **San Lorenzo de Almagro Buenos Aires**

3. Solutions

This section briefly describes submitted solutions. The list is consists of descriptions provided by authors of the competing systems. Therefore, we do not list systems where such descriptions were not provided.

3.1. KNER

KNER utilizes recurrent and convolutional neural networks combined with conditional random fields. The features are provided by the morphological tagger KRNNT (Wróbel 2017). Word embeddings are initialized with embeddings trained on full National Corpus of Polish (Przepiórkowski et al. 2012, Mykowiecka et al. 2017). Nested entities are addressed by two approaches. Sent testing data solution contained a mistake causing 12 percentage points drop of F_1.

3.2. BiLSTM-CRF

The specific distribution of entities in the National Corpus of Polish encouraged us to handle the problem in a following way: to train separate, classic BiLSTM-CRF models (Lample et al. 2016) per (almost) non-overlaping entity groups, that is groups guaranteeing it is at least highly unlikely entities within will collide. Whenever possible, groups consisted of neighboring entities in order to exploit the potential of linear CRF chain. The best-performing solution we tested relied on stacked GloVe and Contextual String Embeddigns. The later were recently proposed by Akbik et al. (2018), who showed that the internal states of a trained character language model can be used to create word embeddings able to outperform the previous state-of-the-art in sequence labeling tasks. Forward and backward character-level language models were trained on 1B words corpus of Polish composed in one third of respectively subsamples from: Polish Wikipedia, PolEval's language modeling task and Polish Common Crawl. Subsamples of Wikipedia and PolEval tasks were selected randomly, whereas from Common Crawl those sentences were selected, whose were characterized with the highest similarity to PolEval sample, as expressed with cross-entropy (Moore and Lewis 2010). GloVe embeddings were trained on a very large, freely available Common Crawl-based Web corpus of Polish. After postprocessing, the corpus consisted of 27 354 330 800 tokens, from which 119 330 367 were unique. Embeddings were generated for all the tokens present in PolEval task's corpora. The final LSTM-CRF sequence labelling models were trained with one bidirectional LSTM layer and 512 hidden states on 300-dimensional GloVe embeddings, as well as embeddings from forward and backward LMs with 2048 hidden states.

3.3. Liner2

Liner2[1] (Marcińczuk et al. 2013) is an open-source generic framework for sequence text labeling. It has been already used to train a state-of-the-art models for named entity recognition following the KPWr guidelines (Marcińczuk et al. 2016), including mention boundary detection, coarse-grained categorization (9 types) and fine-grained categorization (82 types) (Marcińczuk et al. 2017). It was used also to other Information Extraction tasks like the recognition of temporal expressions (Kocoń and Marcińczuk 2017) and events (Kocoń and Marcińczuk 2016). Liner2 uses Conditional Random Fields (Sutton and McCallum 2007) as a statistical model and a rich space of features of different types, including: orthographic, structural, morphological, lexicon-based, wordnet-based and compound features. Liner2 exploits language and

[1] `https://github.com/CLARIN-PL/Liner2`

domain knowledge in a form of external structured resources, i.e. lexicon of named entities, lexicon of named entity triggers, wordnet and morphological dictionary.

3.4. PolDeepNer

PolDeepNer is based on an ensemble of neural networks. The first one is a bidirectional LSTM with sequential conditional random layer above it, similarly as it was presented by Lample et al. (2016). The second one utilizes bidirectional GRU layer (Cho et al. 2014), also together with CRF layer. Both LSTM and GRU are units of the recurrent neural network (RNN), where in case of bidirectional RNN, context of the word is captured through past and future words (Sahu and Anand 2016). The neural networks are fed with three different word embedding models generated with fastText (Bojanowski et al. 2017) from the following corpora: Common Crawl for Polish[2], Polish Wikipedia[3] and KGR10[4].

3.5. simple_ner

This named entity recognition system created for PolEval competition uses Bidirectional GRU neural network, trained on 1-million word subcorpus from the National Corpus of Polish, with 300 dimensions Polish word embeddings and binary features as an input and flat entities as an output.

Three types of binary features were used to describe if each word starts with a capital letter, has a dot or a number inside. Two versions of processed output were created to achieve flat structure. In the first type all overlapping labels in nested entity were modified by concatenating it into one label. In the second one, only the most specific label in the set were used which resulted in removal of all nesting.

3.6. P.g. LSTM-CRF w. C. S. E.

Full name of the submitted system is "Per group LSTM-CRF with Contextual String Embeddings".

The specific distribution of entities in the National Corpus of Polish encouraged us to handle the problem in a following way: to train separate, classic BiLSTM-CRF models (Lample et al. 2016) per (almost) non-overlaping entity groups, that is groups guaranteeing it is at least highly unlikely entities within will collide. Whenever

[2]`http://commoncrawl.org/`
[3]`https://www.wikipedia.org/`
[4]`https://clarin-pl.eu/dspace/handle/11321/600`

possible, groups consisted of neighboring entities in order to exploit the potential of linear CRF chain. The best-performing solution we tested relied on stacked GloVe and Contextual String Embeddigns. The later were recently proposed by Akbik et al. (2018), who showed that the internal states of a trained character language model can be used to create word embeddings able to outperform the previous state-of-the-art in sequence labeling tasks. Forward and backward character-level language models were trained on 1B words corpus of Polish composed in one third of respectively subsamples from: Polish Wikipedia, PolEval's language modeling task and Polish Common Crawl. Subsamples of Wikipedia and PolEval tasks were selected randomly, whereas from Common Crawl those sentences were selected, whose were characterized with the highest similarity to PolEval sample, as expressed with cross-entropy (Moore and Lewis 2010). GloVe embeddings were trained on a very large, freely available Common Crawl-based Web corpus of Polish. After postprocessing, the corpus consisted of 27 354 330 800 tokens, from which 119 330 367 were unique. Embeddings were generated for all the tokens present in PolEval task's corpora. The final LSTM-CRF sequence labelling models were trained with one bidirectional LSTM layer and 512 hidden states on 300-dimensional GloVe embeddings, as well as embeddings from forward and backward LMs with 2048 hidden states.

3.7. VIA_NER

The proposed solution is a bidirectional LSTM-CRF neural network inspired by previous work in this area (Ma and Hovy 2016). As input to the network, the authors used externally trained sub-word vectors derived from a Common Crawl dump of Polish texts (Grave et al. 2018), convolutions of character embeddings and selected morphosyntactic features of word tokens obtained from the APT tagger of Polish (Pęzik and Laskowski 2017). The architecture was implemented in the Tensorflow framework.

4. Evaluation Procedure

We intend to evaluate systems according to two general approaches:

— *Exact match of entities*: in order to count as a true positive, offsets of detected vs golden entities need to be the same.

— *Entity overlap*: in order to count as a true positive, offsets of detected vs golden entities need to have a common range, such as a word or more words.

In both cases, in order to count as a true positive, detected category of an entity need to be the same as the golden one.

Final ranking of participating systems was based on a combined measure of the exact match and entity overlap.

The metrics we used for this task is micro F_1.

5. Results

None of the systems has been declared as using other named entity annotated corpus than the official 1M NKJP, therefore we present the competing systems in just one table.

Table 1 below shows results of evaluation. It contains system names, initials of submitting authors, micro average F_1 scores computed for overlap and exact matches. We computed final scores as weighted means:

$$0.8 * overlap + 0.2 * exact$$

The reason for this are (1) combining both measures while (2) giving strong premium for overlap matches. Hopefully, both overlap and exact matches produce very similar orderings of competing systems, even without using combined weighted scores as in the column Final.

Table 1: Final results of PolEval 2018 NER competition

System	**Author**	**Exact**	**Overlap**	**Final**
P.g. LSTM-CRF w. C. S. E.	[Ł. B.]	0.826	0.877	0.866
PolDeepNer	[M. M.]	0.822	0.859	0.851
Liner2	[M. M.]	0.778	0.818	0.810
OPI_Z3	[S. D.]	0.749	0.805	0.793
joint	[M. L.]	0.748	0.789	0.780
disjoint	[M. L.]	0.747	0.788	0.779
via_ner	[P. P.]	0.692	0.773	0.756
kner_sep	[K. W.]	0.7	0.742	0.733
Poleval2k18	[M. Z.]	0.623	0.743	0.719
KNER	[K. W.]	0.681	0.719	0.711
simple_ner	[P. Ż.]	0.569	0.653	0.636

The best solution turned to be *P.g. LSTM-CRF w. C. S. E.* ("Per group LSTM-CRF with Contextual String Embeddings") from Applica.AI. It was followed closely by two solutions from Wrocław University of Technology: *PolDeepNer* and *Liner2*.

6. Conclusions

Altogether, in the NER task in 2018 we had eleven competing systems, submitted by nine teams or researchers.

Most of the solutions were based on deep learning paradigm, especially recurrent neural networks. Some exceptions include Liner2, based on an older solution of Conditional Random Fields (CRF).

The systems were fine-tuned to various degree, and that was reflected in achieved scores. We had relatively straightforward solutions such as *simple_ner*, as well as heavily tuned and complex ones such as the winning *P.g. LSTM-CRF w. C. S. E.* ("Per group LSTM-CRF with Contextual String Embeddings"). The notion of fine-tuning involves in this case also transfer learning on huge resources, as was the case with the winning solution (glove embeddings trained on Common Crawl). The solutions based on recurrent neural networks used either softmax or conditional random fields for the inference layer.

The results are very good, but the performance of named entity extraction in the English language is still higher. Current state-of-the-art systems reach F_1 scores above 0.9. For example Lample et al. (2016) report F_1 at 0.9094, Chiu and Nichols (2015) report F_1 at 0.9162. The difference in performance between the best Polish and English systems can be very likely attributed to much simpler taxonomy of the CONLL-2003 data set used in the English language. It consists only of one level PLO markup (Person, Location, Organization).

References

Akbik A., Blythe D. and Vollgraf R. (2018). *Contextual String Embeddings for Sequence Labeling*. [in:] *COLING 2018, 27th International Conference on Computational Linguistics*, pp. 1638–1649.

Bojanowski P., Grave E., Joulin A. and Mikolov T. (2017). *Enriching Word Vectors with Subword Information*. „Transactions of the Association for Computational Linguistics", 5, pp. 135–146.

Chiu J. P. C. and Nichols E. (2015). *Named Entity Recognition with Bidirectional LSTM-CNNs*. „CoRR", abs/1511.08308.

Cho K., van Merrienboer B., Bahdanau D. and Bengio Y. (2014). *On the Properties of Neural Machine Translation: Encoder-Decoder Approaches*. [in:] *Proceedings of the 8th Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8), 2014*.

Grave E., Bojanowski P., Gupta P., Joulin A. and Mikolov T. (2018). *Learning Word Vectors for 157 Languages*. [in:] Calzolari N., Choukri K., Cieri C., Declerck T., Goggi S., Hasida K., Isahara H., Maegaard B., Mariani J., Mazo H., Moreno A., Odijk J., Piperidis S. and Tokunaga T. (eds.), *Proceedings of the 11th International Conference on Language Resources and Evaluation (LREC 2018)*, pp. 3483–3487. European Language Resources Association.

Kocoń J. and Marcińczuk M. (2016). *Generating of Events Dictionaries from Polish WordNet for the Recognition of Events in Polish Documents*, *Lecture Notes in Computer Science vol. 9924*, pp. 12–19. Springer International Publishing, Cham.

Kocoń J. and Marcińczuk M. (2017). *Supervised Approach to Recognise Polish Temporal Expressions and Rule-based Interpretation of Timexes*. „Natural Language Engineering", 23(3), pp. 385–418.

Lample G., Ballesteros M., Subramanian S., Kawakami K. and Dyer C. (2016). *Neural Architectures for Named Entity Recognition*. „arXiv:1603.01360".

Ma X. and Hovy E. (2016). *End-to-end Sequence Labeling via Bi-directional LSTM-CNNS-CRF*. „arXiv:1603.01354".

Marcińczuk M., Kocoń J. and Janicki M. (2013). *Liner2 – a Customizable Framework for Proper Names Recognition for Polish*. [in:] Bembenik R., Skonieczny L., Rybiński H., Kryszkiewicz M. and Niezgódka M. (eds.), *Intelligent Tools for Building a Scientific Information Platform*, pp. 231–253.

Marcińczuk M., Oleksy M. and Dziob A. (2016). *KPWr Annotation Guidelines — Named Entities*. CLARIN-PL digital repository.

Marcińczuk M., Kocoń J. and Oleksy M. (2017). *Liner2 — a Generic Framework for Named Entity Recognition*. [in:] Erjavec T., Piskorski J., Pivovarova L., Snajder J., Steinberger J. and Yangarber R. (eds.), *Proceedings of the 6th Workshop on Balto-Slavic Natural Language Processing (BSNLP@EACL 2017)*, pp. 86–91. Association for Computational Linguistics.

Moore R. C. and Lewis W. (2010). *Intelligent Selection of Language Model Training Data*. [in:] *Proceedings of the ACL 2010 Conference Short Papers*, ACLShort '10, pp. 220–224. Association for Computational Linguistics.

Mykowiecka A., Marciniak M. and Rychlik P. (2017). *Testing Word Embeddings for Polish*. „Cognitive Studies / Études Cognitives", 17, pp. 1–19.

Przepiórkowski A., Bańko M., Górski R. L. and Lewandowska-Tomaszczyk B., eds. (2012). *Narodowy Korpus Języka Polskiego*. Wydawnictwo Naukowe PWN, Warsaw.

Pęzik P. and Laskowski S. (2017). *Evaluating an Averaged Perceptron Morphosyntactic Tagger for Polish*. [in:] *Human Language Technologies as a Challenge for Computer Science and Linguistics*, pp. 372–376, Poznań.

Sahu S. and Anand A. (2016). *Recurrent Neural Network Models for Disease Name Recognition Using Domain Invariant Features*. [in:] *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2216–2225. Association for Computational Linguistics.

Sutton C. and McCallum A. (2007). *An Introduction to Conditional Random Fields for Relational Learning*. [in:] Getoor L. and Taskar B. (eds.), *Introduction to Statistical Relational Learning*. MIT Press.

Tjong Kim Sang E. F. and De Meulder F. (2003). *Introduction to the CoNLL-2003 Shared Task: Language-independent Named Entity Recognition*. [in:] *Proceedings of the 7th Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4* (CONLL '03), pp. 142–147. Association for Computational Linguistics.

Wróbel K. (2017). *KRNNT: Polish Recurrent Neural Network Tagger*. [in:] Vetulani Z. and Paroubek P. (eds.), *Proceedings of the 8th Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics*, pp. 386–391. Fundacja Uniwersytetu im. Adama Mickiewicza w Poznaniu.

Approaching Nested Named Entity Recognition with Parallel LSTM-CRFs

Łukasz Borchmann, Andrzej Gretkowski, Filip Graliński (APPLICA.AI)

Abstract

We present the winning system of this year's PolEval nested named entity competition, as well as the justification of handling the particular problem with multiple models rather than relying on dedicated architectures. The description of working out the final solution (parallel LSTM-CRFs utilizing GloVe and Contextual Word Embeddings) is preceded with information regarding recent advances in flat and nested named entity recognition. Significantly, all the tested solutions were developed on the basis of open source implementations, particularly Flair framework, LM-LSTM-CRF, Layered-LSTM-CRF and Vowpal Wabbit.

Keywords

nested named entity, named entity recognition, LSTM-CRF, contextual word embeddings, Polish NER, GloVe embeddings

1. Introduction

Named entity recognition (or entity identification, entity chunking, entity extraction) is a task of locating and classifying spans of text associated with real-world objects, such as person names, organizations and locations, as well as with abstract temporal and numerical expressions (eg. dates).

1.1. Flat Named Entity Recognition

As Young et al. (2017) summarize, after decades of *machine learning approaches utilizing shallow models trained on high dimensional and sparse features*,[1] came time of neural networks based on dense vector representations. It is also the case for named entity recognition systems, where those relying on hand-crafted features and domain-specific resources can be outperformed with simple deep learning frameworks.[2]

Many modern and successful NER solutions follow Huang et al. (2015) and Lample et al. (2016) approaching task with bidirectional LSTM-CRF architecture, which proved to be a strong candidate for structured prediction problems.

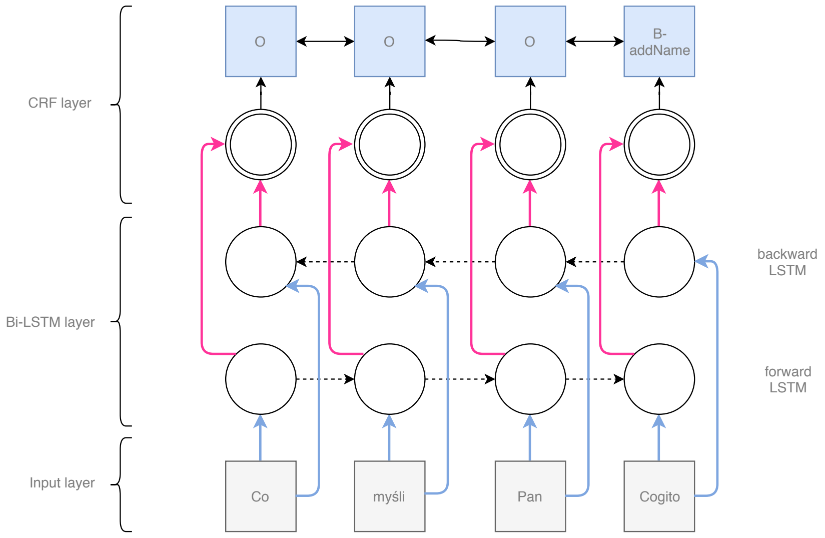


Figure 1: BiLSTM-CRF architecture (Huang et al. 2015, Lample et al. 2016)

Table 1 presents the results of the selected LSTM-CRF-based solutions in the CoNLL 2003 NER task. Liu et al. (2018) showed that LSTM-CRF architecture can be empowered by training a character-level language model at the same time, in addition to the sequence labeling model. Recent approaches by Peters et al. (2018) and Akbik et al. (2018) use embeddings obtained from internal states of deep language models pre-trained on a large text corpus. These are expected to capture context-dependent word semantics.

A common approach is to stack conceptually different embeddings, eg. by concatenating LM's embeddings with count-based approaches of obtaining vector representations

[1]Cf. eg. (Nadeau and Sekine 2007) for a review of pre-neural solutions.

[2]There are, however, also some attempts to incorporate domain-specific knowledge, eg. by injecting it into word embeddings (Celikyilmaz et al. 2015, Pandey et al. 2017).

Table 1: Results of selected LSTM-CRF-based solutions in the CoNLL 2003 NER task

Method	**Span F_1**
Contextual string embeddings (Akbik et al. 2018)	93.09
Deep contextualized word representations (Peters et al. 2018)	92.22
Task-aware neural language model (Liu et al. 2018)	91.71
Classic LSTM-CRF (Lample et al. 2016)	90.94

for words, such as GloVe proposed by Pennington et al. (2014). According to the distributional hypothesis, *difference of meaning correlates with difference of distribution* (Harris 1954), that is words sharing context tent to share similar meanings, which is often perceived as theoretical justification of the former representations.

The current state-of-the-art was established by Akbik et al. (2018) with contextualized string embeddings stacked with GloVe embeddings for English and fastText embeddings for German language (Bojanowski et al. 2017).

1.2. Nested Entity Identification

The methods described above receive particular attention of researchers and are the basis of related nested named entity recognition systems, where it is expected that named entities can overlap and contain other named entities. Figure 2 presents an example of such coming from the National Corpus of Polish (Przepiórkowski et al. 2012), namely street name (here classified as *geogName*), consisting of a person name (*persName*), containing *forename* and *surname*.

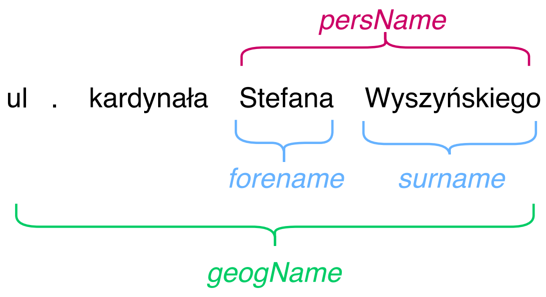


Figure 2: Example of nested named entity from the National Corpus of Polish (*ul. kardynała Stefana Wyszyńskiego* 'Cardinal Stefan Wyszyński Street')

These were proposed to be handled in multiple ways, whereas many of them rely on an old paradigm of handcrafted features, such as cascaded CRF model, constituency parser with constituents for each named entity or mention hypergraph model (Katiyar and Cardie 2018). Recently however, the problem was successfully addressed with neural architectures, by dynamically stacking additional flat CRF layers in LSTM-CRF model (Ju et al. 2018) and learning the entity hypergraph structure (Katiyar and Cardie 2018).

1.3. PolEval Entity Extraction Task

PolEval is an example of nested named entity recognition tasks. Participants were asked to train their models on 1M subcorpus of the National Corpus of Polish, consisting of around 87k entities with 14 distinct types in 86k sentences.

Table 2: Entity types and their respective frequencies in 1M subcorpus of the National Corpus of Polish

Entity type	**Frequency (in thousands)**
persName	20.4
persName.forename	13.2
persName.surname	13.0
orgName	11.8
placeName.settlement	8.4
placeName.country	8.1
geogName	4.7
date	4.5
persName.addName	1.0
placeName.region	0.9
time	0.6
placeName.region	0.4
placeName.district	0.3
placeName.bloc	0.1
(in total)	87.4

Figure 3 presents overlaps of named entities within 1M subcorpus of the National Corpus of Polish. Values are calculated as frequency of both labels overlaps to the frequency of vertical label, eg. *persName.forename* overlaps with *persName* whenever the first one is present, but *persName* overlaps *forename* in only 64% of cases it appeared in the training set (in this case it reflects the fact that all the *persName.forename* are nested in corresponding *persName* but only some of the *persName*s contain *forename*).

In addition to nested named entities, the mentioned dataset contains a marginal number of non-continuous name entities, such as in *gmina miejska Gdynia* 'Gdynia

Municipality' where the single entity is formed from the first and the last words, with the middle one omitted.

These were intentionally ignored. In general, the tested solutions were selected with the assumption that the final test set will share a similar distribution of entity types, overlapping and related problems.

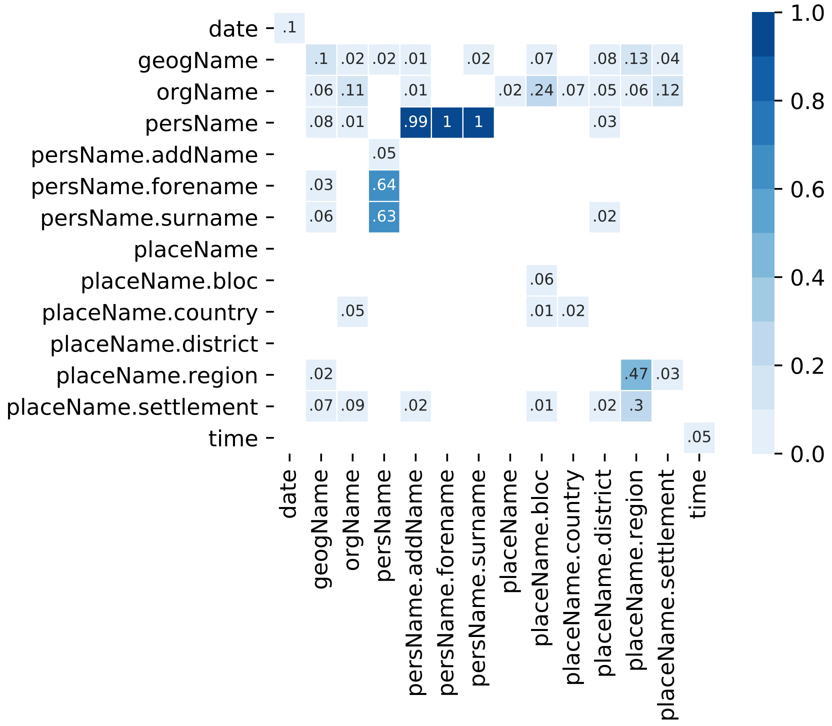


Figure 3: Overlaps of named entities within 1M subcorpus of the National Corpus of Polish. Values calculated as frequency of both labels overlaps to the frequency of vertical label

2. Towards Choosing an Optimal Solution

Subcorpora described in the previous section was divided into new train (80k sentences), dev and tests sets (both ca. 3k sentences), that were used to start an internal challenge within the local instance of an open source, git-based Gonito.net platform (Graliński et al. 2016). Span F_1 mentioned in the present section is the result of evaluation on so-created, local test set, calculated with the use of *geval* tool. After the official results were published, the submissions described in this paper were uploaded to an open instance of Gonito.net platform, where all the readers are encouraged to compete.[3]

Most of the solutions rely on training the separate models per (almost) non-overlapping entity groups, that is groups guaranteeing that individual entities within will not collide with each other. Whenever possible, groups consisted of neighboring entities in order to exploit the potential of linear CRF chain. Groups distinguished were (cf. Figure 3 for justification):

— *geogName, placeName,*

— *orgName,*

— *persName.addName, persName.forename, persName.surname,*

— *persName,*

— *placeName.bloc, placeName.region, placeName.country, placeName.district,*

— *time, date, placeName.settlement*.

This approach excludes the possibility of nesting the same type of named entity by design, ignoring that eg. half of *placeName.region* objects have a lower-level *placeName.region* inside. The problem was intentionally left for further exploiting, bearing the expected classes' popularity and limited time in mind.

2.1. Baseline: Search-Based Structured Prediction

As a baseline we decided to rely on the search-based structured prediction, an effective algorithm for reducing structured prediction problems to classification problems (Daumé III et al. 2009), implemented in the *Vowpal Wabbit* machine learning system.[4] Training was performed in 3 passes, with copying features from neighboring lines and search history length set to 6, utilizing the following features:

— token length;

— whether token contains: uppercase letter, lowercase letter, digits, punctuation, dash, colon, only digits, only uppercase letters, only lowercase letters, only punctuation;

— if token was found on the predefined list of first names, surnames, towns, communes, streets, institutions, music bands, geographical names and countries (sourced from Wikipedia, TERYT database and Rymut's dictionary (Rymut 1992));

[3]See: `https://gonito.net/challenge/poleval-2018-ner`

[4]`https://github.com/JohnLangford/vowpal_wabbit`

— character n-grams (ranging from 4 to 6) and distinguished affixes,

— rough representation of the token, eg. `Aa+` for *Adam*, `A+` for *NASA* and `9+#9+` for *20:27*;

— effect of analysis with *LanguageTool*, namely: length of lemma, affixes, lemma, morphological tags.

The system described above was able to achieve a span F_1 of 0.82 on test set ({4a1327}[5]).

2.2. LM-LSTM-CRF

The first neural approach tested was based on LM-LSTM-CRF sequence labeling tool[6], implementing the method proposed by Liu et al. (2018), where a character-level language model is trained at the same time, in addition to the sequence labeling model (note that in this method LM is not pre-trained on a large corpus, but trained only on the task data, which is one of the distinguishing features when compared to contextual string embeddings (Akbik et al. 2018)).

For the purposes of using the method, GloVe embeddings (Pennington et al. 2014) were trained on a very large, freely available[7] Common Crawl-based Web corpus of Polish (Buck et al. 2014). After basic filtering, tokenization was performed with *toki* utility (Radziszewski and Śniatowski 2011), because it is distributed along with compatible SRX rules mimicking the standard can be found in the National Corpus of Polish. After postprocessing, the corpus consisted of 27 354 330 800 tokens, 119 330 367 of which were unique. Embeddings were generated for all the tokens present in PolEval task's corpora (symmetric, cased, 300 dimensions, 30 iterations, window size of 15).

The best-performing models of this type were trained for 100 epochs, with the default settings (except higher dimension of word embeddings and disabled word embedding fine tuning), achieving a span F_1 of 0.87 on our test set, outperforming baseline by 5 percentage points ({f2c8fc}).

2.3. Contextual String Embeddings

Contextual String Embeddings were proposed by Akbik et al. (2018), who showed that the internal states of a trained character language model can be used to create

[5]This is the reference code to a repository stored at Gonito.net. The repository may be also accessed by going to `http://gonito.net/q` and entering the code there.

[6]`https://github.com/LiyuanLucasLiu/LM-LSTM-CRF`

[7]`http://data.statmt.org/ngrams/raw/`

word embeddings able to outperform the previous state-of-the-art in sequence labeling tasks. The method was implemented in Flair framework[8] we used for the purposes of training the best-performing models.

Forward and backward character-level language models were trained on 1B words corpus of Polish composed in one third of respectively subsamples from: Polish Wikipedia, PolEval's language modeling task (supposedly the National Corpus of Polish) and Polish Common Crawl. The text was tokenized using the same pipeline as in the preparation of GloVe embeddings described above. Subsamples of Wikipedia and PolEval tasks were selected randomly, whereas those sentences were selected from Common Crawl which were characterized by the highest similarity to PolEval sample, as expressed with cross-entropy (Moore and Lewis 2010).

We used exactly the same parameters, settings and assumptions as Akbik et al. (2018), achieving the final perplexity of 2.44 for forward and 2.47 for backward LM.

The final LSTM-CRF sequence labeling models were trained with one bidirectional LSTM layer and 512 hidden states on 300-dimensional GloVe embeddings (cf. the previous section), as well as embeddings from forward and backward LMs with 2048 hidden states. No progress in terms of span F_1 measured on dev set was observed after 30 epochs which distinguishes the method from LM-LSTM-CRF approach. As expected, the models outperformed previous neural solution achieving F-score of 0.88 on the internal test set ({82e4d1}). The submitted models, trained with our dev set included, performed even better, resulting in F-measure about 0.89.

PolEval nested NER task was evaluated in a different manner, combining weighted measures calculated for overlap and exact matches, giving strong premium for the former. The official, final score turned out to be 0.866, compared to 0.851 for the second best and 0.810 for the third.

Code and models accompanying the paper, which can be used to reproduce the results are publicly available at: `https://github.com/applicaai/poleval-2018`.

3. Discussion

The described solutions and settings were not the only ones tested, eg. 300-dimensional fastText embeddings provided by Grave et al. (2018) were considered, but we found the GloVe ones better suiting the task. Moreover, the Layered-LSTM-CRF[9] was examined, but the results achieved were disappointing, when following the detection order rule proposed by authors, even when contextual string embeddings

[8]`https://github.com/zalandoresearch/flair`

[9]`https://github.com/meizhiju/layered-bilstm-crf`

were used. It may be due to the specific character of the attempted dataset, where given two entity classes it is not known which one will appear in inside and which in outside layers. Since this approach was not sufficiently tested due to the lack of time, we are not reporting it in details.

Furthermore, the layered-LSTM inspired method was tested for second-order LSTM-CRF models whenever it could be beneficial, especially for *persName* tag, that should appear outside every, lower-lever classes group (*persName.forename*, *persName.surname*, *persName.addName*). Including information about those had no impact on the overall performance despite substantially affecting learning speed.

After the predicted answers were sent, LM training continued until no progress was observed, achieving the final perplexity of 2.41 for the forward and 2.46 for the backward model. This encouraged us to test how it could affect the overall results. However, no improvement of sequence labeling model was observed, and the only change was a steeper learning curve (the same accuracy was achieved after fewer epochs).

References

Akbik A., Blythe D. and Vollgraf R. (2018). *Contextual String Embeddings for Sequence Labeling*. [in:] *COLING 2018, 27th International Conference on Computational Linguistics*, pp. 1638–1649.

Bojanowski P., Grave E., Joulin A. and Mikolov T. (2017). *Enriching Word Vectors with Subword Information*. „Transactions of the Association for Computational Linguistics”, 5, pp. 135–146.

Buck C., Heafield K. and van Ooyen B. (2014). *N-gram Counts and Language Models from the Common Crawl*. [in:] *Proceedings of the Language Resources and Evaluation Conference*, Reykjavik, Iceland.

Celikyilmaz A., Hakkani-Tür D., Pasupat P. and Sarikaya R. (2015). *Enriching Word Embeddings Using Knowledge Graph for Semantic Tagging in Conversational Dialog Systems*. [in:] *AAAI Spring Symposium Series*. Association for the Advancement of Artificial Intelligence.

Daumé III H., Langford J. and Marcu D. (2009). *Search-based Structured Prediction*. „Machine Learning Journal”.

Graliński F., Jaworski R., Borchmann Ł. and Wierzchoń P. (2016). *Gonito.net – Open Platform for Research Competition, Cooperation and Reproducibility*. [in:] Branco A., Calzolari N. and Choukri K. (eds.), *Proceedings of the 4REAL Workshop: Workshop on Research Results Reproducibility and Resources Citation in Science and Technology of Language*, pp. 13–20, Portoroz, Slovenia.

Grave E., Bojanowski P., Gupta P., Joulin A. and Mikolov T. (2018). *Learning Word Vectors for 157 Languages*. [in:] Calzolari N., Choukri K., Cieri C., Declerck T., Goggi S., Hasida K., Isahara H., Maegaard B., Mariani J., Mazo H., Moreno A., Odijk J., Piperidis S. and Tokunaga T. (eds.), *Proceedings of the 11th International Conference on Language Resources and Evaluation (LREC 2018)*, pp. 3483–3487. European Language Resources Association.

Harris Z. S. (1954). *Distributional Structure*. „WORD”, 10(2-3), pp. 146–162.

Huang Z., Xu W. and Yu K. (2015). *Bidirectional LSTM-CRF Models for Sequence Tagging*. „CoRR”, abs/1508.01991.

Ju M., Miwa M. and Ananiadou S. (2018). *A Neural Layered Model for Nested Named Entity Recognition*. [in:] *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1446–1459. Association for Computational Linguistics.

Katiyar A. and Cardie C. (2018). *Nested Named Entity Recognition Revisited*. [in:] *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 861–871. Association for Computational Linguistics.

Lample G., Ballesteros M., Subramanian S., Kawakami K. and Dyer C. (2016). *Neural Architectures for Named Entity Recognition*. „CoRR”, abs/1603.01360.

Liu L., Shang J., Xu F., Ren X., Gui H., Peng J. and Han J. (2018). *Empower Sequence Labeling with Task-aware Neural Language Model*. [in:] *AAAI*.

Moore R. C. and Lewis W. (2010). *Intelligent Selection of Language Model Training Data*. [in:] *Proceedings of the ACL 2010 Conference Short Papers*, ACLShort ’10, pp. 220–224. Association for Computational Linguistics.

Nadeau D. and Sekine S. (2007). *A Survey of Named Entity Recognition and Classification*. „Linguisticae Investigationes”, 30(1), pp. 3–26. Publisher: John Benjamins Publishing Company.

Pandey P., Pudi V. and Shrivastava M. (2017). *Injecting Word Embeddings with Another Language’s Resource: An Application of Bilingual Embeddings*. [in:] *Proceedings of the 8th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pp. 116–121. Asian Federation of Natural Language Processing.

Pennington J., Socher R. and Manning C. D. (2014). *Glove: Global Vectors for Word Representation*. [in:] *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543.

Peters M. E., Neumann M., Iyyer M., Gardner M., Clark C., Lee K. and Zettlemoyer L. (2018). *Deep Contextualized Word Representations*. „CoRR", abs/1802.05365.

Przepiórkowski A., Bańko M., Górski R. and Lewandowska-Tomaszczyk B. (2012). *Narodowy Korpus Języka Polskiego*. Wydawnictwo Naukowe PWN.

Radziszewski A. and Śniatowski T. (2011). *Maca — a Configurable Tool to Integrate Polish Morphological Data*. [in:] *Proceedings of the 2nd International Workshop on Free/Open-Source Rule-Based Machine Translation (FreeRBMT11)*.

Rymut K. (1992). *Słownik nazwisk współcześnie w Polsce używanych*. Instytut Języka Polskiego Polskiej Akademii Nauk.

Young T., Hazarika D., Poria S. and Cambria E. (2017). *Recent Trends in Deep Learning Based Natural Language Processing*. „CoRR", abs/1708.02709.

Named Entity Recognition for Polish Using Contextual String Embeddings

Adam Kaczmarek, Paweł Rychlikowski, Michał Zapotoczny
(Computational Intelligence Research Group, Institute of Computer Science, University of Wrocław)

We provided a system for PolEval 2018 Named Entity Recognition task for Polish. This task was a nested NER task with named entities either overlapping themselves or containing sub-structures of the same named entity family. Problem was represented as a sequence labelling task with flat tag structure converted to IOBES scheme. All the tokens belonging to more than one named entity at the same time were considered as labelled with composite tags consisting of all the constituent single named entity tags annotated for these tokens. In our solution we used two types of word embeddings. First, from "standard" word embeddings we chose *fastText* (Joulin et al. 2016) over *glove* (Pennington et al. 2014) due to their better handling of subword information. We used pretrained *fastText* word embeddings for Polish provided in: `https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md`. Second type of embeddings used in our solutions are character-based contextual string embeddings. We decided to use the embeddings implemented in the Flair (Akbik et al. 2018) framework constructed using forward and backward language models. The implementation of the model presented in competition was based on the state-of-the art sequence labelling approaches using the Flair framework. As the final model we used a bidirectional LSTM neural network with 512 hidden units and a CRF layer on the top. For input we provided *fastText* word embeddings of size 300 and both forward and backward character language models with embeddings of size 1024, trained on the NKJP corpus, for contextual string embeddings. Model was trained using SGD optimizer with simple learning rate annealing for learning rate adjusted in range between 0.1 and 0.001. For internal evaluation purposes system was trained and validated on the training dataset from the National Corpus of Polish split in 80%/20% proportion, achieving F1 score of 86% during training. The final system was trained on whole NKJP corpus achieving in the final evaluation results of: 62.3% in exact metric and 74.3% in overlap metric resulting in final score of 71.9%.

References

Akbik A., Blythe D. and Vollgraf R. (2018). *Contextual String Embeddings for Sequence Labeling*. [in:] *Proceedings of the 27th International Conference on Computational Linguistics (COLING 2018)*, pp. 1638–1649.

Joulin A., Grave E., Bojanowski P. and Mikolov T. (2016). *Bag of Tricks for Efficient Text Classification*. „arXiv:1607.01759”.

Pennington J., Socher R. and Manning C. D. (2014). *Glove: Global Vectors for Word Representation*. [in:] *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543. Association for Computational Linguistics.

Recognition of Named Entities for Polish — Comparison of Deep Learning and Conditional Random Fields Approaches

Michał Marcińczuk, Jan Kocoń, Michał Gawor (Wrocław University of Science and Technology)

Abstract

In the paper we present two systems for named entities recognition for Polish submitted to PolEval 2018 competition (Task 2). The first one, called Liner2, utilizes Conditional Random Fields with a rich set of features. The other one, called PolDeepNer, is an ensemble of three neural networks using a Bi-directional Long Short-Term Memory (Bi-LSTM) or a Bi-directional Gated Recurrent Units (Bi-GRU) with a CRF layer. Each of the networks was trained on different word embeddings. These approaches are the state-of-the-art techniques used in many tasks from Information Extraction field. The presented models got the second (PolDeepNer) and the third (Liner2) place in he PolEval competition. We also present a comparison of these two models in terms of their accuracy and algorithmic efficiency. The evaluation showed that the model based on deep learning outperformed conditional random fields, however with the cost of higher requirements for memory usage, model size and time processing. Both systems are publicly available under the GPL license.

Keywords

named entity recognition, information extraction, conditional random fields, deep learning, natural language processing, LSTM, GRU

1. Introduction and Related Works

Named entity recognition (henceforth, NER) is one of the tasks from the field of natural language processing and it relies on finding in text mentions which refer to some named entities, for instance persons, locations or organizations. The set of semantic categories of entities and types of mentions depends on the specific application. For instance, one of the best known datasets for English called CoNLL 2003 (Sang et al. 2003) defined only four types of named entity: LOC (location), ORG (organization), PER (person) and MISC (miscellaneous). In turn, an another dataset for English called OntoNotes 5.0 (Weischedel et al. 2012) defines 11 types of named entities. For Polish there are two major datasets: The National Corpus of Polish (NKJP; Przepiórkowski et al. 2012) and Polish Corpus of Wrocław University of Technology (KPWr; Broda et al. 2012).

In the recent years we observe the rise of deep learning applications in the natural language processing field (Young et al. 2018), including POS tagging, NER, parsing, semantic role labelling, sentiment classification and many other tasks. In many cases the application of deep learning brings a significant improvement in accuracy comparing to the other methods. For English, the state-of-the-art systems for NER are based on different types of neural nets, including BiLSTM-CNN-CRF (Ma and Hovy 2016), LM-LSTM-CRF (Liu et al. 2017), GRU-CRF (Yang et al. 2017) and GRU-LM-CRF (Peters et al. 2017). According to our best knowledge there are no systems for NER for Polish based on deep learning. The state-of-the-art systems for Polish are based on conditional random fields — Liner2 (Marcińczuk et al. 2017) trained on the KPWr corpus and NERF (Savary and Waszczuk 2012) trained on the NKJP corpus.

In the article we present two systems for named entity recognition for Polish which were submitted to the PolEval 2018 competition — Liner2[1] based on conditional random fields (see Section 3) and PolDeepNer[2] based on deep learning (see Section 4). In Section 2 we describe the official PolEval 2018 training dataset and our approach to simplify the named entity annotation model. In the last section (Section 5) we compare the two different approaches in terms of their accuracy, processing time, model size and memory usage.

2. Dataset

The National Corpus of Polish (NKJP) was the official training dataset used to train models within PolEval 2018 Task 2. The corpus contains 87 300 named entities

[1]Liner2 is available at `https://github.com/CLARIN-PL/Liner2`

[2]PolDeepNer is available at `https://github.com/CLARIN-PL/PolDeepNer`

annotated with six main types and eight subtypes. Figure 1 presents a complete hierarchy of the NE types.

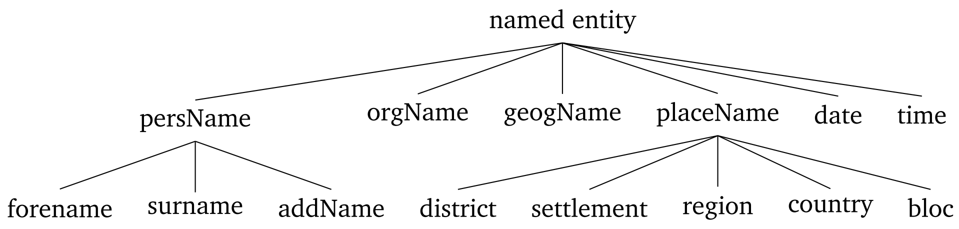


Figure 1: A complete hierarchy of the named entity types

Comparing to other corpora annotated with named entities the NKJP corpus contains nested annotations of the same type and disjoint annotations. The two assumptions make the task more difficult. In the following part of this section we discuss how we addressed this difficulty.

Named entity recognition is solved as a sequence labelling task. The text is divided into a sequence of tokens and each token has a label which encodes the information if the token is part of an annotation. In case of multi-token annotations we use the IOB (or BIO) encoding, where the first token of annotation of type NAME is labelled as B-NAME and the following tokens of the same annotation are labelled as I-NAME. Tokens which are not part of any annotation are labelled as O. In case of nested annotations, the token label is created by joining labels for every single annotation, i.e. B-TYPE1#B-TYPE2. For instance the sentence:

> (...) twierdzi prof. Jacek Ruszkowski, dyrektor Centrum Zdrowia Publicznego Wyższej Szkoły Przedsiębiorczości i Zarządzania im. Leona Koźmińskiego w Warszawie

contains the following structure of named entities:

1. *persName*: Jacek Ruszkowski,
   (a) *persName-forename*: Jacek,
   (b) *persName-surname*: Ruszkowski,
2. *orgName*: Centrum Zdrowia Publicznego Wyższej Szkoły Przedsiębiorczości i Zarządzania im. Leona Koźmińskiego w Warszawie,
   (a) *orgName*: Wyższej Szkoły Przedsiębiorczości i Zarządzania im. Leona Koźmińskiego w Warszawie,

   i. *persName*: Leona Koźmińskiego,

      A. *persName-forename*: Leona,

      B. *persName-surname*: Koźmińskiego,

   ii. *placeName-settlement*: Warszawie.

Figure 2 presents the sample sentence as a sequence of tokens and labels. Disjoint and nested annotations causes the following problems:

1. Nested annotations lead to a huge number of unique labels — there are exactly 262 unique labels in the NKJP corpus. 100 of them appeared only once or twice in the whole corpus. Infrequent annotations are difficult to be learned from the training data. In case of CRF such a number of labels increases the training time dramatically.

2. Nested or overlapping annotations of the same type are problematic because they may be ambiguous when decoding from IOB format. For instance, the sequence:

   ```
   AAA B-nam
   BBB I-nam#B-nam
   CCC I-nam
   ```

   might be interpreted as (1) "AAA BBB CCC" and "BBB" or (2) "AAA BBB" and "BBB CCC".

3. Disjoint annotations also might be ambiguous. Consider the following example:

   ```
   AAA B-nam
   BBB B-nam
   CCC O
   DDD I-nam
   ```

   "DDD" might be a part of the first or the second annotation.

Taking into account the above issues, we have decided to simplify the annotation model by: (1) splitting disjoint annotations into sets of continuous parts, (2) ignore nested annotations of the same type and (3) ignore infrequent nested annotations. We used Liner2 to convert the original NKJP corpus in TEI format to a simplified

```
(...)
twierdzi            twierdzić           fin:sg:ter:imperf                 O
prof                profesor            brev:pun                          O
.                   .                   interp:                           O
Jacek               Jacek               subst:sg:nom:m1                   B-persName#B-persName-forename
Ruszkowski          Ruszkowski          subst:sg:voc:m1                   I-persName#B-persName-surname
,                   ,                   interp:                           O
dyrektor            dyrektor            subst:sg:nom:m1                   O
Centrum             centrum             subst:sg:gen:n                    B-orgName
Zdrowia             zdrowie             subst:pl:voc:n                    I-orgName
Publicznego         publiczny           adj:sg:gen:n:pos                  I-orgName
Wyższej             wysoki              adj:sg:gen:f:com                  I-orgName#B-orgName
Szkoły              szkoła              subst:pl:nom:f                    I-orgName#I-orgName
Przedsiębiorczości  przedsiębiorczość   subst:pl:nom:f                    I-orgName#I-orgName
i                   i                   conj:                             I-orgName#I-orgName
Zarządzania         zarządzanie         subst:sg:gen:n                    I-orgName#I-orgName
im                  on                  ppron3:pl:dat:m1:ter:akc:npraep   I-orgName#I-orgName
.                   .                   interp:                           I-orgName#I-orgName
Leona               Leona               subst:sg:nom:f                    I-orgName#I-orgName#B-persName
                                                                                   #B-persName-forename
Koźmińskiego        koźmiński           adj:sg:gen:m2:pos                 I-orgName#I-orgName#I-persName
                                                                                   #B-persName-surname
w                   w                   prep:acc:nwok                     I-orgName#I-orgName
Warszawie           warszawa            subst:sg:loc:f                    I-orgName#I-orgName
                                                                                   #B-placeName-settlement
.                   .                   interp:                           O
```

Figure 2: A sample sentence from NKJP converted to a sequence of tokens and annotations as IOB labels

version in IOB format[3]. The simplified version of the corpus contains 96.76% of the original set of annotations and after converting it to the IOB format there were only 40 unique labels (six times less than in the original version).

During development we split the NKJP corpus into two parts in the ratio of 4:1 – 80% of the documents were used as a training part and the remaining 20% were used for testing. This split was used to obtain possibly the best configuration. The final models were trained on the whole NKJP corpus.

The evaluation corpus contained 1828 documents annotated according to NKJP guidelines. In contrast to NKJP, the evaluation corpus contained documents in plain text. In NKJP the documents were already tokenised. In order to have the tokenisation compatible with NKJP we applied the WCRFT tagger (Radziszewski 2013).

3. Liner2 — Conditional Random Fields

Liner2[4] (Marcińczuk et al. 2013) is an open-source generic framework for sequence text labeling and annotation manipulation, including recognition of named entities. It has been already used to train a state-of-the-art NER models following the KPWr guidelines (Marcińczuk et al. 2016), including NE boundary detection, coarse-grained categorization (9 types) and fine-grained categorization (82 types) (Marcińczuk et al. 2017). It was used also to other Information Extraction tasks like the recognition of temporal expressions (Kocoń and Marcińczuk 2017) and events (Kocoń and Marcińczuk 2016). Liner2 uses Conditional Random Fields as a statistical model and a rich space of features of different types, including: orthographic, structural, morphological, lexicon-based, wordnet-based and compound features. Liner2 exploits language and domain knowledge in a form of external structured resources, including:

— NELexicon2[5] — a dictionary of more than 2.3 million proper names (base and inflected forms) with fine-grained categorization (107 categories),

— PNET (Polish Named Entity Triggers[6]) — an electronic lexicon containing partly inflected external or internal evidences, or trigger words, for Polish named entities,

[3]The simplified version of the NKJP corpus can be found under this link: `https://github.com/CLARIN-PL/PolDeepNer/blob/master/poldeepner/data/nkjp-nested-simplified-v2.iob.7z`

[4]`https://github.com/CLARIN-PL/Liner2`

[5]`https://clarin-pl.eu/dspace/handle/11321/247`

[6]`http://zil.ipipan.waw.pl/PNET`

— plWordNet (Maziarz et al. 2016) — Polish wordnet is used to exploit synonyms and hypernyms,

— complex features — features constructed on the basis of atomic features. They are used to model relationships between combinations of input features and output labels (Marcińczuk 2015),

— morphological features — are motivated by the NER grammars which utilize morphological information (Piskorski 2004).

We trained the Liner2 model on the simplified version of NKJP corpus described in Section 2 using the configuration from the coarse-grained KPWr model (Marcińczuk et al. 2017). Liner2 NKJP model got the 3rd place in the competition with the final score of 0.810 (see Table 1). The CRF-based model was outperformed by two approaches based on deep learning.

Table 1: PolEval 2018 Task 2 official evaluation

System	**Exact**	**Overlap**	**Final**
Per group LSTM-CRF with Contextual String Embeddings	0.826	0.877	0.866
PolDeepNer	0.822	0.859	0.851
Liner2	0.778	0.818	0.810
OPI_Z3	0.749	0.805	0.793
joint	0.748	0.789	0.780
disjoint	0.747	0.788	0.779
via_ner	0.692	0.773	0.756
kner_sep	0.700	0.742	0.733
Poleval2k18	0.623	0.743	0.719
KNER	0.681	0.719	0.711
simple_ner	0.569	0.653	0.636

4. PolDeepNer — Ensemble of Deep Learning

We introduced two neural architectures for the recognition of named entities. The first one is based on bidirectional LSTMs with sequential conditional random layer above it, similarly as it was presented by Lample et al. (2016). The second one utilizes bidirectional GRU layer (Cho et al. 2014), also together with CRF layer. Both LSTMs and GRUs are units of the recurrent neural network (RNN), where in case of bidirectional RNN, context of the word is captured through past and future words

(Sahu and Anand 2016). These models rely on three sources of information about words which are three different unsupervised distributional word representations learned from unannotated corpora. The detailed description of the network structure is presented in Section 4.1.

4.1. Deep Neural Network Structure

The input text is tokenised using WCRFT tagger. In the next step each tokenised sentence goes as the input for each of the given three models:

— MODEL1 – `cc.pl.300.bin+BiGRU-CRF`

— MODEL2 – `kgr10-plain-sg-300-mC50.bin+BiGRU-CRF`

— MODEL3 – `kgr10_orths.vec.bin+BiLSTM-CRF`

The major difference between models is the usage of different distributional word representations. All word embedding models were built using fastText (Bojanowski et al. 2017) – a library for efficient learning of word representations and sentence classification. The first word embedding model for Polish (within MODEL1) was obtained from the main fastText site[7] (Grave et al. 2018). It was trained on Polish part of Common Crawl[8] and Wikipedia[9]. The dimension of word vectors is 300. The second word embedding model (within MODEL2) was built using KGR10 corpus (only using words occurring more than 50 times in corpus), which contains more than 4 billion words from Polish part of the Internet. The dimension of word vectors in this model is also 300 and the build method is *skipgram*. The last model is also *skipgram* model build using KGR10, but with dimension *100* and minimal frequency of word in a corpus is *5*. This model can be downloaded from CLARIN DSpace repository[10].

The whole schema of processing is presented in Figure 3. At the level of processing of tokenised sentence each input word is mapped as a vector (using all the presented word embedding models) and these vectors are inputs for the corresponding deep neural networks. Within each deep neural network model there are 5 layers: input layer, dropout layer, bidirectional LSTM (or GRU), dense layer and CRF layer. Figure 4 presents this schema.

[7]`https://s3-us-west-1.amazonaws.com/fasttext-vectors/word-vectors-v2/cc.pl.300.bin.gz`

[8]`http://commoncrawl.org/`

[9]`https://www.wikipedia.org/`

[10]`https://clarin-pl.eu/dspace/handle/11321/600`

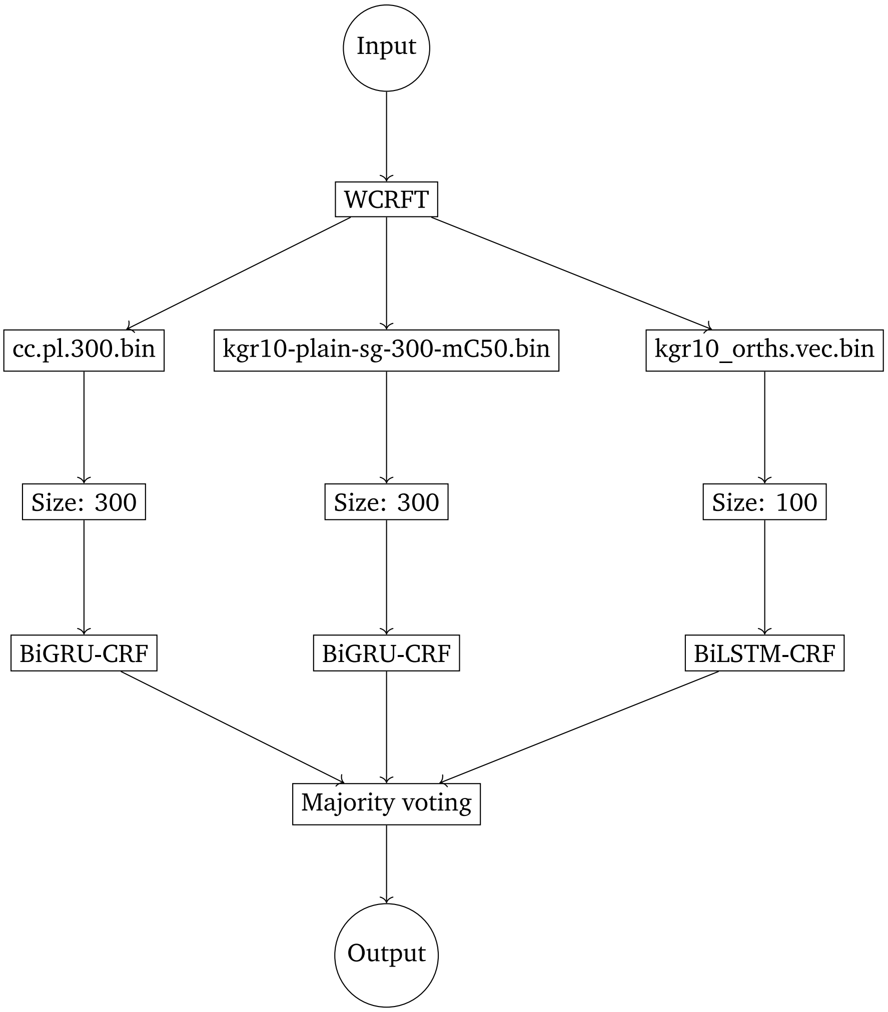


Figure 3: Processing pipeline of the given input (plain text)

Outputs from each deep neural model are combined using majority voting method and the single output (a vector of labels attached to each token from the input sequence) is returned.

4.2. Handling of Unknown Words

Popular continuous word representations, such as Word2Vec (Goldberg and Levy 2014) or GloVe (Pennington et al. 2014), produce vectors only for the known words. FastText approach, instead of assigning a distinct vector to each word, proposes a solution based on the *skipgram* model. Each word is represented as a bag-of-character

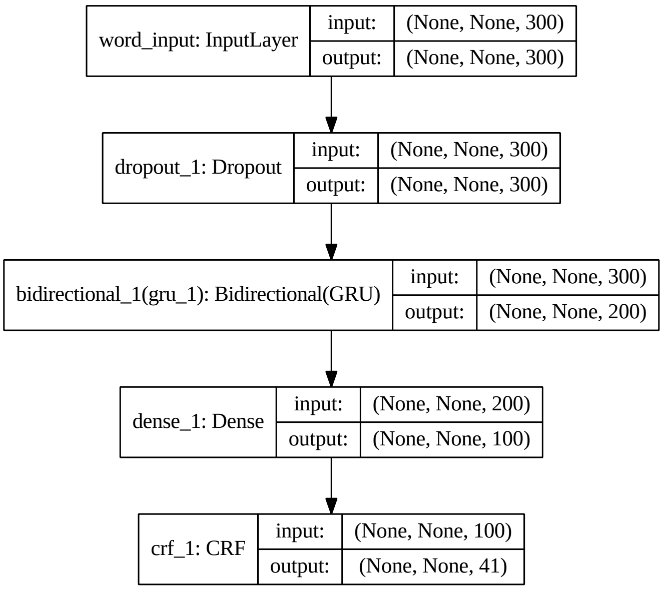


Figure 4: Layers composing a single deep neural network model

n-grams and a vector representation is attached to each n-gram. Words are represented as a sum of these n-gram representations. This approach allows to compute word representations also for words that did not appear in the training data (Bojanowski et al. 2017).

4.3. Evaluation

In the official evaluation PolDeepNer got the 2nd place with the final score for of 0.851 (see Table 1). In order to publish PolDeepNer source code and models we had to retrain them. The re-trained model obtained slightly different final score of 0.853 (see Table 2). However, the difference is not statistically significant.

5. Algorithmic Efficiency

In the recent years the state-of-the-art systems for NER are based on deep learning as they outperform the other methods including conditional random fields. The highest accuracy scored by a CRF-based system on the CoNLL-2003 dataset was 90.90% of F_1 measure (Passos et al. 2014). In turn the highest accuracy obtained by a deep learning-based system was scored by GRU-LM-CRF and it was 91.93% (Peters et al.

2017). The same trend can be observed in PolEval results (see Table 1). The two best tools are based on deep learning and they outperformed by 4–5 pp. the CRF-based system which got the third place. The PolEval results showed that also for NER in Polish the deep learning outperformed conditional random fields.

In our research we observed that the relatively small improvement achieved by PolDeepNer was bought with a significant increase in time processing, model size and memory usage. We decided to explore the topic and we compared *Liner2*, *PolDeepNer* and a single *BiLSTM-CRF* (one of the nets used in PolDeepNer) using the four following measures:

— **accuracy** according to PolEval evaluation procedure,

— **processing time** — how long the evaluation dataset is being processed (in seconds),

— **model size** — total size of the model and external resources required to process the documents. Liner2 requires tokenised text on the input, so the total size of the model includes also morphological dictionary Morfeusz (Woliński 2006) and WCRFT tagger model (Radziszewski 2013).

— **memory usage** — total amount of memory required by the operating system (Ubuntu 16.04) to run the tool and process the documents.

We included the single BiLSTM-CRF to see the difference between Liner2 and a simple neural net, and the difference between the single NN and the ensemble of NNs — PolDeepNer. Our findings are presented in Table 2 and Figure 5.

Table 2: Comparison of models in terms of their score, processing time, model size and memory usage

		Liner2	**BiLSTM-CRF**	**PolDeepNer**
Score	[%]	81.0	83.9	85.3
Processing speed	[s]	173	164	535
Model size	[GB]	0.43	7.20	16.3
Memory usage	[GB]	1	7.3	17.2

In Figure 5 we can see that the differences in scores between the systems are relatively small comparing to the other measures. The single BiLSTM-CRF obtained higher accuracy by 2.9 pp. than Liner2 with a slightly shorter processing time — 164 seconds comparing to 174 seconds for Liner2. However, in terms of model size and memory usage there is a huge difference — the size of BiLSTM-CRF model is 16 times larger than Liner2 model and it requires 7 times more memory. The increase is even larger for PolDeepNer. With an improvement of accuracy by 4.3 pp. the model size is near

38 times larger and the memory usage is 17 times higher. The increase in model size and memory usage is caused by word embedding model — the core element for text processing with neural nets. In case of PolDeepNer there are three different word embedding models.

Our findings show that in case of unlimited resources the best approach is the one for which the accuracy is maximized, i.e. PolDeeper. In other cases when the resources are limited by hardware capabilities the decision should be taken with care and should be the resultant of mentioned factors. In other words, despite Liner2 achieved lower accuracy than PolDeepNer, in some cases it might be better or the only acceptable choice.

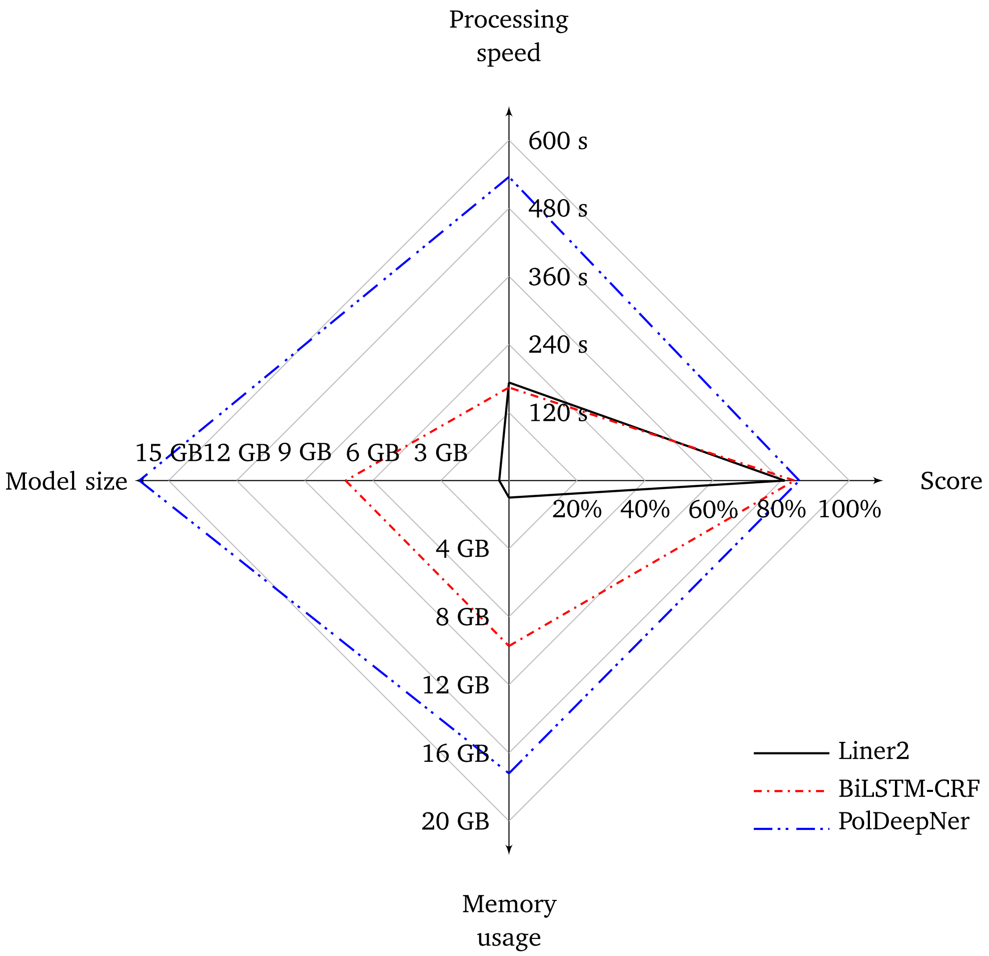


Figure 5: Comparison of models in terms of their score, processing time, model size and memory usage

6. Summary

In the paper we presented two system for named entity recognition for Polish submitted to PolEval 2018 competition. The first one called PolDeepNer which is based on deep learning got 2nd place with the final score of 0.851. The other one called Liner2 which is based on conditional random fields got 3rd place with the score of 0.811. Both system are publicly available under GPL license. In spite of the approach based on deep learning achieved higher score than the one based on conditional random fields we argued that due to the lower algorithmic efficiency in terms of processing time, memory usage and model size, in some cases the approach based on conditional random fields might be a better or the only acceptable choice.

Acknowledgments

Work financed as part of the investment in the CLARIN-PL research infrastructure funded by the Polish Ministry of Science and Higher Education.

References

Bojanowski P., Grave E., Joulin A. and Mikolov T. (2017). *Enriching Word Vectors with Subword Information*. „Transactions of the Association for Computational Linguistics", 5, pp. 135–146.

Broda B., Marcińczuk M., Maziarz M., Radziszewski A. and Wardyński A. (2012). *KPWr: Towards a Free Corpus of Polish*. [in:] *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC 2012)*, pp. 3218—-3222.

Cho K., van Merrienboer B., Bahdanau D. and Bengio Y. (2014). *On the Properties of Neural Machine Translation: Encoder-decoder Approaches*. [in:] *Proceedings of the 8th Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8), 2014*.

Goldberg Y. and Levy O. (2014). *word2vec Explained: Deriving Mikolov et al.'s Negative-Sampling Word-Embedding Method*. „arXiv:1402.3722".

Grave E., Bojanowski P., Gupta P., Joulin A. and Mikolov T. (2018). *Learning Word Vectors for 157 Languages*. [in:] Calzolari N., Choukri K., Cieri C., Declerck T., Goggi S., Hasida K., Isahara H., Maegaard B., Mariani J., Mazo H., Moreno A., Odijk J., Piperidis S. and Tokunaga T. (eds.), *Proceedings of the Eleventh International Conference on*

Language Resources and Evaluation (LREC 2018), pp. 3483–3487. European Language Resources Association.

Kocoń J. and Marcińczuk M. (2016). *Generating of Events Dictionaries from Polish WordNet for the Recognition of Events in Polish Documents*, t. 9924 serii *Lecture Notes in Computer Science*, pp. 12–19. Springer International Publishing, Cham.

Kocoń J. and Marcińczuk M. (2017). *Supervised Approach to Recognise Polish Temporal Expressions and Rule-based Interpretation of Timexes*. „Natural Language Engineering”, 23(3), pp. 385–418.

Lample G., Ballesteros M., Subramanian S., Kawakami K. and Dyer C. (2016). *Neural Architectures for Named Entity Recognition*. „arXiv:1603.01360”.

Liu L., Shang J., Xu F., Ren X., Gui H., Peng J. and Han J. (2017). *Empower Sequence Labeling with Task-aware Neural Language Model*. „arXiv:1709.04109”.

Ma X. and Hovy E. (2016). *End-to-end Sequence Labeling via Bi-directional LSTM-CNNS-CRF*. „arXiv:1603.01354”.

Marcińczuk M. (2015). *Automatic Construction of Complex Features in Conditional Random Fields for Named Entities Recognition*. [in:] *Proceedings of Recent Advances in Natural Language Processing (RANLP 2015)*, pp. 413–419.

Marcińczuk M., Kocoń J. and Janicki M. (2013). *Liner2 – a Customizable Framework for Proper Names Recognition for Polish*. [in:] Bembenik R., Skonieczny L., Rybiński H., Kryszkiewicz M. and Niezgódka M. (eds.), *Intelligent Tools for Building a Scientific Information Platform*, pp. 231–253.

Marcińczuk M., Oleksy M. and Dziob A. (2016). *KPWr Annotation Guidelines — Named Entities*. CLARIN-PL digital repository.

Marcińczuk M., Kocoń J. and Oleksy M. (2017). *Liner2 — a Generic Framework for Named Entity Recognition*. [in:] Erjavec T., Piskorski J., Pivovarova L., Snajder J., Steinberger J. and Yangarber R. (eds.), *Proceedings of the 6th Workshop on Balto-Slavic Natural Language Processing (BSNLP@EACL 2017)*, pp. 86–91. Association for Computational Linguistics.

Maziarz M., Piasecki M., Rudnicka E., Szpakowicz S. and Kędzia P. (2016). *PlWordNet 3.0 – a Comprehensive Lexical-Semantic Resource*. [in:] Calzolari N., Matsumoto Y. and Prasad R. (eds.), *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers*, pp. 2259–2268. Association for Computational Linguistics.

Passos A., Kumar V. and McCallum A. (2014). *Lexicon Infused Phrase Embeddings for Named Entity Resolution*. „arXiv:1404.5367".

Pennington J., Socher R. and Manning C. (2014). *Glove: Global Vectors for Word Representation*. [in:] *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543.

Peters M. E., Ammar W., Bhagavatula C. and Power R. (2017). *Semi-supervised Sequence Tagging with Bidirectional Language Models*. „arXiv:1705.00108".

Piskorski J. (2004). *Extraction of Polish Named Entities*. [in:] *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC 2004)*, pp. 313–316, European Language Resource Association,

Przepiórkowski A., Bańko M., Górski R. L. and Lewandowska-Tomaszczyk B. (2012). *Narodowy Korpus Języka Polskiego*. Wydawnictwo Naukowe PWN.

Radziszewski A. (2013). *A Tiered CRF Tagger for Polish*. [in:] Bembenik R., Skonieczny B., Rybiński H., Kryszkiewicz M. and Niezgódka M. (eds.), *Intelligent Tools for Building a Scientific Information Platform: Advanced Architectures and Solutions*. Springer Verlag.

Sahu S. and Anand A. (2016). *Recurrent Neural Network Models for Disease Name Recognition Using Domain Invariant Features*. [in:] *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2216–2225. Association for Computational Linguistics.

Sang T. K., F. E. and De Meulder F. (2003). *Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition*. [in:] *Proceedings of the 7th Conference on Natural Language Learning at HLT-NAACL 2003*, vol. 4, pp. 142–147. Association for Computational Linguistics.

Savary A. and Waszczuk J. (2012). *Narzędzia do anotacji jednostek nazewniczych*. [in:] Przepiórkowski A., Bańko M., Górski R. L. and Lewandowska-Tomaszczyk B. (eds.), *Narodowy Korpus Języka Polskiego*, pp. 225–252. Wydawnictwo Naukowe PWN, Warsaw.

Weischedel R., Pradhan S., Ramshaw L., Kaufman J., Franchini M., El-Bachouti M., Xue N., Palmer M., Hwang J. D., Bonial C., Choi J., Mansouri A., Foster M., aati Hawwary A., Marcus M., Taylor A., Greenberg C., Hovy E., Belvin R. and Houston A. (2012). *OntoNotes Release 5.0 with OntoNotes DB Tool v0.999 beta*.

Woliński M. (2006). *Morfeusz — a Practical Tool for the Morphological Analysis of Polish*. [in:] Kłopotek M. A., Wierzchoń S. T. and Trojanowski K. (eds.), *Intelligent Information Processing and Web Mining*, pp. 511–520, Berlin, Heidelberg. Springer Berlin Heidelberg.

Yang Z., Salakhutdinov R. and Cohen W. W. (2017). *Transfer Learning for Sequence Tagging with Hierarchical Recurrent Networks*. „arXiv:1703.06345".

Young T., Hazarika D., Poria S. and Cambria E. (2018). *Recent Trends in Deep Learning Based Natural Language Processing [review article]*. „IEEE Computational Intelligence Magazine", 13(3), pp. 55–75.

A Bidirectional LSTM-CRF Network with Subword Representations, Character Convolutions and Morphosyntactic Features for Named Entity Recognition in Polish

Mateusz Piotrowski (VoiceLab), **Wojciech Janowski** (VoiceLab), **Piotr Pęzik** (VoiceLab, University of Łódź)

Abstract

This paper describes a neural network architecture developed by the VoiceLab team for the task of Named Entity Recognition in the PolEval 2018 competition. As input to the network we used externally trained (sub)word vectors, convolutions of character embeddings and selected morphosyntactic features of word tokens. The network architecture was implemented in the Tensorflow framework and it featured two bi-LSTM layers and a Conditional Random Fields output layer.

Keywords

Bidirectional LSTMs, character embeddings, Named Entity Recognition, Polish

1. Introduction

Named Entity Recognition (NER) is a well-established task of Natural Language Processing. Annotated mentions of named entites such as people, organizations or places are used in information extraction tasks, for instance to retrieve relations between named persons or in information retrieval systems to enrich search indexes with basic semantic information. Robust NER is also important in the area of natural language understanding, where mentions of generic and domain-specific entities such as dates, places and products need to be recognized in order to accurately support human-machine dialog interactions. NER is also relevant to the task of probabilistic language modeling in Automatic Speech Recognition, which is the core area of expertise of the VoiceLab team. For example, models of morphologically rich languages as utilized in ASR could potentially be more generalizable if rare proper names are correctly labeled as named entity types rather than modeled as infrequent word tokens. In order to be useful, however, NER systems need to be sufficiently precise and comprehensive for each of these tasks. While considerable improvements in the performance of "deep neural network" architectures over more traditional approaches to NER have been reported for English (Chiu and Nichols 2015) and other languages, published NER systems for Polish (Marcińczuk et al. 2011) have mostly relied on previous generation methods such as Conditional Random Fields (Lafferty et al. 2001). The PolEval 2018 competition has created an opportunity to implement and evaluate state-of-the art methods for Polish as well.

2. NER in PolEval 2018

Technically, the task of NER is usually defined as the assignment of named entity labels to sequences of naturally-occurring word tokens. Constituents of named entities are conventionally marked as `B-TYPE` or `I-TYPE`, where `B` and `I` stand for *beginning* and *inside* respectively, and `TYPE` indicates the type of entity such as `PLACE` or `PERSON`. Considering that word tokens which are not part of a named entity of interest are marked with the `O` label (i.e. *outside*), the term `BIO` Notation is sometimes used to describe the format of named entity corpora.

The gold data sets provided by the organizers consisted of the 1,088,136 word token (77,065 sentences) manually annotated subcorpus of the National Corpus of Polish (NKJP 1M, see Savary et al. 2012) and a smaller test set of 119,527 word tokens (8 563 sentences). Table 1 shows the numbers of different types of named entity annotations available in these two sets.

If we accept that the size of these annotated corpora warrants some speculations about the frequency of named entities in written Polish texts, then we may observe that around 6% of all tokens in NKJP 1M are part of some named entity. This in turn indirectly proves the importance of NER in natural language processing tasks. Additionally, it is interesting to note that personal names and organisation names are the two most frequent types of named entities in the test and training sets.

Table 1: Statistics of different entity types in the training and test sets

Entity type / subtype	Train tokens number	Test tokens number
persName	1620	182
addName	1060	107
forename	12413	1373
surname	12385	1363
	27478	**3025**
orgName	20504	2294
placeName	341	39
bloc	156	9
country	7506	804
district	344	73
region	1246	153
settlement	7743	922
	17336	**2000**
date	8635	976
geogName	6910	777
time	1430	181

3. The Architecture

The approach to NER described in this paper is directly inspired by recent work on recurrent neural networks and their applications in sequential classification tasks (Ma and Hovy 2016, Lample et al. 2016). Fig. 1 shows an overview of the Tensorflow implementation of the neural network architecture used in our proposed solution. The first important component of this architecture is responsible for transforming word tokens and their features into vector representations, which are then used to train two bidirectional recurrent network layers responsible for the sequential classification of the input word tokens.

3.1. Word Token Representations

We derive three types of word representations from sequences of tokens to be marked with named entity labels. First, vectors of character embeddings are derived from the word tokens and passed through a convolutional subsection of the network with a max-over-time discretization. The resulting character-based representations of token sequences are passed to the input layer of the recurrent section of the network, marked

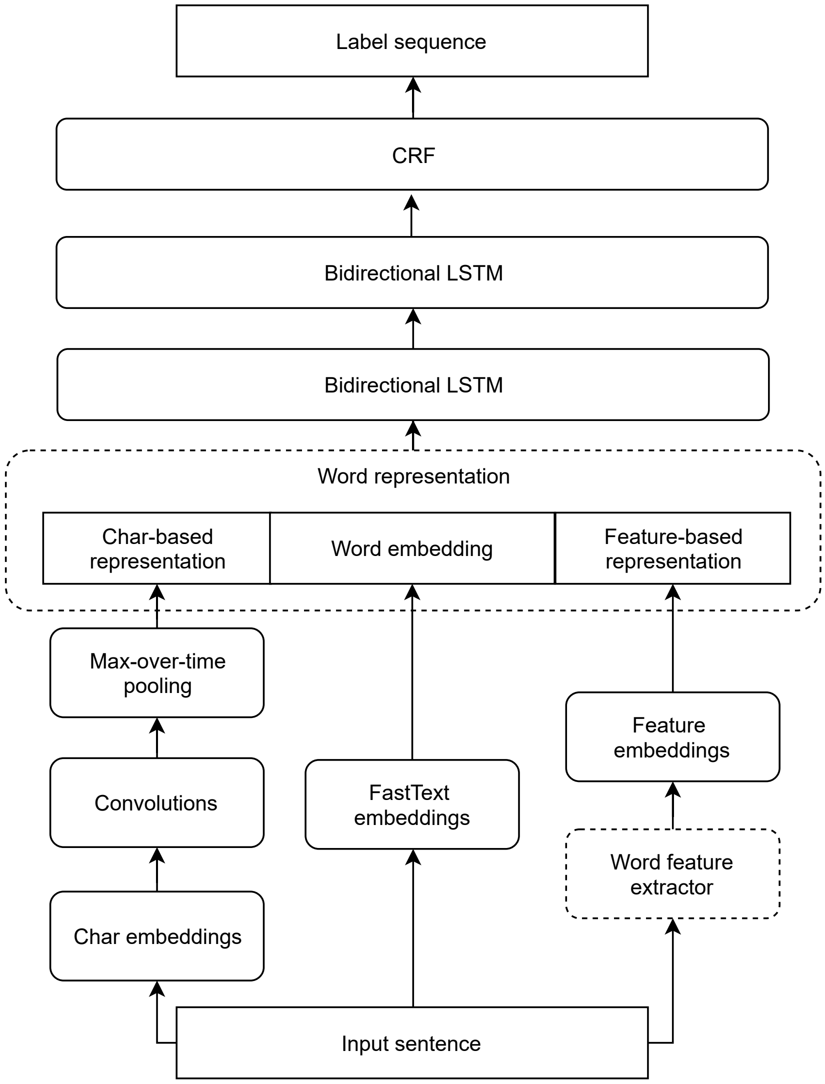


Figure 1: An overview of the recurrent neural network architecture

as `Word Representation` in the diagram. Adding character-based embeddings is motivated by the need to capture certain word shape characteristics of named entities, such as upper case initials in proper nouns which make up named entities (cf. Chiu and Nichols 2015, Ma and Hovy 2016).

Secondly, the input token sequences are matched against 300-dimensional FastText subword embeddings vectors derived from a Common Crawl dump of Polish texts (Grave et al. 2018) and concatenated to the `Word Representation` layer. Interestingly, we have observed a considerable improvement resulting from the application of the Common Crawl embeddings over embeddings computed from the balanced version of the National Corpus of Polish. This may be partly explained by the fact that the former embeddings are not only derived from a much larger corpus, but also case-sensitive, thus preserving an important clue in the task of named entity recognition in written texts.

Finally, main part-of-speech and detailed morphosyntactic labels (selected by applying a frequency threshold of 20 occurrences in the training set) obtained from the APT_PL tagger (Pęzik and Laskowski 2017) were also appended to the word representation layer. The obvious intuition in this case was that named entities are usually composed of word tokens from a narrow set of grammatical categories, such as nouns or adjectives which in Polish also occur in predictable grammatical configurations preserving case, gender and/or number agreement or government relations. This in turn potentially provides more generic clues about the boundaries of named entities.

3.2. Bidirectional LSTMs with a CRF layer

Two bidirectional recurrent layers composed of long short-term memory cells with output dropout values of 0.25 were used to model the context preceding and following the word tokens to be sequentially classified as part of named entities. Additionally, we used a Conditional Random Fields layer to model the restrictions inherent to the BIO annotation of named entities in texts. As a theoretical advantage, using CRFs over a softmax function on the final layer made it possible to jointly model sequence probabilities of the predicted labels rather than separated local probabilities of their outputs (Ma and Hovy 2016).

3.3. Combining Models

Using the network architecture described above, we trained six separate models, one for each of the main types of named entities and its subtypes described in Table 1. The labels are predicted independently by each of the six models are then combined to produce nested entity annotations as specified in the PolEval task. For example, given the sequence of tokens `ulica Adama Mickiewicza`, the `geogName` model is expected to assign the sequence of labels `B-geogName I-geogName I-geogName`. At the same time, the `persName` model is expected to predict the labels `O B-persName-forname I-persName-surname` for this sequence of word tokens. Given these outputs, our algorithm will mark the `persName` tokens of the sequence as fully nested in the `geogName` entity and calculate their character offsets accordingly.

4. Results and Discussion

Table 2 shows the averaged results for overlapping and exact boundary matches of named entities obtained with the official evaluation script. The overall results are marginally better than the scores calculated for our submission by the PolEval organizers due to the fact that we corrected an inconsistent spelling of one of labels in the test set.[1]

More detailed results for both overlapping and exact named entity matching are available in Tables 4 and 3 respectively. In general, there is a considerable difference in the precision and recall scores obtained for different types named entities. For example, forenames were recognized quite well (with an F-score of 0.85–0.88), whereas dates and time expressions, which were more varied, challenging to accurately tokenize and less represented in the training data were recognized with significantly lower accuracy. Such differences in the performance on different types of entities should be considered carefully when comparing different NER systems.

Table 2: Averaged evaluation results obtained on the corrected PolEval test set using the official evaluation script

Matching	**Precision**	**Recall**	**F_1**
Overlapping	0.765	0.786	0.775
Exact	0.685	0.704	0.694

5. Conclusions

The architecture described in this paper was inspired by recent advances in deep learning research and vector-based word representations. One of the important conclusions of our experiments is that differences in the exact configuration, tuning and optimization of seemingly similar neural network architectures may result in significant differences in the rates of accuracy obtained for a particular system. Equally important is the quality and coverage of vector word representations: one of the most considerable improvements we observed in our experiments resulted from using the FastText Common Crawl embeddings in the input layer fed into the bidirectional LSTM layer of our network.

[1]The `persName_addName` label was also spelled as `persName_addname` in the test dataset.

Table 3: Evaluation results for exact matches of different entity types in the test set

Entity type / subtype	Precision	Recall	F_1
persName	0.769	0.845	0.805
addName	0.123	0.323	0.178
forename	0.826	0.886	0.855
surname	0.764	0.831	0.796
orgName	0.612	0.737	0.669
placeName	0.453	0.749	0.527
bloc	0.079	0.632	0.140
country	0.831	0.910	0.869
district	0.254	0.630	0.362
region	0.379	0.704	0.493
settlement	0.742	0.844	0.790
date	0.313	0.426	0.361
geogName	0.484	0.610	0.540
time	0.224	0.380	0.282

Table 4: Evaluation results for partial (overlapping) matches of different entity types in the test set

Entity type / subtype	Precision	Recall	F_1
persName	0.849	0.932	0.888
addName	0.128	0.337	0.186
forename	0.857	0.919	0.887
surname	0.829	0.902	0.864
orgName	0.727	0.876	0.794
placeName	0.501	0.866	0.591
bloc	0.118	0.947	0.211
country	0.865	0.947	0.905
district	0.269	0.667	0.383
region	0.510	0.947	0.663
settlement	0.771	0.877	0.821
date	0.719	0.979	0.829
geogName	0.615	0.775	0.685
time	0.346	0.587	0.436

Acknowledgments

The work described in this paper was carried out within a grant funded by the European Union Regional Operation Program for the Pomeranian Voivodship for the years 2014-2020 based on the Resolution no 190/214/17 of the Pomeranian Voivodship Board, project RPPM.01.01.01-22-0026/16.

References

Chiu J. P. C. and Nichols E. (2015). *Named Entity Recognition with Bidirectional LSTM-CNNs*. „CoRR", abs/1511.08308.

Grave E., Bojanowski P., Gupta P., Joulin A. and Mikolov T. (2018). *Learning Word Vectors for 157 Languages*. [in:] Calzolari N., Choukri K., Cieri C., Declerck T., Goggi S., Hasida K., Isahara H., Maegaard B., Mariani J., Mazo H., Moreno A., Odijk J., Piperidis S. and Tokunaga T. (eds.), *Proceedings of the 11th International Conference on Language Resources and Evaluation (LREC 2018)*, pp. 3483–3487. European Language Resources Association.

Lafferty J. D., McCallum A. and Pereira F. C. N. (2001). *Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data*. [in:] Brodley C. E. and Danyluk A. P. (eds.), *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001)*, pp. 282–289. Morgan Kaufmann.

Lample G., Ballesteros M., Subramanian S., Kawakami K. and Dyer C. (2016). *Neural Architectures for Named Entity Recognition*. „CoRR", abs/1603.01360.

Ma X. and Hovy E. H. (2016). *End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF*. „CoRR", abs/1603.01354.

Marcinczuk M., Stanek M., Piasecki M. and Musiał A. (2011). *Rich Set of Features for Proper Name Recognition in Polish Texts*. [in:] Bouvry P., Kłopotek M. A., Leprévost F., Marciniak M., Mykowiecka A. and Rybiński H. (eds.), *Security and Intelligent Information Systems - International Joint Conferences (SIIS 2011), Revised Selected Papers, Lecture Notes in Computer Science* vol. 7053, pp. 332–344. Springer.

Pęzik P. and Laskowski S. (2017). *Evaluating an Averaged Perceptron Morphosyntactic Tagger for Polish*. [in:] Vetulani Z. and Paroubek P. (eds.), *Proceedings of the 8th Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics (LTC 2017)*, pp. 372–376. Fundacja Uniwersytetu im. Adama Mickiewicza w Poznaniu.

Savary A., Chojnacka-Kuraś M., Wesołek A., Skowrońska D. and Śliwiński P. (2012). *Anotacja jednostek nazewniczych*. [in:] Przepiórkowski A., Bańko M., Górski R. L. and Lewandowska-Tomaszczyk B. (eds.), *Narodowy Korpus Języka Polskiego*, pp. 129–165. Wydawnictwo Naukowe PWN, Warsaw.

KNER: Named Entity Recognition for Polish

Krzysztof Wróbel (Jagiellonian University and AGH University of Science and Technology), **Aleksander Smywiński-Pohl** (AGH University of Science and Technology)

Abstract

The article presents named entity recognition system, which participated in the second task of PolEval 2018 competition. It utilizes recurrent and convolutional neural networks with conditional random fields. The only external resources are provided by the morphological tagger KRNNT and word embeddings. The distinctive aspect of the solution is the lack of use of gazetteers or lexicons. Two approaches are presented to address nested annotation of named entities, each with its own advantages. The solution obtains 81.4% F_1 measure.

Keywords

Named Entity Recognition, NER, recurrent neural networks, Polish

1. Introduction

Named entity recognition (NER) is a natural language processing task, which aim is to find all named entities in a text and classify to pre-defined categories (usually names of person, locations, organizations). It is a very important step in NLP pipelines because named entities should be treated as one unit.

Available Polish NER tools Nerf (Savary et al. 2010) and Liner2 (Marcińczuk et al. 2013) utilize CRF as the main learning algorithm and exploits gazetteers. Nerf supports nested categories. Other research focused on a knowledge-based system without the need of training data (Pohl 2013).

The state-of-the-art NER solutions for English utilizes combination of neural networks (recurrent and convolutional) with conditional random fields (CRF; Huang et al. 2015, Ma and Hovy 2016) achieving above 91% F_1 score.

This work presents NER system named KNER, which uses neural networks combined with CRF and morphological tagger for feature generation. The solution participated in the second task of PolEval 2018 competition.

2. Data

National Corpus of Polish (NKJP; Przepiórkowski et al. 2012) was used as training data. It is well-balanced corpus, which includes articles from newspapers and journals, transcriptions of spoken conversations, and user-generated content from web forums. The corpus was manually annotated by 2 annotators and 1 person who was resolving disagreements.

In comparison to other NER datasets, NKJP uses nested annotation, so simple data format (e.g. CONLL) cannot be used. Example of nested annotation:

$$[[\text{John}]_{forename}\ [\text{Smith}]_{surname}]_{persName} \text{ eats a cake.}$$

In the NKJP following named entity categories and its subcategories are recognized:

- persName (person name)
  - forename
  - surname
  - addName (alias, pseudonym, nickname)
- orgName (organization name)
- geogName (geographical name)
- placeName (geopolitical name)
  - district
  - settlement
  - region
  - country
  - bloc (geopolitical unit comprising two or more countries)
- date
- time

Even single personal entities have to be annotated twice, e.g. John as *forename* and also *persName*. It disrupts the accuracy of Polish NER tools because it can be automatically inferred.

$$[[\text{John}]_{forename}]_{persName} \text{ eats a cake.}$$

What is more, derivative phrases are annotated, e.g. *Polish* is a derivative of class *country*.

For the purpose of PolEval, organizers provided new testing data. In the evaluation process derivative named entities are treated as non-derivative.

One error, related to the name of category, is present in the testing data: almost whole (91%) *addName* annotations are wrongly spelled (*addname* — small letter *n*) and they decrease the scores by almost 1.8 percentage points. However, it can be simply corrected e.g. by lower casing categories in comparison in the official testing script. In the script is also a bug discarding some annotations — updated version is available at `https://github.com/kwrobel-nlp/poleval2018-ner`.

Distributions of categories in training and testing data are presented in Fig. 1. Training data is more than two times larger than testing data and it consists of 86793 annotations, where the testing data consists of 40220 annotations.

2.1. Adaptation

Two approaches have been employed to address nested annotation and in the end allow use of CONLL format and IOB/BIO encoding. The encoding labels each token as *B-category* if the token **b**egins a named entity, *I-category* if the token is **i**nside the named entity and *O* for **o**ther tokens not in named entities.

In both solutions more than one model is trained and the results are combined.

2.2. First Approach

The first model recognizes the first level named entities (the longest). The second model recognizes second level entities among the first level entities.

Firstly, all nested named entities are discarded. It reduces number of named entities to 64.15%. Example sentence:

$$[[\text{John}]_{forename}\ [\text{Smith}]_{surname}]_{persName} \text{ eats a cake in } [\text{Cracow}]_{settlement}.$$

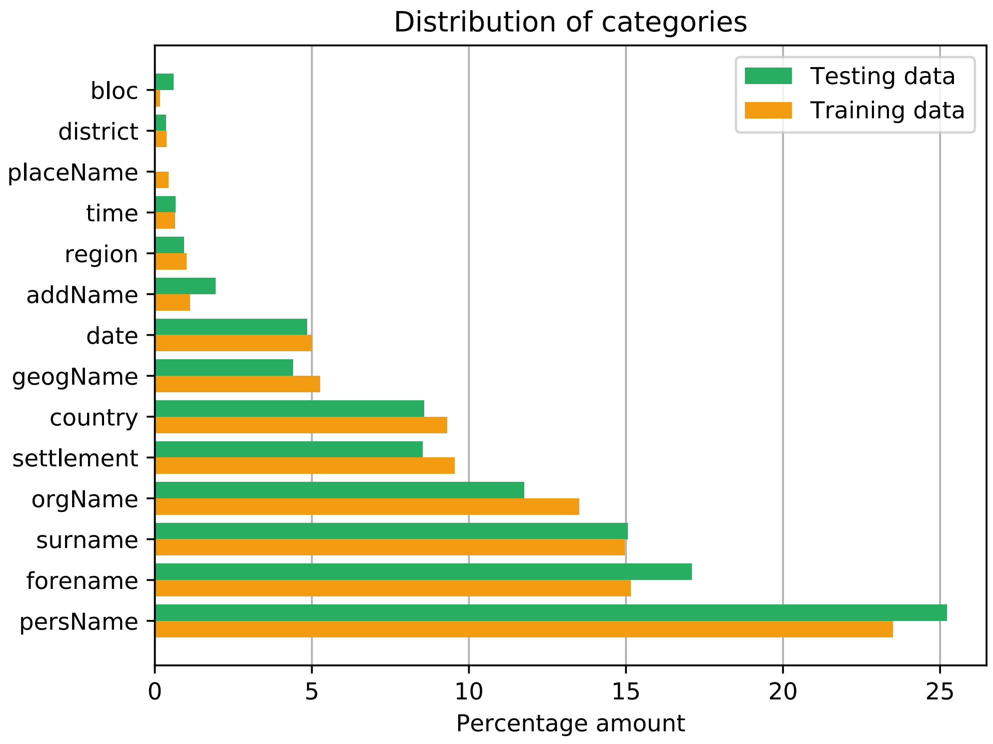


Figure 1: Distributions of categories in training and testing data (*addName* category is corrected)

is transformed to:

$$\text{[John Smith]}_{persName} \text{ eats a cake in [Cracow]}_{settlement}.$$

Then all named entities are extracted with assigned labels as a feature in IOB encoding. New training dataset is created with annotated only nested entities.

The sentence presented above produces 2 samples (features are defined in superscript):

$$[\text{John}^{B-persName}]_{forename}\ [\text{Smith}^{I-persName}]_{surname}$$

$$\text{Cracow}^{B-settlement}$$

This whole approach covers 99.97% of all named entities. Multiple nesting is rare.

In conclusion, two models are trained: the first general and the second only for detection of nested NE.

2.3. Second Approach

For each category, a separate model is trained. However, there are cases when named entities with the same category are nested (1.1% of all named entities), e.g. [[Wawel]$_{geogName}$ Castle]$_{geogName}$ — they are discarded.

Example sentence:

[[John]$_{forename}$ [Smith]$_{surname}$]$_{persName}$ eats a cake.

is transformed to:

— for *persName* model: [John Smith]$_{persName}$ eats a cake.

— for *forename* model: [John]$_{forename}$ Smith eats a cake.

— for *surname* model: John [Smith]$_{surname}$ eats a cake.

— for other models: John Smith eats a cake.

3. Features

Input sentences are tokenized using morphological tagger KRNNT (Wróbel 2017).

Features are assigned to tokens. The main feature is a word form. Additional features based on KRNNT and Liner2 (Marcińczuk et al. 2013) were added (dimension of feature embeddings is provided in parenthesis):

— shape (10) — 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. *Wrobel2018* gives *ulllllldddd* and after collapsing *uld*),

— 3 characters prefix (10),

— 3 characters suffix (10),

— whether token is all upper case (1),

— whether token is all lower case (1),

— whether first character of token is upper case and the rest lower case (1),

— whether token is a number (1),

— whether space is before token (1),

— base form (50),

— full tag (20),

— part-of-speech (10),

— case (5),

— person (5),

— number (5),

— gender (5).

Some of the features are overlapping, e.g. a case of characters with a shape feature. Also, character word embeddings should be able to infer these form-based features.

4. Network Architecture

The solution is based on a neural network combined with CRF (Yang et al. 2018). The first layer is a bidirectional long short-term memory (LSTM). On input are features represented by embeddings. There is one special input: embedding of word form computed on its characters. The character word embedding is generated by convolutional neural network (CNN). The output of the first layer is connected to the second bidirectional LSTM and CRF is at the end.

5. Results

The network was trained using stochastic gradient descent by 100 iterations. Word embeddings were trained on full NKJP (Mykowiecka et al. 2017).

An evaluation was performed using F-measure for an exact and overlapping (partial) match. Final score is calculated as weighted average between them ($0.8 \cdot exact\_score + 0.2 \cdot overlap\_score$).

Official results of PolEval 2018 are presented in Table 1. Systems KNER_v1 and KNER_v2 employ the first and the second approach respectively. However, KNER solutions have been wrong, because they discarded all entities marked as derivative. After correcting the mistake, it could take the third place in the ranking. More detailed scores are presented in Table 2. The mistake cost 12 percentage points in the final score.

Table 1: Results of PolEval 2018 task 2. Final score is weighted average between exact and overlap score

System	**Exact**	**Overlap**	**Final**
Per group LSTM-CRF with Contextual String Embeddings	**0.826**	**0.877**	**0.866**
PolDeepNer	0.822	0.859	0.851
Liner2	0.778	0.818	0.810
OPI_Z3	0.749	0.805	0.793
joint	0.748	0.789	0.780
disjoint	0.747	0.788	0.779
via_ner	0.692	0.773	0.756
KNER_v2	0.700	0.742	0.733
Poleval2k18	0.623	0.743	0.719
KNER_v1	0.681	0.719	0.711
simple_ner	0.569	0.653	0.636

Table 2: Results of KNER system employing second approach with fixed mistake

Measure	**Precision**	**Recall**	**F$_1$**
Exact	0.913	0.808	0.857
Overlap	0.867	0.767	0.814
Final			**0.823**

6. Conclusion

Presented solutions have some drawbacks. In the first approach, nested entities are being recognized without the full context of a sentence. The second approach needs many models to run and does not support nested entities within the same category. The solutions do not utilize any gazetteers or lexicons and achieve satisfactory results.

An error analysis showed that the same token in the same paragraph can be classified differently by the presented neural network. This information could be exploited to improve scores.

Acknowledgments

This work was supported by the Polish National Centre for Research and Development LIDER Program under Grant LIDER/27/0164/L-8/16/NCBR/2017 titled “Lemkin —

inteligentny system informacji prawnej”. This research was also supported in part by PLGrid Infrastructure.

References

Huang Z., Xu W. and Yu K. (2015). *Bidirectional LSTM-CRF Models for Sequence Tagging*. „CoRR”, abs/1508.01991.

Ma X. and Hovy E. H. (2016). *End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF*. „CoRR”, abs/1603.01354.

Marcińczuk M., Kocoń J. and Janicki M. (2013). *Liner2 – a Customizable Framework for Proper Names Recognition for Polish*. [in:] Bembenik R., Skonieczny L., Rybiński H., Kryszkiewicz M. and Niezgódka M. (eds.), *Intelligent Tools for Building a Scientific Information Platform*, pp. 231–253.

Mykowiecka A., Marciniak M. and Rychlik P. (2017). *Testing Word Embeddings for Polish*. „Cognitive Studies / Études Cognitives”, 17, pp. 1–19.

Pohl A. (2013). *Knowledge-based Named Entity Recognition in Polish*. [in:] *Proceedings on 2013 Federated Conference on Computer Science and Information Systems (FedCSIS 2013)*, pp. 145–151. IEEE.

Przepiórkowski A., Bańko M., Górski R. L. and Lewandowska-Tomaszczyk B. (eds.) (2012). *Narodowy Korpus Języka Polskiego*. Wydawnictwo Naukowe PWN, Warsaw.

Savary A., Waszczuk J. and Przepiórkowski A. (2010). *Towards the Annotation of Named Entities in the National Corpus of Polish*. [in:] Calzolari N., Choukri K., Maegaard B., Mariani J., Odijk J., Piperidis S., Rosner M. and Tapias D. (eds.), *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC 2010)*, European Language Resources Association.

Wróbel K. (2017). *KRNNT: Polish Recurrent Neural Network Tagger*. [in:] Vetulani Z. and Paroubek P. (eds.), *Proceedings of the 8th Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics*, pp. 386–391. Fundacja Uniwersytetu im. Adama Mickiewicza w Poznaniu.

Yang J., Liang S. and Zhang Y. (2018). *Design Challenges and Misconceptions in Neural Sequence Labeling*. [in:] *Proceedings of the 27th International Conference on Computational Linguistics (COLING)*.

Flat Approach to Finding Nested Named Entities

Paulina Żak (Polish Japanese Academy of Information Technology)

Abstract

We describe a system implemented for recognizing Named Entities evaluated as a part of task 2 in PolEval 2018, the competition designed for the creation of natural language processing tools in Polish. The presented system uses Bidirectional GRU neural network, trained on 1-million word subcorpus from the National Corpus of Polish, with word embeddings and binary features as an input and flat entities as an output. Two types of approaches were tested for flattening nested named entities in the data, removing all but one label for each word and concatenating labels of each word into a new category.

Keywords

Natural Language Processing, Named Entity Recognition, NER, GRU

1. Introduction

Recognition of Named Entities (NER) is a well-known problem in which the system locates and classifies specific phrases into categories describing a real-world objects that can be denoted with a proper names. List of what is considered to be named entity varies by the usage. Most datasets, such as CONLL-2003 (Sang and Meulder 2003), limit themselves only to general categories such as Person, Location and Organization, as shown in Fig. 1. Others, such as Sekine Extended Named Entity Hierarchy (Sekine et al. 2002), add more varied categories for e.g. God, Product or Disease. The numbers of categories also differ between tasks, from simple 4 classes in CONLL-2003 (Sang and Meulder 2003) to fine-grained with multiple levels of hierarchy such as HYENA (Yosef et al. 2012).

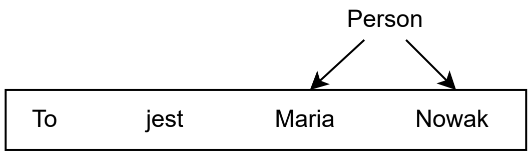


Figure 1: Example of the text with named entity "This is Maria Nowak", where "Maria Nowak" is tagged as the entity

Named Entity Recognition is an important task in processing text information. Found and processed entities are not only valuable on their own but they can be a feature or input for multiple other tasks such as machine translation, knowledge graphs creation or question answering. Thanks to such a varied usage in Natural Language Processing, multiple different datasets and competitions were created to enhance the performance of the NER systems. PolEval 2018 Task 2 is one of them and focuses on targeting information written in the Polish language.

2. Previous Approaches

Interest in Named Entity Recognition started to rise around 1995 with the first major event related to the task MUC-6 (Nadeau and Sekine 2007). Back then most of the solutions were based on rules and gazetteers, however, the task have evolved through the years with more complex systems. With more computing power and new research in AI field, the focus changed from man-made solutions to machine learning approaches.

2.1. Rule-based

Early tasks were dominated by rule based solutions (Jyoti Mahanta 2013). In those systems man-made rules were created and applied to input text in search for named entities. Those rules were created based on syntactic, linguistic and domain knowledge. In the recent years, most of rule-based systems were replaced by supervised learning, with models trained on corpora labeled with named entities. Although, if the training corpus is not available, rule-based systems are still viable solutions (Nadeau and Sekine 2007).

2.2. Gazetteers

Systems based on gazetteers, frequently with the addition of rules, were also a popular choice for early Named Entity Recognition. Gazetteer consists of some external knowledge source containing names of entities such as locations, names of famous people, organizations etc. This approach requires either the hand crafting of name lexicons or some dynamic approach to obtaining a gazette from the corpus or another external source (Data Community 2013). Depending on the task they can still be used as a feature to machine learning systems.

2.3. Machine Learning

Currently most of the systems are based on machine learning and especially neural networks. The number of possible architectures for machine learning NER systems are vast and the decision of which one to choose depends on many variables. In the case of no annotated data, semi-supervised or even unsupervised NER systems can be created. However in the standard case when a corpus is available, supervised methods are still the most popular. In these approaches, NER is essentially posed as a classification problem. The most frequently applied technique in the CoNLL-2003 shared task was the Maximum Entropy Model. Five systems used this statistical learning method (Sang and Meulder 2003). In the last couple of years, statistical models were overtaken by neural networks. Currently, for CONLL-2003 dataset the state of the art was achieved by Bidirectional LSTM-CRF Model (Huang et al. 2015) with another interesting model that achieves high results in this task being Bidirectional LSTM-CNN (Chiu and Nichols 2015).

3. The Data

3.1. 1-million-word Subcorpus

The training dataset for PolEval 2018 Task 2 was 1 million word sub-corpus created from National Corpus of Polish (Savary et al. 2012). The data set has undergone manual verification of named entities. In the most popular English datasets, CONLL-2003 (Sang and Meulder 2003) and MUC-7 (Chinchor 2001), the annotation of the corpora were made as simple as possible. Those tasks were created with a purpose of automatically detecting and categorizing entities in mind and adding more complexity to the task would not be beneficial.

The taxonomy of named entities in the National Corpus of Polish has average complexity, we present it in Fig. 2.

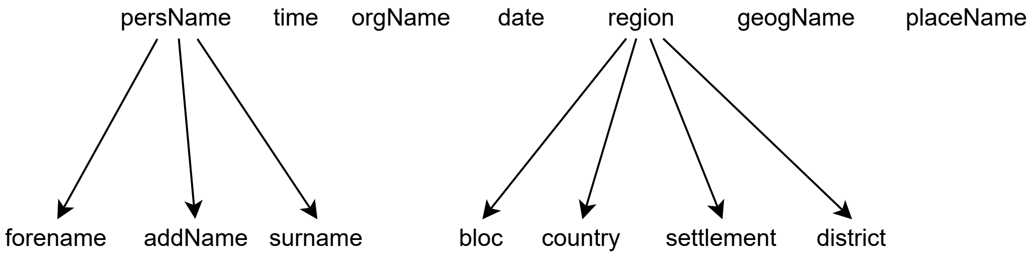


Figure 2: Named entity hierarchy in National Corpus of Polish

We want to highlight three key aspects:

1. **Labels hierarchy**. The named entity hierarchy consists of two levels of categories, which makes in total 14 possible labels to annotate (Fig. 2).
2. **Nested phrases**. Different labels can overlap with each other, if part of the named entity is a different named entity in itself (Fig. 3).
3. **Non-continuous named entity**. When two or more words are a part of one entity but in between them other words are placed (Fig. 4).

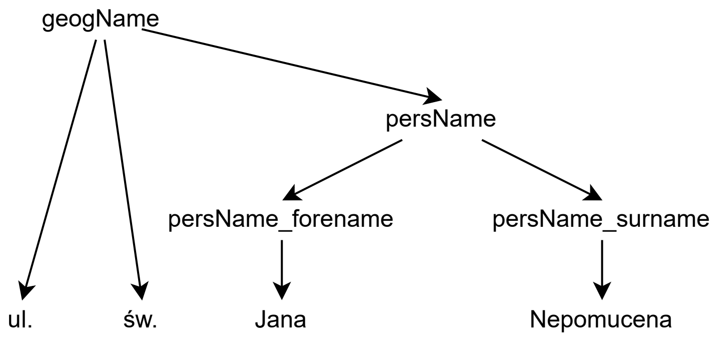


Figure 3: A sentence with nested entity "Saint John Nepomucene street" where the whole sentence is categorized as geogName and "John Nepomucene" is persName

3.2. Nested Entities

In this paper we want to focus on the impact of nested phrases on the performance of the systems. Nested named entities were not as widely developed in previous

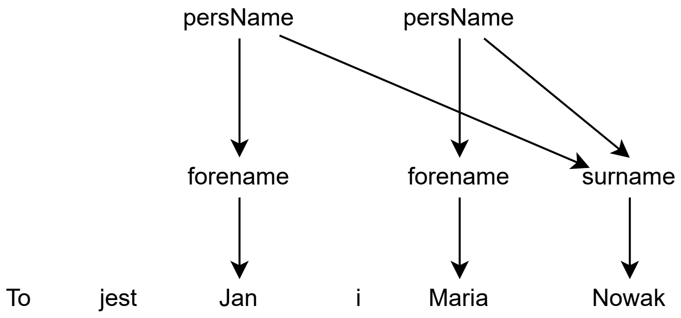


Figure 4: A sentence with non-continuous named entities “This is Jan and Maria Nowak” where both “Jan Nowak” and “Maria Nowak” are named entities

tasks mostly due to practical reasons (Finkel and Manning 2009). The most known datasets such as MUC and CONLL were flatly annotated and ignored any nesting all together. However some corpora, particularly in biomedical domain, were created with nested entities in mind. For example 17% of the entities in the GENIA corpus are embedded within another entity and in the ACE corpora, 30% of sentences contain nested entities (Katiyar and Cardie 2018).

Thanks to the design of the National Corpus of Polish, we can analyze the nested data in this corpus. In the 1-million sub-corpus, 3599 words were found to be part of the nested entity, which makes $\tilde{4}.1\%$ of all words classified as named entity.

4. RNN and GRU

4.1. Recurrent Neural Network

A recurrent neural network (RNN) is a machine learning model designed to handle sequential data. It is designed to add information about the previous parts of the sequence into present input. However due to an issue called vanishing gradient, simple RNN has difficulty to retain more information than few previous inputs. One of the most popular solutions to this issue (next to Long-Short Term Memory Network) is Gated Recurrent Unit network architecture, which will be described in a paragraph below.

4.2. GRU

GRU is a mechanism introduced in 2014 by Cho et al. (2014). To solve the vanishing gradient problem of a standard RNN, GRU introduces update gate and reset gate.

Those gates can decide what information will be passed into the next cell in the sequence. With this ability they can retain the most important information even in longer sequences. In comparison to LSTM, GRUs have been shown to perform better on smaller datasets (Chung et al. 2014).

4.3. Bidirectional RNN

Bidirectional RNN (BRNN) means using two RNNs, one for reading the text input from beginning to the end (forward) and the other — backwards. Outputs of both RNN networks are then concatenated for each token. Thanks to this architecture the neural network can use information about preceding and following words in a sentence.

5. System

5.1. Input

Each document in the dataset was split to words and divided into two parts, embedding and binary feature. For the system we have used word embeddings vectors of 300 dimensions for Polish, trained on Wikipedia (Bojanowski et al. 2016). Those embeddings were also additionally trained during NER model training. Three types of binary features were included. For each word in a sequence, they answer the following questions:

1. Does the word start with a capital letter?
2. Does the word has a dot in it?
3. Does the word has a number in it?

5.2. Architecture

The model was created in Python using Keras API with Tensorflow backend. We have created a two-input model using functional API. Trainable word embeddings were use as the first input and binary features as the second one. After one Bidirectional GRU layer, dropout of value 0.5 was applied. As the last element, dense layer with softmax was added.

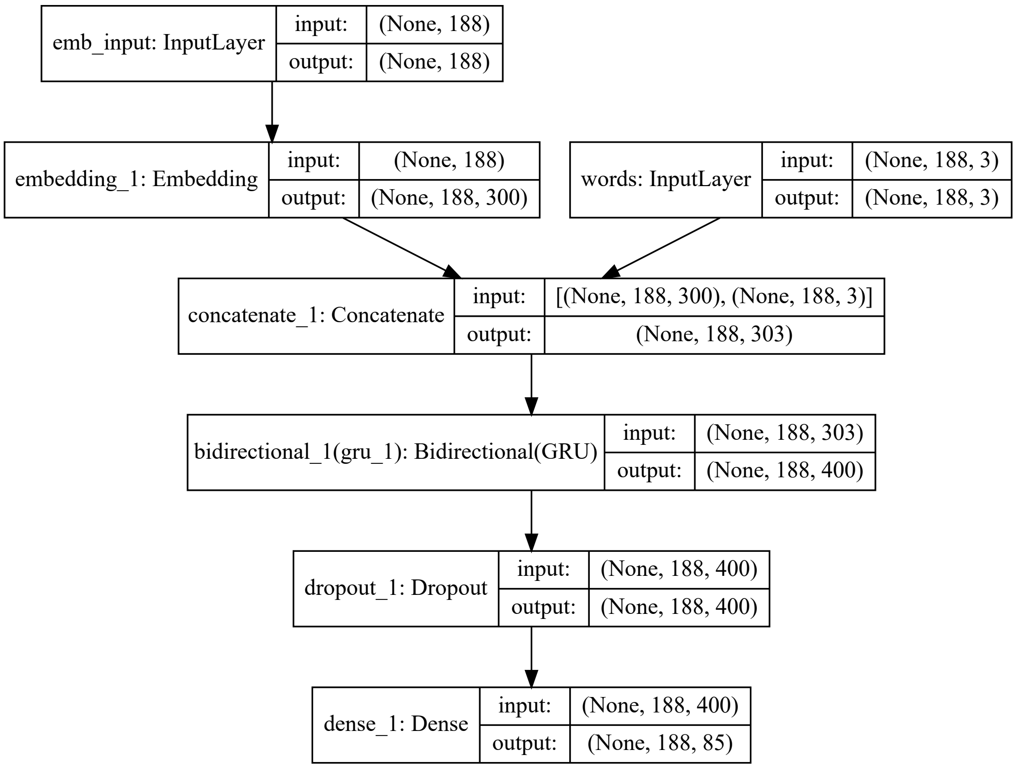


Figure 5: Architecture of trained network

5.3. Output

For the output we have chosen a simple IO tagging. We have created two versions of the system, differing only in the structure of the output. The differences are illustrated in Table 1 by a comparison in the sentence from Fig. 3.

In the unprocessed dataset we would have 8 labels in total: 4 labels geogName, 2 persName, 1 persName_surname and 1 persName_forename.

The first system modifies all overlaping labels in nested entity by concatenating it into one label. After preprocessing the system would produce 4 labels: 2 geogName, 1 label geogName-persName-persName_forename and 1 geogName-persName-persName_-surname.

The second system would only use the most specific value in the set which would remove all nested phrasing. After preprocessing the system would produce 4 labels: 2 geogName, 1 label persName_forename and 1 persName_surname.

Unfortunately none of the solutions was adapted to properly categorize non-continuous categories and will not be able to assemble them properly.

Table 1: Example of the output labeling for a sentence

Type	**Sentence**				**# of**
	ul.	św.	Jana	Nepomucena	**labels**
Original	geogName	geogName	geogName persName, persName _foreName	geogName persName, persName _surname	8
Concatenated	geogName	geogName	geogName-persName _forename	geogName-persName-persName _surname	4
One label	geogName	geogName	persName _forename	persName _surname	4

5.4. Concatenated Labels Analysis

From 14 original labels after concatenation 84 labels were created. The distribution of labels was highly imbalanced as almost half of them occurred fewer than 10 times in the training set, as in Table 2.

Table 2: Distribution of the number of occurrences per labels

Occurrences	**Number of labels**
10000 or more	5
1000 or more	13
100 or more	26
10 or more	43
1 or more	85

6. Evaluation

The system was evaluated on the test set provided by organizers of PolEval 2018. The evaluation script measured the performance of the system in two ways, if either whole or only partial entities were matched:

- exact match — only fully matched named entities were counted,
- overlap match — if gold and predicted named entities matched partially they were counted as a true positive.

For each evaluation type three metrics were calculated:

- precision — ratio of correctly predicted positive observations to the total predicted positive observations,
- recall — ratio of correctly predicted positive observations to all observations in the class,
- F_1 score — the weighted average of precision and recall.

We have evaluated two types of output design as described in Section 5, as well as impact of different types of binary features on the performance of the system. The results are presented in Tables 3 and 4.
Concatenated system performed better than one-label solution by a significant margin. This is probably due to a missing information of more generic entities from the hierarchy. The phrase "Anna Nowak", should have 3 labels, Anna — persName_-foreName, Nowak — persName_surname, "Anna Nowak" — persName. But the system with one label output, by design, cannot process labels higher in the hierarchy of categories. We have tried to generate the more generic entities by merging labels with subtypes in post-processing scripts but the results still not matched the results from concatenated system.

Table 3: Overlap match results of different version of the system with features

Output	**Features**	**Precision**	**Recall**	**F_1 score**
Concatenated	none	0.873	0.554	0.678
Concatenated	first capital letter	0.826	0.618	0.707
Concatenated	has dot	0.865	0.559	0.680
Concatenated	is digit	0.884	0.553	0.681
Concatenated	**all**	**0.852**	**0.615**	**0.714**
One label	all	0.837	0.498	0.625

7. Conclusion and Future Work

In this paper we have shown how a simple, flat system can recognize nested named entities. In our system the best results, for both exact as well as overlap matching, were achieved by the version with all three binary features used and the output

Table 4: Exact match results of different version of the system with features

Output	**Features**	**Precision**	**Recall**	**F_1 score**
Concatenated	none	0.746	0.474	0.580
Concatenated	first capital letter	0.731	0.547	0.626
Concatenated	has dot	0.729	0.471	0.573
Concatenated	is digit	0.748	0.469	0.576
Concatenated	**all**	**0.753**	**0.543**	**0.631**
One label	all	0.682	0.406	0.509

entities concatenated into one label. However, more work can be done in assessing the best method of finding and labeling nested entities in text. Firstly, other neural networks architectures and machine learning methods should be researched, such as LSTM-CRF(Conditional Random Fields) (Lample et al. 2016) or LSTM-CNN (Chiu and Nichols 2015) which achieves state of the art results for English language datasets in the NER task. After setting a proper baseline on a flat system, the research might go more deeply into understanding the information transmitted by the nested structure. With one of the interesting possibilities being usage of TreeLSTM architecture, with tree representation of the sentences, applied successfully to other tasks such as sentiment analysis (Tai et al. 2015).

References

Bojanowski P., Grave E., Joulin A. and Mikolov T. (2016). *Enriching Word Vectors with Subword Information*. „CoRR", abs/1607.04606.

Chinchor N. (2001). *Message Understanding Conference (MUC) 7. LDC2001T02. Linguistic Data Consortium Web download.*

Chiu J. P. C. and Nichols E. (2015). *Named Entity Recognition with Bidirectional LSTM-CNNs*. „CoRR", abs/1511.08308.

Cho K., van Merrienboer B., Gülçehre Ç., Bougares F., Schwenk H. and Bengio Y. (2014). *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*. „CoRR", abs/1406.1078.

Chung J., Gülçehre Ç., Cho K. and Bengio Y. (2014). *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. „CoRR", abs/1412.3555.

Data Community (2013). *A Survey of Stochastic and Gazetteer Based Approaches for Named Entity Recognition — Part 2*. `http://www.datacommunitydc.org/blog/2013/04/a-survey-of-stochastic-and-gazetteer-based-approaches-for-named-entity-recognition-part-2`.

Finkel J. R. and Manning C. D. (2009). *Nested Named Entity Recognition*. [in:] *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1* (EMNLP'09), pp. 141–150. Association for Computational Linguistics.

Huang Z., Xu W. and Yu K. (2015). *Bidirectional LSTM-CRF Models for Sequence Tagging*. „CoRR", abs/1508.01991.

Jyoti Mahanta H. (2013). *A Study on the Approaches of Developing a Named Entity Recognition Tool*. International Journal of Research in Engineering and Technology 02, pp. 58–61.

Katiyar A. and Cardie C. (2018). *Nested Named Entity Recognition Revisited*. [in:] *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 861–871. Association for Computational Linguistics.

Lample G., Ballesteros M., Subramanian S., Kawakami K. and Dyer C. (2016). *Neural Architectures for Named Entity Recognition*. „CoRR", abs/1603.01360.

Nadeau D. and Sekine S. (2007). *A Survey of Named Entity Recognition and Classification*. „Lingvisticae Investigationes", (1), pp. 3–26.

Sang E. F. T. K. and Meulder F. D. (2003). *Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition*. „CoRR", cs.CL/0306050.

Savary A., Chojnacka-Kuraś M., Wesołek A., Skowrońska D. and Śliwiński P. (2012). *Anotacja jednostek nazewniczych*. [in:] Przepiórkowski A., Bańko M., Górski R. L. and Lewandowska-Tomaszczyk B. (eds.), *Narodowy Korpus Języka Polskiego*, pp. 129–165. Wydawnictwo Naukowe PWN, Warsaw.

Sekine S., Sudo K. and Nobata C. (2002). *Extended Named Entity Hierarchy*. [in:] *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC 2002)*, pp. 1818–1824. European Language Resources Association.

Tai K. S., Socher R. and Manning C. D. (2015). *Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks*. „CoRR", abs/1503.00075.

Yosef M. A., Bauer S., Hoffart J., Spaniol M. and Weikum G. (2012). *HYENA: Hierarchical Type Classification for Entity Names*. [in:] *Proceedings of COLING 2012: Posters*, pp. 1361–1370. The COLING 2012 Organizing Committee.

Results of the PolEval 2018 Shared Task 3: Language Models

Grzegorz Wojdyga (Institute of Computer Science, Polish Academy of Sciences)

Abstract

PolEval is a national competition for Natural Language Processing researchers that allows them to create and validate tools for Polish language. It is inspired by SemEval competition and is organized similarly — the organizers provides data and evaluation rules. In 2018 there were three tasks — Dependency Parsing, Named Entity Recognition and Language Models. This paper describes motivation, preparation and results of the Language Models task.

1. Introduction

Natural Language Processing is one of the fastest-growing branch of Artificial Intelligence. For many tasks in NLP there is no possibility to transfer solutions from one natural language to another, so there is a necessity to create them independently for Polish language. This was the most important motivation for Institute of Computer Science to create PolEval. The goals of this competition are to provide free data in Polish language, encourage Polish researchers to create tools for Polish language and to make an opportunity of networking.

Last year's tasks at PolEval concentrated on POS tagging and sentiment analysis. This year in the second edition of PolEval we have decided to focus on Dependency Parsing, Named Entity Recognition and Language Models. This paper describes the Language Model task. There is a description of the task in Section 2. In Section 3 there is a summary of previous Polish open-source language model and information about data which was used to build them. In Section 4 there is a description of gathering and preparing data for this task. Section 5 contains information about evaluation that was chosen. The methods that were submitted by participants are presented in Section 6. Finally, Section 7 contains summary.

2. Task Description

The idea behind this task was very simple — encourage researchers to create the modern language model for Polish. Language model should assign a probability to each possible next word based on the previous words. In the latest years there is a rapid growth of methods for creating language models. First language models were purely statistical n-gram models, recently they are mostly done using recurrent neural network e.g. Bengio et al. (2003) and Mikolov et al. (2010). In english there is ongoing research on this topic. There are many corpuses for english, the most famous is probably Penn Treebank (Marcus et al. 1993), but nowadays the most popular one is "One billion word benchmark" created by Google (Chelba et al. 2013). We have decided to follow Google approach, hence we provided huge corpus (half billion words) in which the sentenced were shuffled and tokenized. Moreover we have chosen that evaluation metric would be perplexity, which is the most popular metric for Language Models. Participant had access to train data and later to test data. They have measured the perplexity of their models on the test date following the instructions and submit the result to organizers.

3. Previous work

Previous work in the field of language models is quite limited. Many companies have developed language models for Polish but they haven't shared their models due to licenses policy. Academic research focused mostly on statistical methods like n-grams — by Ziółko and Skurzok (2011), Wołk et al. (2017), Banasiak et al. (2017). Some language models were developed for Automated Speech Recognition e.g. Kłosowski (2017) and Rapp (2008).

4. Train and Test Data

In order to create the best language model the data should have been gathered in a very large amount and from various sources. We could have use only the databases with licenses that allowed to reuse for scientific purposes. We have decided to use

Table 1: Sources of the corpus

Source	Sentence count	Percentage
books	44348	0,17 %
NKJP	40756	0,15 %
Sejm	9461662	37,01 %
forums	6954771	27,20 %
Wikipedia	9066909	35,46 %
ALL	25568446	100,00 %

Polish Wikipedia[1], Internet forums, Polish books — wolnelektury.pl[2], National Corpus of Polish (Przepiórkowski et al. 2012)[3] and Polish Parliament Corpus (Ogrodniczuk 2012)[4]. The numbers of sentences and the percentage of all the content are presented in Table 1.

We have extracted the sentences from them. Later we have tokenized them using perl script similar to these provided by "One billion word benchmark" by Google[5]. Then we left only those that had more than five tokens in it. Example of tokenized sentences are presented in listing 13.1. Later we have divided the data for train and test set so that ratio of data from different sources would be same both in train and test set. The trainset was 90% of all data and testset was 10%. In the end we have shuffled sentences.

```
W tym roku Adam jest studentem .
Z tymi sprawami sobie poradzimy .
W tej sytuacji wycofali swoje uwagi .
```

Listing 13.1: Example of tokenized data

As for the dictionary we have followed Google's "One billion word benchmark" approach. We have decided to create vocabulary from the words from trainset. Every token that occurred three or more time was included to dictionary, every token that occurred less than three time was excluded and replaced with <UNK> token. The testset was transformed with trainset dictionary — any word in testset that doesn't appear in trainset dictionary is replaced with <UNK>.

[1] https://dumps.wikimedia.org/plwiki

[2] https://wolnelektury.pl/api

[3] http://www.nkjp.pl

[4] http://clip.ipipan.waw.pl/PSC

[5] https://github.com/ciprian-chelba/1-billion-word-language-modeling-benchmark

5. Evaluation Procedure

We have decided to use perplexity metric. Perplexity is a measure commonly used to estimate the quality of a language model (see e.g. Google One Billion Word Benchmark). Perplexity is the inverse probability of the test set, normalized by the number of words.

The language model should estimate the possibility of occurrence for every word in every sentence. The probability should be calculated for every word and the sentence ending in the test set.

Perplexity can be calculated using the following equation:

$$PPL = \exp(\frac{\Sigma_{i=1}^{N} \log_e(\frac{1}{q(x_i)})}{N})$$

where:

— N — number of samples in the test set,

— X — discrete random variable with possible values $x_1, ..., x_n$,

— $q(X)$ — probability function.

After transformation of the tokens in testset using trainset dictionary, participants were asked to evaluate perplexity following standard procedure. The short example was provided on webpage [6]

Moreover, to verify if participants were using the same vocabulary we have asked them to provide OOV rate for test set.

6. Submitted solutions

There were three participant who submitted 11 results.

The best language model was developed by Piotr Czapla and Marcin Kardas from n-waves company. They used Universal language model fine-tuning for text classification (Howard and Ruder 2018) with subword tokenization using SentencePiece (Kudo 2018). Universal Language Model for Fine-tuning [6] (ULMFiT) is one of the first NLP methods for efficient inductive transfer learning. Unsupervised pretraining results in improvements on many NLP tasks for English.

[6]http://poleval.pl/files/8515/2940/9834/ExampleOfPerplexity.pdf

Table 2: Results of Language Model task

Model	**Perplexity**
ULMFiT-SP-PL	117.6705
AGHUJ	146,7082
PocoLM Order 6	208.6297
PocoLM Train + Semantic	211.6264
SRILM Interpolate Witten-Bell	216.3766

Authors adopted ULMFiT for Polish. Their model is based on the fastai[7] implementation of ULMFiT, that was extended with subword tokenization provided by Sentence Piece. The model is a 4 layered LSTM with multiple dropouts and trained using Slanted Triangular Learning Rate. The language model is intended for transfer learning. Therefore, authors intentionally ignored any approaches that do not benefit downstream tasks like neural cache or dynamic evaluation. The code[8] with model[9] are available online.

Second best model was submitted by a team from Jagiellonian University. The system utilizes Kneser-Ney 5-gram language model, where discounted n-gram probability estimates are interpolated with lower order estimates. The system was trained using SRILM toolkit [10]. The code[11] with model[12] are available online.

Third best solution was provided by Krzysztof Wołk, who also provided n-gram statistical model. It was created using PocoLM toolkit [13]. The author provided nine different solutions, every with different method, in the table 2 there are only three best. The model of solution is available online[14].

The results are presented in Table 2.

[7]http://nlp.fast.ai

[8]https://github.com/n-waves/poleval2018

[9]https://go.n-waves.com/poleval2018-modelv1

[10]http://www.speech.sri.com/projects/srilm/

[11]https://github.com/kwrobel-nlp/lm/

[12]http://wierzba.wzks.uj.edu.pl/~kwrobel/LM/lm_o5.gz

[13]https://github.com/danpovey/pocolm

[14]https://drive.google.com/drive/folders/1Bvyoff_8D88pYVmoUjBq_-ocAggksp3F?usp=sharing

7. Summary

It is noticeable that most of the submissions were based on statistical methods. The winning model was based on recurrent neural network architecture and it seems that this approach is the most promising. Moreover it is possible that the language model would work better for smaller vocabulary — the boundary of 3 might be too few for such a large corpus.

The whole corpus and two best solutions are available online for everyone. We hope, that the corpus that we have gathered will be benchmark in the future and more neural-network-based models will occur.

Acknowledgments

The organizers would like to thank Dariusz Czerski from Institute of Computer Science, Polish Academy of Sciences for providing data from Polish Internet forums.

References

Banasiak D., Mierzwa J. and Sterna A. (2017). *Extended N-gram Model for Analysis of Polish Texts*. [in:] *International Conference on Man–Machine Interactions*, pp. 355–364. Springer.

Bengio Y., Ducharme R., Vincent P. and Jauvin C. (2003). *A Neural Probabilistic Language Model*. „Journal of machine learning research", 3(Feb), pp. 1137–1155.

Chelba C., Mikolov T., Schuster M., Ge Q., Brants T., Koehn P. and Robinson T. (2013). *One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling*. „arXiv:1312.3005".

Howard J. and Ruder S. (2018). *Universal Language Model Fine-tuning for Text Classification*. [in:] *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, pp. 328–339.

Kłosowski P. (2017). *Polish Language Modelling for Speech Recognition Application*. [in:] *Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA 2017)*, pp. 313–318. IEEE.

Kudo T. (2018). *Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates*. „arXiv:1804.10959".

Marcus M. P., Marcinkiewicz M. A. and Santorini B. (1993). *Building a Large Annotated Corpus of English: The Penn Treebank*. „Computational linguistics", 19(2), pp. 313–330.

Mikolov T., Karafiát M., Burget L., Černocký J. and Khudanpur S. (2010). *Recurrent Neural Network Based Language Model*. [in:] *11th Annual Conference of the International Speech Communication Association*.

Ogrodniczuk M. (2012). *The Polish Sejm Corpus* [in:] *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC 2012)*, pp. 2219–2223.

Przepiórkowski A., Banko M., Górski R. L. and Lewandowska-Tomaszczyk B. (2012). *Narodowy Korpus Języka Polskiego [En.: National Corpus of Polish]*. „Wydawnictwo Naukowe PWN, Warsaw".

Rapp B. (2008). *N-gram Language Models for Polish Language. Basic Concepts and Applications in Automatic Speech Recognition Systems*. [in:] *Computer Science and Information Technology, 2008. IMCSIT 2008. International Multiconference on*, pp. 321–324. IEEE.

Wołk K., Wołk A. and Marasek K. (2017). *Big Data Language Model of Contemporary Polish*. [in:] *Computer Science and Information Systems (FedCSIS), 2017 Federated Conference on*, pp. 389–395. IEEE.

Ziółko B. and Skurzok D. (2011). *N-grams Model for Polish*. „Speech and language technologies", pp. 107–127.

Universal Language Model Fine-Tuning with Subword Tokenization for Polish

Piotr Czapla (n-waves, Wrocław), **Jeremy Howard** (fast.ai, University of San Francisco), **Marcin Kardas** (n-waves, Wrocław)

Abstract

Universal Language Model for Fine-tuning (ULMFiT; Howard and Ruder 2018) is one of the first NLP methods for efficient inductive transfer learning. Unsupervised pretraining results in improvements on many NLP tasks for English. In this paper, we describe a new method that uses subword tokenization to adapt ULMFiT to languages with high inflection. Our approach results in a new state-of-the-art for the Polish language, taking first place in Task 3 of PolEval'18. After further training, our final model outperformed the second best model by 35%. We have open-sourced our pretrained models and code.[1]

1. Introduction

Language Modeling recently gained in importance as it is being used as a base for transfer learning in multiple supervised tasks, obtaining impressive improvements over state-of-the-art (Howard and Ruder 2018, Peters et al. 2018, Radford et al. 2018). For example the error in text classification tasks was reduced by 18% – 24% (Howard and Ruder 2018). More complex tasks like commonsense reasoning and question answering were significantly improved by applying transfer learning from a Language Model (Radford et al. 2018). Use of unsupervised learning and transfer learning has the additional benefits of greatly reduced computing time and data requirements for downstream supervised tasks. In some cases data requirements were reduced by 100 times (Howard and Ruder 2018).

[1] `https://n-waves.com/poleval2018`, `http://nlp.fast.ai`

Use of transfer learning is even more important for languages such as Polish, where access to large supervised data sets is very limited. Most of the language models published to date are n-gram models, that do not allow for transfer learning and are very memory hungry.

2. Our Contribution

We adapt Universal Language Model Fine-Tuning (ULMFiT; Howard and Ruder 2018) to handle Polish inflection with subword tokenization using SentencePiece (Kudo 2018). We trained multiple models on the PolEval 2018 LM dataset. Our best model achieved a perplexity of 117.7 on the test set, resulting in first place in the competition (second place scored a perplexity of 146.7). With further tuning after the competition of the model's hyperparameters, we lowered the perplexity to 95.0.

We hope to see the use of FastText (Bojanowski et al. 2017) as the most common way of representing text in Polish replaced with our combination of SentencePiece and ULMFiT.

3. Related Work

Language models traditionally were approximated with non-parametric models based on counting statistics. This were recently replaced with deep neural network for popular languages like English. However most of the literature devoted to the Polish language considers n-gram models (Ziółko and Skurzok 2011, Wołk et al. 2017, Smywiński-Pohl and Ziółko 2016, Pohl and Ziółko 2013). Brocki et al. (2012) showed that a simple neural network (5 context words with 50 dimensional embeddings and one hidden layer) greatly outperforms a 4-gram solution on a Polish corpus. Regardless of performance, the n-gram models tend to be large (several dozens gigabytes for 5-gram; Wołk et al. 2017), making their use in web or mobile applications infeasible. For comparison, our best performing model is around 150 MB without compression. Moreover non-parametric models do not allow for transfer learning, which is the key to good performance on many NLP tasks.

Natural language processing tasks show the best performance when transfer learning is applied either from an LSTM language model (Howard and Ruder 2018, Peters et al. 2018) or from self-attention language models (Radford et al. 2018).

The latter may hold the most promise as has been shown to work well on advanced NLP tasks like question answering, however, it is hard to train and requires extensive computing power and time (Al-Rfou et al. 2018). Therefore, we decided to first adopt an LSTM based model for Polish.

LSTMs are the most widely used RNNs. Recent state of the art performance of language models can be tracked to Merity et al. (2018), who propose a way to efficiently use dropout in LSTM networks as well as other regularization and performance techniques like averaged stochastic gradient descent, or randomized-length backpropagation through time (BPTT). This work was later extended to transfer learning and classification by Howard and Ruder (2018). Transfer learning in language modeling was shown to benefit from slanted triangular learning rates and other techniques described by Smith (2017), originally used to quickly train computer vision models with minimal resources.

LSTM based language models can be improved with use of adaptive methods during inference: neural cache (Grave et al. 2016) and dynamic evaluation (Krause et al. 2017). Both methods depend on observing sequence elements after they are predicted in order to perform adaptation. As our Polish language model is intended for transfer learning and not just the language modeling, we intentionally ignored any approaches that do not benefit downstream tasks.

A few papers investigate using some more sophisticated activation functions for the output layer, e.g., mixture of softmaxes (Yang et al. 2017) and Hebbian softmax (Rae et al. 2018). The use of mixture of softmaxes has been criticized for large computing and memory requirements. Whilst Hebbian softmax is a new work that holds a promise for a better language model for downstream tasks, it requires significant computing power. Their models where trained for 6 days with 8 P100s, while ULMFiT can be trained in around 6 to 10 hours on one P100.

ULMFiT's approach (Howard and Ruder 2018) contributes a number of training tactics that allow for inexpensive training of language models. It introduced a successful approach to transfer learning and fine-tuning for NLP tasks. We selected it as our base for practical reasons such as small memory footprint, quick training time and the direct applicability to other downstream tasks like sentiment analysis.

A popular approach to transfer learning explored earlier in NLP was word embeddings. They appear in the Polish NLP space in form of word2vec (Rogalski and Szczepaniak 2016, Mykowiecka et al. 2017) and FastText (Bojanowski et al. 2017). However this approach only pretrains the first layer of a model, which greatly limits its effectiveness.

All of the word embeddings before FastText were hindered by the inflection of the Polish language, which renders most approaches to finding embeddings for full words incapable of learning useful features. The most successful attempt was FastText, which uses pieces of words.

Another approach to address inflections in Polish is to use byte pair encoding (Sennrich et al. 2016), character level language models (Peters et al. 2018) or unigram subword tokenization (Kudo 2018). We used the unigram algorithm as its representation of Polish words most closely fitted the training pipeline of ULMFiT, and because it has shown state of the art performance in downstream tasks such as machine translation.

4. Model

4.1. Dataset

Our language model was trained only on PolEval 2018 LM data[2]. A summary of the datasets is presented in Table 1.

Table 1: Summary of PolEval 2018 LM datasets. The tokens denoting beginning and end of sentence are not included

Dataset	Sentences	Tokens	OOV rate
train	23.0 M	451.8 M	0.73%
train (dedup.)	21.3 M	423.9 M	0.78%
test	2.6 M	50.2 M	0.91%
test (dedup.)	2.4 M	48.6 M	0.94%

The vocabulary is created from all tokens appearing at least 3 times in the training data, yielding a vocabulary of 1.38 M tokens.

4.2. Subword Tokenization

Similarly as in the requirements of the competition, ULMFiT represents tokens using a fixed-size vocabulary. Tokens not in the vocabulary are replaced by the special `<unk>` token. However, by mapping tokens to integer identifiers we get rid of information regarding words structure. As a result, a language model operating on full words needs much more data to learn rules of highly inflected languages like Polish.

One of solutions to this problem is to use a character level model (Krause et al. 2017, Al-Rfou et al. 2018). Compared to word-based models, character level models are larger and require higher computational costs to achieve the same performance (Bojanowski et al. 2015). To gain the advantages of both approaches we trained a model working

[2] `https://n-waves.com/poleval2018/competition` — the url to the competition will most likely change in 2019 so here is an up to date redirection.

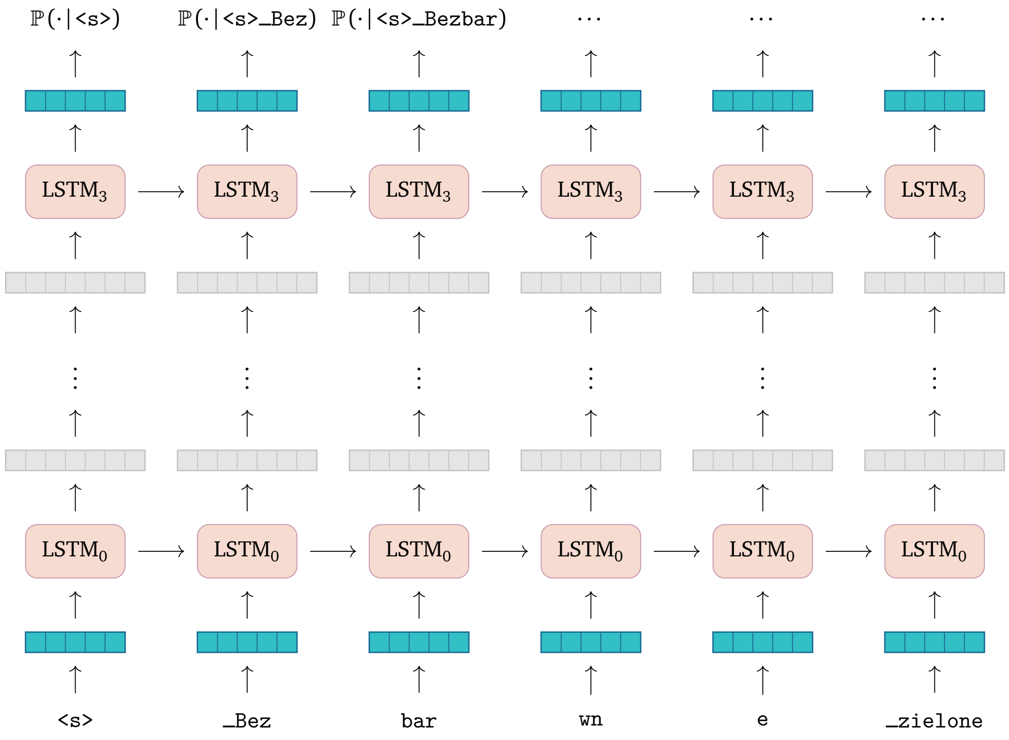


Figure 1: An ULMFiT architecture with 4 recurrent layers

on parts of words. The subword vocabulary is created by training a SentencePiece[3] tokenization model. We use a unigram segmentation algorithm (Kudo 2018). Table 2 shows an example of subword tokenization of a sentence for various vocabulary sizes. An important property of SentencePiece tokenization, necessary for us to obtain a valid word-based language model, is its reversibility. We do not use subword regularization as we decided that the available training dataset is large enough to avoid overfitting.

We now present a formal justification of our approach. For a multiset of sentences $S = \{s_1, \ldots, s_N\}$ and LM $q\colon W^* \to [0,1]$, the empirical perplexity per token is given by

$$\mathrm{ppl}_S(q) \overset{def}{=} 2^{H(\tilde{p},q)/\mathbb{E}_{s\sim\tilde{p}}(|s|_W)}\,,$$

where

$$H(\tilde{p},q) = -\frac{1}{N}\sum_{s\in S} \lg q(s)$$

[3]https://github.com/google/sentencepiece

Table 2: An example split of sentence “Bezbarwne zielone idee wściekle śpią.” (Colorless green ideas sleep furiously) into subword tokens using SentencePiece models differing by vocabulary sizes. Ratio denotes an average number of subword tokens used to encode an input token. The bottom part of the table was obtained by applying a lowercasing preprocessing step (see Section 4.3).

$\lvert V\rvert$	**Ratio**	
$4k$	1.90	`▁B e z bar w ne ▁zielon e ▁i de e ▁w ści ek le ▁ ś pi ą ▁.`
$8k$	1.67	`▁Bez bar w ne ▁zielon e ▁ide e ▁w ści ek le ▁ ś pi ą ▁.`
$25k$	1.42	`▁Bez bar wn e ▁zielone ▁ide e ▁w ście kle ▁śpi ą ▁.`
$50k$	1.34	`▁Bez bar wne ▁zielone ▁idee ▁w ście kle ▁śpi ą ▁.`
$100k$	1.29	`▁Bez bar wne ▁zielone ▁idee ▁w ście kle ▁śpią ▁.`
$4k$	2.04	`<up> ▁bez bar w ne ▁ zielon e ▁i de e ▁w ści ek le ▁ ś pi ą ▁.`
$8k$	1.83	`<up> ▁bez bar w ne ▁zielone ▁ide e ▁w ści ek le ▁ ś pi ą ▁.`
$25k$	1.61	`<up> ▁bezbarwn e ▁zielone ▁ide e ▁w ście kle ▁śpi ą ▁.`
$50k$	1.53	`<up> ▁bezbarwn e ▁zielone ▁idee ▁w ście kle ▁śpi ą ▁.`
$100k$	1.49	`<up> ▁bezbarwne ▁zielone ▁idee ▁w ście kle ▁śpią ▁.`

is an empirical cross-entropy and

$$\mathbb{E}_{s\sim\tilde{p}}\left(|s|_W\right) = \frac{1}{N}\sum_{s\in S}|s|_W$$

is the average sentence length (in tokens).

Let $F\colon W^* \xrightarrow{1:1} V^*$ be a one-to-one mapping from sentences/sequences over tokens in W into sequences over tokens in V. Having a LM $q_V\colon V^* \to [0,1]$ we can create a LM $q_W\colon W^* \to [0,1]$ with $q_W(s) = q_V(F(s))$. F being injective guarantees that $Z \overset{def}{=} \sum_{w\in W^*} q_W(s) \le 1$. To make q_W a valid distribution we could normalize it by Z (computing of which could be infeasible) or simply assume that $q_W(\#) = 1 - Z$ for some additional symbol $\# \notin W$ (and 0 for any other sequence containing $\#$). With $(F\circ q_V)(s) \overset{def}{=} q_V(F(s))$ we have

$$\begin{aligned}
\lg\left(\mathrm{ppl}_S\left(q_W\right)\right) = \lg\left(\mathrm{ppl}_S\left(F\circ q_V\right)\right) &= \frac{H\left(\tilde{p}, F\circ q_V\right)}{\mathbb{E}_{s\sim\tilde{p}}\left(|s|_W\right)} \\
&= \frac{H\left(\tilde{p}, F\circ q_V\right)}{\mathbb{E}_{s\sim\tilde{p}}\left(|F(s)|_V\right)}\cdot\frac{\mathbb{E}_{s\sim\tilde{p}}\left(|F(s)|_V\right)}{\mathbb{E}_{s\sim\tilde{p}}\left(|s|_W\right)} \\
&= \lg\left(\mathrm{ppl}_{F(S)}\left(q_V\right)\right)\cdot\frac{\mathbb{E}_{s\sim\tilde{p}}\left(|F(s)|_V\right)}{\mathbb{E}_{s\sim\tilde{p}}\left(|s|_W\right)}
\end{aligned}$$

or equivalently

$$\mathrm{ppl}_S\left(q_W\right) = \left(\mathrm{ppl}_{F(S)}\left(q_V\right)\right)^{\mathbb{E}_{s\sim\tilde{p}}\left(|F(s)|_V\right)/\mathbb{E}_{s\sim\tilde{p}}\left(|s|_W\right)}\,. \tag{14.1}$$

In our case, W consists of 3 control tokens (`<unk>`, `<s>` and `</s>`) and 1 378 027 tokens[4] occurring 3 or more times in the training data. V is constructed by unigram model (Kudo 2018) using SentencePiece subword tokenizer and consists of 4 control tokens (additional `<pad>` token) and 24 996 subword tokens. For any sentence $s \in W^*$ we use the most probable tokenization as $F(s)$. To get even better results we could sum over all possible splits of s. We believe, however, that the normalization factor Z can be neglected as model should learn to ignore non-existent words or alternative tokenizations.

4.3. Universal Language Model Fine-tuning

Our model is based on the fast.ai[5] implementation of ULMFiT. Table 3 gives details of our final submission as well as the best model trained after the competition.

Table 3: Details of our submission and the best model trained after competition

	PolEval submission	**tuned model**
vocabulary size	50K	25 K
RNN type	LSTM	LSTM
recurrent layers	4	4
embeddings dimension	400	400
hidden state dimension	1150	1150
training time	18 epochs	30 epochs
data set used for training	≈25%	100%
batch size	192	128
sampled softmax	15 K samples	no
text transforms	none	none
perplexity	117.8	95.0

4.4. Data Preprocessing

Our preprocessing pipeline for the training data starts with counting occurrences of word tokens and extracting a dictionary consisting of words with at least 3 occurrences. The tokenized file is then deduplicated.

During development we experimented with an optional step of encoding words with an initial letter being the only capital letter. Such words are preceded with a special

[4]Even though not all tokens in PolEval datasets are words (e.g., there are tokens consisting of punctuation marks) and some tokens produced by SentencePiece are valid words, for simplicity we call the former *word tokens* and the later *subword tokens*.

[5]`http://nlp.fast.ai/`

<up> token and the initial letter is lower-cased (see Table 2 for an example). However, the experiments showed that there is no significant difference.

After the deduplication (and optional lower-casing) the full dataset is used to train a SentencePiece unigram model. The dictionary extracted in the first step is used to remove rare (i.e., out-of-vocabulary) word tokens. The resulting sentences are encoded by the SentencePiece model. Due to large size of the training dataset we do not use subword regularization – each sentence is tokenized only once with the best encoding. The final dataset is randomly shuffled and split into a validation dataset (around 10 million subword tokens) and a training dataset.

For the test dataset we optionally perform a lower-casing step, remove the out-of-vocabulary words and encode word tokens into subword tokens with SentencePiece model. The deduplication step ensures that training and validation sets are disjoint. However, because the test and the training datasets share some sentences (around 0.23 M / 9.29% test sentences are present in the training dataset), the cross validation perplexity was always higher than the test one.

5. Experiments

We run multiple experiments on around 10 M subword tokens of data to gain an intuition on how to tune ULMFiT hyperparameters for best performance on the Polish language. Most promising solutions were trained further on the whole training set, and the best (based on validation perplexity) was selected. In this Section we present our findings regarding tuning various hyperparameters of the ULMFiT model.

5.1. Vocabulary Size

Our experiments showed that out of all tested hyperparameters, the vocabulary size has the greatest impact on model performance on Polish language. Unlike the English ULMFiT on full words, our vocabulary size influences how the subword tokens are formed. For a large enough vocabulary two words with the same lemma are represented as two different ids, and the similarity information is lost. The smaller the vocabulary, the closer we get to character level models.

5.2. Number of Recurrent Layers

We tested our models with 3, 4 and 5 recurrent layers. Each additional layer noticeably increases memory usage of model and time necessary for a single training epoch. On a small training dataset the 5-layer models performed significantly worse. We do not know whether longer training, more data or subword regularization could improve the performance relative to smaller models. The performance of 3-layer and 4-layer models were almost identical on a small dataset, but training on the full dataset proved that the latter is more capable, and achieves lower validation perplexity.

5.3. Text Preprocessing

In some experiments we applied lower-casing of the initial letter of each word. To make the transform reversible, such words were preceded by <up> control token (see Table 2). For most of the tested vocabulary sizes and number of layers there was no noticeable difference in perplexity, with an exception of 100 K tokens, where lower-casing resulted in degraded performance.

5.4. Results

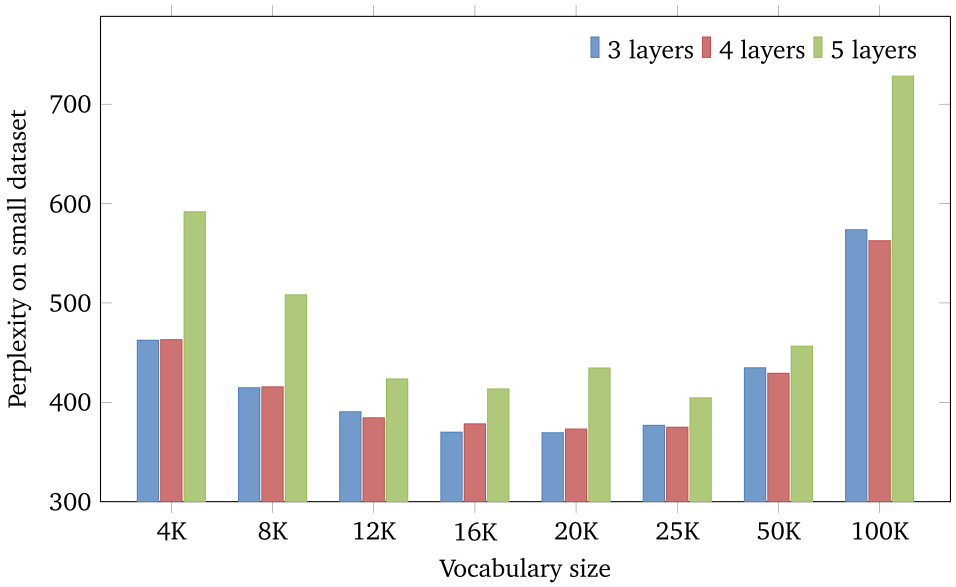


Figure 2: Plot showing an impact of number of recurrent layers and vocabulary size on validation perplexity. Models trained for 12 epochs on a small dataset consisting of around 10 M subword tokens. Models with vocabulary of size 25K and more were trained with sampled softmax (with 15K samples).

6. Final Remarks

We showed that a subword tokenization can be used to achieve a high-performing language model for Polish, a morphologically rich language. The presented model achieves state-of-the-art perplexity. However, we did not use the main advantage of ULMFiT, i.e., its ability for transfer learning. The natural next steps are to implement custom heads for common NLP tasks (named entity recognition, sentiment analysis) with a pretrained ULMFiT model as a backbone.

References

Al-Rfou R., Choe D., Constant N., Guo M. and Jones L. (2018). *Character-level Language Modeling with Deeper Self-Attention*. „arXiv:1808.04444".

Bojanowski P., Joulin A. and Mikolov T. (2015). *Alternative Structures for Character-Level RNNS*. „CoRR", abs/1511.06303.

Bojanowski P., Grave E., Joulin A. and Mikolov T. (2017). *Enriching Word Vectors with Subword Information*. „Transactions of the Association for Computational Linguistics", 5, pp. 135–146.

Brocki Ł., Marasek K. and Koržinek D. (2012). *Connectionist Language Model for Polish*. [in:] *Intelligent Tools for Building a Scientific Information Platform*, pp. 243–250. Springer.

Grave E., Joulin A. and Usunier N. (2016). *Improving Neural Language Models with a Continuous Cache*. „arXiv:1612.04426".

Howard J. and Ruder S. (2018). *Universal Language Model Fine-tuning for Text Classification*. [in:] *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 328–339. Association for Computational Linguistics.

Krause B., Kahembwe E., Murray I. and Renals S. (2017). *Dynamic Evaluation of Neural Sequence Models*. „arXiv:1709.07432".

Kudo T. (2018). *Subword Regularization: Improving Neural Network Translation models with multiple subword candidates*. [in:] *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 66–75. Association for Computational Linguistics.

Merity S., Keskar N. S. and Socher R. (2018). *Regularizing and Optimizing LSTM Language Models*. [in:] *International Conference on Learning Representations*.

Mykowiecka A., Marciniak M. and Rychlik P. (2017). *Testing Word Embeddings for Polish*. „Cognitive Studies", (17).

Peters M. E., Neumann M., Iyyer M., Gardner M., Clark C., Lee K. and Zettlemoyer L. (2018). *Deep Contextualized Word Representations*. [in:] *Proceedings of NAACL*.

Pohl A. and Ziółko B. (2013). *Using Part of Speech N-grams for Improving Automatic Speech Recognition of Polish*. [in:] Perner P. (ed.), *Machine Learning and Data Mining in Pattern Recognition*, pp. 492–504, Berlin, Heidelberg. Springer Berlin Heidelberg.

Radford A., Narasimhan K., Salimans T. and Sutskever I. (2018). *Improving Language Understanding by Generative Pre-training*.

Rae J. W., Dyer C., Dayan P. and Lillicrap T. P. (2018). *Fast Parametric Learning with Activation Memorization*. „CoRR", abs/1803.10049.

Rogalski M. and Szczepaniak P. S. (2016). *Word Embeddings for the Polish Language*. [in:] *International Conference on Artificial Intelligence and Soft Computing*, pp. 126–135. Springer.

Sennrich R., Haddow B. and Birch A. (2016). *Neural Machine Translation of Rare Words with Subword Units*. [in:] *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016), Volume 1: Long Papers*.

Smith L. N. (2017). *Cyclical Learning Rates for Training Neural Networks*. [in:] *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 464–472. IEEE.

Smywiński-Pohl A. and Ziółko B. (2016). *Application of Morphosyntactic and Class-based Language Models in Automatic Speech Recognition of Polish*. „International Journal on Artificial Intelligence Tools", 25(02).

Wołk K., Wołk A. and Marasek K. (2017). *Big Data Language Model of Contemporary Polish*. [in:] *Federated Conference on Computer Science and Information Systems (FedCSIS 2017)*, pp. 389–395. IEEE.

Yang Z., Dai Z., Salakhutdinov R. and Cohen W. W. (2017). *Breaking the Softmax Bottleneck: A High-rank RNN Language Model*. „arXiv:1711.03953".

Ziółko B. and Skurzok D. (2011). *N-grams Model for Polish*. „Speech and Language Technologies", pp. 107–127.

Survey on Statistical and Semantic Language Modelling Based on PolEval

Krzysztof Wołk (Polish-Japanese Academy of Information Technology)

Abstract

Based on PolEval 2018 provided training data we provide a set of 6-gram language models of contemporary Polish which are based on over 20 million sentences in Polish. We do pre-processing and evaluation similarly, as in Google One Billion Word Benchmark. We survey most common statistical language modelling toolkits and implement semantic language models with two different approaches. We train the language model, evaluate how toolkits perform for the Polish language and finally present advances of perplexity (PPL) values, through the utilization of our models.

Keywords

language modelling, perplexity, Polish language model

1. Introduction

There are a large number of language processing tasks available that make web-scale corpora attractive and needed due in most, to the vast amount of information which exists in different languages. Language modelling is of great significance, where web-scale models for language have demonstrated their ability to enhance automated speech recognition performance and machine translation quality (Brants et al. 2007, Guthrie and Hepple 2010, Chelba and Schalkwyk 2013). There are also other NLP tasks that depend greatly on language modelling e.g. language quantification (Lenko-Szymanska 2016).

In this article we focus on statistical toolkits and semantic language models. A statistical language model is a probability distribution over sequences of words. Given a set of sentences in Polish in a random order, each with original punctuation. We did this work within the PolEval 2018 task. The goal of the task was to create a language model for Polish. Participants were allowed to use any approach to creating the model. But the data was limited.

As the training data set, the PolEval team has prepared a corpus containing over 20 million sentences in Polish. The corpus consists of a single plain text. The order of the sentences was randomized, and their sources were revealed together with the test data. The punctuation and spelling has been left untouched from the original sources. The participants were provided the segmented training corpus. All the tokens (words and punctuation marks) were separated with space.

Data pre-processing was performed similarly, as in Google One Billion Word Benchmark. Words were assumed to be the basic units of the language model. Punctuation marks are treated as words and they were a part of language model. A vocabulary was extracted from the training set. Every word and punctuation mark that has occurred three or more times in the training set was included in the vocabulary. Additionally, the vocabulary included sentence boundary markers ("–") and the token which was used to map Out Of Vocabulary (OOV) words <unk>. OOV words are tokens that are present in the training or test set, but not included in the vocabulary.

2. Differences between Polish and English Languages

The main goal of this survey was to check how mainstream language modelling toolkits perform for Polish language without any special adaptation even that those tools were invented for English. Polish ang English differ greatly. In general, Polish and English differ in syntax and grammar. English is a positional language, which means that the syntactic order (the order of words in a sentence) plays a very important role, particularly due to the limited inflection of words (e.g., lack of declension endings). Sometimes, the position of a word in a sentence is the only indicator of the sentence's meaning. In a Polish sentence, a thought can be expressed using several different word orderings, which is not possible in English. For example, the sentence "I bought myself a new car." can be written in Polish as "Kupiłem sobie nowy samochód.", or "Nowy samochód sobie kupiłem.", or "Sobie kupiłem nowy samochód.", or "Samochód nowy sobie kupiłem.". The only exception is when the subject and the object are in the same clause and the context is the only indication which is the object and which is subject. For example, "Mysz liże kość. (A mouse is licking a bone.)" and "Kość liże mysz. (A bone is licking a mouse).".

Differences in potential sentence word order make the translation process more complex, especially when using a phrase-model with no additional lexical information

(Swan 2003). In addition, in Polish it is not necessary to use the operator, because the Polish form of a verb always contains information about the subject of a sentence. For example, the sentence “On jutro jedzie na wakacje.” is equivalent to the Polish “Jutro jedzie na wakacje.” and would be translated as “He is going on vacation tomorrow.” (Choong and Power 2003).

In the Polish language, the plural formation is not made by adding the letter “s” as a suffix to a word, but rather each word has its own plural variant (e.g., “pies — psy”, “artysta — artyści”, etc.). Additionally, prefixes before nouns like “a”, “an”, “the”, do not exist in Polish (e.g., “a cat — kot”, “an apple — jabłko”, etc.) (Swan 2003).

The Polish language has only three tenses (present, past, and future). However, it must be noted that the only indication whether an action has ended is an aspect. For example, “Robiłem pranie.” Would be translated as “I have been doing laundry”, but “Zrobiłem pranie.” as “I have done laundry”, or “płakać — wypłakać” as “cry — cry out” (Swan 2003).

The gender of a noun in English does not have any effect on the form of a verb, but it does in Polish. For example, “Zrobił to. — He has done it.”, “Zrobiła to. — She has done it.”, “lekarz/lekarka — doctor”, “uczeń/uczennica = student”, etc. (Cao et al. 2005).

Because of this complexity, progress in the development of SMT systems for West-Slavic languages has been substantially slower than for other languages.

3. Toolkits Used in the Research

For language model training we firstly used the most common SRILM toolkit (Stolcke 2002). The fundamental challenge that language models handle is sparse data. It is possible that some possible translations were not present in the training data but occur in real life. There are some methods in SRILM, such as add-one smoothing, deleted estimation, and Good-Turing smoothing, that cope with this problem (Koehn 2010).

Interpolation and back-off are other methods of solving the sparse data problem in n-gram LMs. Interpolation is defined as a combination of various n-gram models with different orders. Back-off is responsible for choosing the highest-order n-gram model for predicted words from its history. It can also restore lower-order n-gram models that have shorter histories. There are many methods that determine the back-off costs and adapt n-gram models. The most popular method is known as KneserNey smoothing. It analyses the diversity of predicted words and takes their histories into account (Koehn et al. 2007). We used this smoothing method and trained 6-gram language models.

Other used tool was KenLM that estimates, filters, and queries language models. Estimation is fast and scalable due to streaming algorithms. The KenLM is a library that implements two data structures for efficient language model queries, reducing both time and memory costs. The probing data structure uses linear probing hash tables and is designed for speed. Compared with the widely used SRILM, the KenLM probing model is 2.4 times as fast while using 57
citep of the memory. The trie data structure is a trie with bit-level packing, sorted records, interpolation search, and optional quantization aimed at lower memory consumption. Trie simultaneously uses less memory than the smallest lossless baseline and less CPU than the fastest baseline. The toolkit code is open-source, thread-safe, and integrated into the Moses, cdec, and Joshua translation systems.

Lastly, we put our attention on PocoLM toolkit. The new PocoLM toolkit is designed to provide better results in modelling word sequences for use in speech recognition than standard toolkits like SRILM and KenLM and offers a good training time.

Currently, most popular language modelling tools first have to estimate partial language models separately in order to conduct interpolation and finally interpolate those estimates. The PocoLM authors state that this does not seem to be the optimal solution because it interacts with proper backoff propagation. In PocoLM, the data sources are interpolated before the language model estimation at the level of data counts.

The PocoLM authors also try to improve the most popular Stolcke entropy pruning method. PocoLM method operates on the same principle as Stolcke pruning but manages to be more optimal because, when it removes a probability from the LM, it assigns the removed data to the backed-off propagation and updates its probabilities accordingly.

4. Semantically-enhanced Language Models

Within the research we also implemented tool that extends the monolingual corpora with semantical information. We used two methods for this task. Firstly, the artificially generated monolingual corpus was obtained using statistical models, which are based purely on how frequently “things” happen, and not on what they really mean. This means that they do not really understand what was augmented. In the first method the data was additionally extended with semantic information from the plWordNet, so as to improve the quality and scope of the data text domain. The word relationships were integrated into generated data using the plWordNet database.

The way in which WordNet was used to obtain a probability estimator was shown by Cao et al. (2005). In particular, we wanted to obtain $P(w_i|w)$, where w_i and w are assumed to have a relationship in WordNet. The formula is as follows:

$$P(w_i|w) = \frac{c(w_i, w|W, L)}{\sum_{w_j} c\left(w_j, w|W, L\right)}$$

where W is a window size and $c(w_i, w|W, L)$ is the count of w_i and w appearing together within W-window. This can be obtained simply by counting each within a certain corpus. In order to smooth the model, we applied interpolated Kneser-Ney (Chen and Goodman 1999) smoothing strategies.

The following relationships were considered: synonym, hypernym, hyponym, and hierarchical distance between words.

Another common approach to semantic analysis that is also used within this research is latent semantic analysis (LSA). LSA has already been shown to be very helpful in automatic speech recognition (Bellegarda 2000) and many other applications, which was the reason for incorporating it within the scope of this research. The high-level idea of LSA is to convert words into concept representations and to assume that if the occurrence of word patterns in documents is similar, then the words are also similar. The mathematical model can be defined as follows.

In order to build the LSA model, a co-occurrence matrix W will first be built, where w_{ij} is a weighted count of word w_j and document d_j.

$$w_{ij} = G_i L_{ij} C_{ij}$$

where C_{ij} is the count of w_i in document d_j; L_{ij} is local weight and G_i is global weight. Usually, L_{ij} and G_i can use TF/IDF.

Then, singular value decomposition (SVD) analysis will be applied to W, as

$$W = USV^T$$

where W is a $M * N$ matrix (M is vocabulary size, N is document size); U is $M * R$, S is $R * R$, and V is a $R * N$ matrix. R is usually a predefined dimension number between 100 and 500.

After that, each word w_i can be denoted as a new vector $U_i = u_i$. Based on this new vector, the distance between two words is defined as:

$$K\left(U_i, U_j\right) = \frac{u_{i\,m}^{2T}}{|u_i| * |u_m|}$$

Therefore, clustering can be performed to organize words into K clusters, $C_1, C_2, \ldots, C_K$.

If H_{q-1} is the history for word W_q, then it is possible to obtain the probability of W_q given H_{q-1} using the following formula:

$$\begin{aligned} P\left(W_q|H_{q-1}\right) &= P\left(W_q|W_{q-1}, W_{q-2}, \ldots W_{q-n+1}, d_{q_1}\right) \\ &= P\left(W_q|W_{q-1}, W_{q-2}, \ldots W_{q-n+1}\right)\left(W_q|d_{q_1}|\right) \end{aligned}$$

where $P\left(W_q|W_{q-1}, W_{q-2}, \ldots W_{q-n+1}\right)$ is the N-gram model; $P\left(W_q|d_{q_1}|\right)$ is the LSA model.

Additionally,

$$P\left(W_q|d_{q_1}\right) = P\left(U_q|V_q\right) = \frac{K\left(U_q, V_{q_1}\right)}{Z\left(U, V\right) K\left(U_q, V_{q_1}\right)} = \frac{U_{q\,q-1}^{\;T}}{|U_q^{\frac{1}{2}}| * |V_{q-1}^{1/2}|}$$

where $Z(U, V)$ is the normalized factor.

It is possible to also apply word smoothing to the model-based K-Clustering as follows:

$$P\left(W_q|d_{q_1}\right) = \sum_{k=1}^{K} P\left(W_q|C_k\right) PC_k|d_{q_1}$$

where $P\left(W_q|C_k\right) PC_k|d_{q_1}$ can be computed using the distance measurement given above by a normalized factor.

In this way, the N-gram and LSA model are combined into a single language model and can be used for word comparison and text generation. The Python code for such LSA analysis was implemented in Thomo's (2009) research.

5. Evaluation and Results

The evaluation was done using the perplexity value (PPL) in accordance with the Google One Billion Word Benchmark. The test set was provided by the PolEval 2018 organizers. The model with the lowest perplexity has the best quality.

Perplexity is a measure commonly used to estimate the quality of a language model it is the inverse probability of the test set, normalized by the number of words. The language model should estimate the possibility of occurrence for every word in every sentence. The probability should be calculated for every word and the sentence ending in the test set.

Perplexity can be calculated using below equations:

$$exp\left(\frac{\sum_{i=1}^{N} log_e\left(\frac{1}{q(x_i)}\right)}{N}\right) or \sqrt[N]{\prod_{i=1}^{N}\frac{1}{q(x_i)}}$$

where

— N — number of samples in the test set,

— X — discrete random variable with possible values $x_1, x_2, \ldots, x_n$,

— $q(X)$ — probability function.

For instance, for the bigram language model perplexity equals:

$$PP(W) = \sqrt[N]{\prod_{i=1}^{N}\frac{1}{P(w_1|w_{i-1})}}$$

where

— N — number of words in the testset + number of sentence endings,

— W — the testset {w_1, w_2, ..., w_N},

— $P(w_1|w_{i-1})$ is the conditional probability of word w_i that occurs after word w_{i-1}.

The Table 1 provides results of the experiments.

6. Discussion and Conclusions

Summing up, we successfully released n-gram counts and language models built using plain textual data as well as semantically augmented corpora which overcome limitations of other smaller, publicly available resources. As anticipated the state-of-the-art language modelling PocoLM toolkit for English also outperforms other tools for Polish. This was proven by the improvements in perplexity value. The usage of

Table 1: Results of the experiments

Toolkit	Settings	Perplexity
KenLM	Order 6	259
PocoLM	Order 6	208
PocoLM	Order 6, semantic	211
SRILM	Order 6, float counts	219
SRILM	Order 6, Kneser-ney, Semantic	1336
SRILM	Order 6, Kneser-ney, Semantic and train	454
SRILM	Order 6, Kneser-ney	289
SRILM	Order 6, Kneser-ney, Interpolated	273
SRILM	Order 6, Witten-bell, Interpolated	216

semantical models reduced the OOV problem but did make any positive impact on the PPL, this is probably due to increased data sparsity.

As far as language models are concerned, we will try interpolating the results of continuous space word representations (eg. LSTM RNN) models and log bilinear language models with our n-gram models (Mikolov et al. 2013). As described in literature (Sundermeyer et al. 2012, Xiong et al. 2016) it should allow us to reduce the PPL and WER even more, while retaining ARPA format compatibility. To be more precise, using such method with feature vectors, from tools such as word2vec, GloVe etc. should allow us to get as much as 5% reduction in WER.

In addition, we plan to deal with OOV words problem by modelling only the most popular words. The less popular ones are going to be replaced by some kind of lexical unit smaller than a single word. To be more precise, we are going to uses syllables and stemming to obtain sub word units. What is more, in order to maintain as much lexical information as possible we will use "++ – –" as division mark. This will allow us to keep information about that how the units should be merged to build a word (Sennrich et al. 2015).

We are also going to try the well-established, for English language, algorithm called BPE. The BPE segments rare words into their more commonly appearing sub-words. The BPE achieves its goal by adapting byte pair encoding (BPE; Gage 1994), a compression algorithm, to the task of word segmentation. BPE allows for the representation of an open vocabulary through a fixed-size vocabulary of variable-length character sequences, making it a very suitable word segmentation strategy for neural network models.

Finally, the data can be augmented with Part-of-Speech tags (POS) or/and grammatical groups. We plan to use this information not only for training of factored language models (Bilmes and Kirchhoff 2003) but also incorporate it in stemming phase. In fact, this will allow us to more correctly distinguish sub-word units (like word core, suffixes and prefixes). Such additional tags will be marked by additional tag “@@” in order to distinguish them from sub-word divider.

References

Bellegarda J. (2000). *Data-driven Semantic Language Modeling*. „Institute for Mathematics and Its Applications Workshop”.

Bilmes J. A. and Kirchhoff K. (2003). *Factored Language Models and Generalized Parallel Backoff*. „Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology”.

Brants T., Popat A. C., Xu P., Och F. J. and Dean J. (2007). *Large Language Models in Machine Translation*. „Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning”.

Cao G., Nie J. and Bai J. (2005). *Integrating Term Relationships into Language Models*. „Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval”.

Chelba C. and Schalkwyk J. (2013). *Empirical Exploration of Language Modeling for the google.com Query Stream as Applied to Mobile Voice Search*. „Mobile Speechand Advanced Natural Language Solution”.

Chen S. and Goodman J. (1999). *An Empirical Study of Smoothing Techniques for Language Modeling*. „Computer Speech & Language”.

Choong C. and Power M. S. (2003). *The Difference between Written and Spoken English*. „Assignment Unit, 1”.

Gage P. (1994). *A New Algorithm for Data Compression*. „The C Users Journal”.

Guthrie D. and Hepple M. (2010). *Storing the Web in Memory: Space-efficient Language Models with Constant Time Retrieval*. „Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing”.

Koehn P. (2010). *Moses, Statistical Machine Translation System. User Manual and Code Guide*. „Monograph”.

Koehn P., Hoang H., Birch A., Callison-Burch C., Federico M., Bertoldi N. and Dyer C. (2007). *Moses: Open Source Toolkit for Statistical Machine Translation*. „Proceedings of the 45th Annual Meeting of the ACL”.

Lenko-Szymanska A. (2016). *A Corpus-based Analysis of the Development of Phraseological Competence in EFL Learners Using the Collgram Profile*. „Paper presented at the 7th Conference of the Formulaic Language Research Network (FLaRN)”.

Mikolov T., Sutskever I., Chen K., Corrado G. S. and Dean J. (2013). *Distributed Representations of Words and Phrases and Their Compositionality*. „Advances in Neural Information Processing Systems”.

Sennrich R., Haddow B. and Birch A. (2015). *Neural Machine Translation of Rare Words with Subword Units*. „arXiv:1508.07909”.

Stolcke A. (2002). *SRILM — an Extensible Language Modeling Toolkit*. „Interspeech Conference”.

Sundermeyer M., Schlüter R. and Ney H. (2012). *LSTM Neural Networks for Language Modeling*. „13th Annual Conference of the International Speech Communication Association”.

Swan O. E. (2003). *Polish Grammar in a Nutshell*. „University of Pittsburgh”.

Thomo A. (2009). *Latent Semantic Analysis (LSA) Tutorial*. „`http://webhome.cs.uvic.ca/~thomo/svd.pdf`”.

Xiong W., Droppo J., Huang X., Seide F., Seltzer M., Stolcke A., Yu D. and Zweig G. (2016). *Achieving Human Parity in Conversational Speech Recognition*. „arXiv:1610.05256”.