Proceedings of the PolEval 2020 Workshop

Maciej Ogrodniczuk, Łukasz Kobyliński (eds.)

Institute of Computer Science, Polish Academy of Sciences
Warszawa, 2020

ISBN 978-83-63159-29-0

Warszawa 2020

Contents

PolEval 2020: Introduction

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

PolEval is an evaluation campaign focused on Natural Language Processing tasks for Polish, intended to promote research on language and speech technologies, create objective evaluation procedures and improve state-of-the-art.

Since the beginnings we are glad to observe a steady growth of interest in our initiative — thank you! We started in 2017 with only 2 tasks which already attracted 20 submissions. In 2018 we have received 24 systems competing in 3 tasks and in 2019 — 34 systems in 6 tasks. 2020 brought 42 submissions in 4 tasks (see Figure 1).

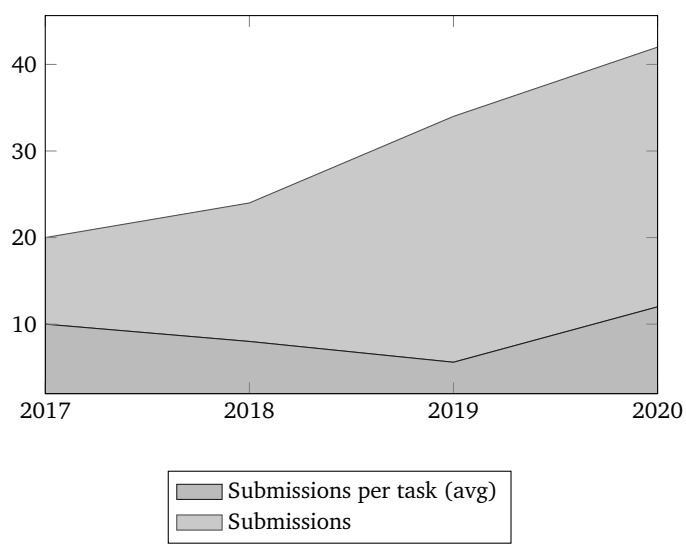


Figure 1: Number of PolEval submissions and average submissions per task in 2017–2020

This volume consists of proceedings of the online workshop session organized during the AI & NLP Day conference[1] on October 26th, 2020, presenting the results of the 2020 edition of the shared task.[2]

In 2020 the systems competed in the following tasks:

— Task 1: Post-editing and rescoring of automatic speech recognition results

— Task 2: Morphosyntactic tagging of Middle, New and Modern Polish

— Task 3: Word Sense Disambiguation

— Task 4: Information extraction and entity typing from long documents with complex layouts

The number of submissions per each task has varied greatly (see Figure 2): this year it was the post-editing and rescoring of ASR results task which has attracted the most submissions.

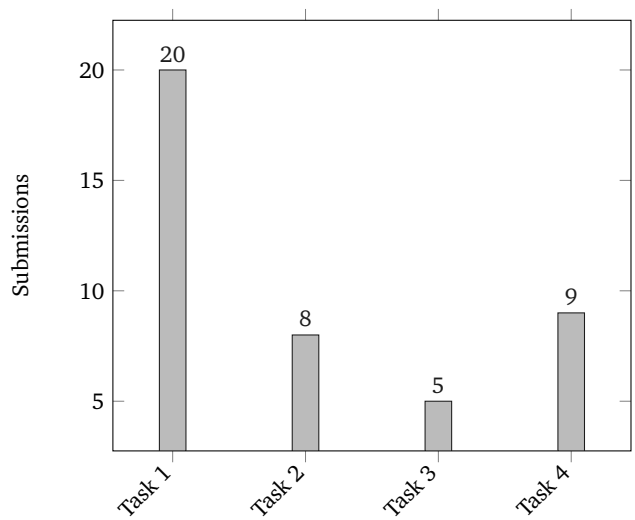


Figure 2: Number of PolEval submissions per task in 2020

We are very sorry that we could not meet in person this time due to the COVID situation but we hope we will be able to do it next year. Please join us to celebrate the fifth edition of PolEval in 2021!

At all times, please feel free to share your ideas for improving this competition or willingness to help in organizing your own NLP tasks.

[1] https://2020.nlpday.pl

[2] http://2020.poleval.pl

Organizers

Concept and administrative issues

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

Task 1: Post-Editing and Rescoring of Automatic Speech Recognition Results

Danijel Koržinek (Polish-Japanese Academy of Information Technology)

Task 2: Morphosyntactic Tagging of Middle, New and Modern Polish

Włodzimierz Gruszczyński (SWPS University of Social Sciences and Humanities)
Witold Kieraś, Marcin Woliński, Dorota Komosińska (Institute of Computer Science, Polish Academy of Sciences)

Task 3: Word Sense Disambiguation

Arkadiusz Janz, Maciej Piasecki (Wrocław University of Science and Technology)

Task 4: Information Extraction and Entity Typing from Long Documents with Complex Layouts

Anna Wróblewska, Filip Graliński, Dawid Lipiński (Applica.ai)
Paweł Rzepiński, Ryszard Szymański, Tymon Czarnota (Data Science students at Warsaw University of Technology, Faculty of Mathematics and Information Science)

Results of the PolEval 2020 Shared Task 1: Post-editing and Rescoring of Automatic Speech Recognition Results

Danijel Koržinek (Polish-Japanese Institute of Information Technology)

Abstract

This paper describes a challenge during the 2020 Poleval competition regarding the topic of post-editing in the context of automatic speech recognition (ASR). The task was to create a system which corrects common language mistakes made by an ASR system. The input to the system could be either a single-best output, an n-best output or a lattice generated by the ASR system. Most contestants used modern NLP and deep learning approaches. The best system achieved an improvement of 13% relative WER.

Keywords

automatic speech recognition, natural language processing, post-editing, rescoring

1. Introduction

Automatic speech recognition (ASR) systems are used to convert audio recordings of speech into text. Just like most machine learning systems, ASR makes mistakes. A common way of expressing this is through the Word Error Rate (WER), which is equivalent to the ratio number of words mistakenly substituted, deleted or inserted with regard to the number of words in the correct transcript of a particular utterance. One of the most often used research goals is the reduction of this value within specific contexts.

This error can be reduced either by improving the internal mechanisms and models of the system itself, but a very popular method used in various problems involving a text output is post-editing (see e.g. Bassil and Alwani 2012, Lee et al. 2019, Lopes et al. 2019, Shterionov et al. 2019, Vu and Haffari 2018). This means using various external techniques to convert the raw text output of the system into a more correct version. A few motivating factors for the post-editing approach are given below.

A typical ASR system is usually created using a combination of an acoustic and a language model, where the language model is often based on a basic statistical n-gram approach. Errors produced by such a system are often trivial and easy to spot. One might argue that using a better language model could fix this problem, but this is often intractable as the system is expected to produce a language score of many hundreds of acoustic-level hypotheses at any given time. Even the more modern end-to-end ASR systems often suffer from this same issue, as the model size is already extremely large in order to contain both the acoustic and language parts in the same deep neural network.

Given the lack of constraints of the typical ASR system, a post-editing solution can take any design approach into account. It can be as simple as using a larger n-gram language model or as complicated as a combination of several NLP-oriented subsystems targeted at fixing specific language problems. It would be especially interesting to explore the new recurrent neural network based language models like BERT or other similar seq2seq approaches.

Finally, it turns out that a form of post-editing is actually a common approach used in many state-of-the-art results (e.g. Xiong et al. 2018, Xu et al. 2018), as a way of getting around the limitations of online speech recognition process. This technique is commonly referred to as re-scoring and it utilized more than just a single best output of the ASR system. It relies either on the N-best outputs (usually $\approx$100 per utterance) or a more efficient data structure known as the lattice. The N-best approach works by simply taking the new model and re-scoring each hypothesis of the N-best hypotheses, which leads to a new ordering and a new sentence ending up as the best result.

The lattice approach is a bit more involved. A lattice is a connected graph where each arc emits a word with a specific weight attached to it. A path starting from a specific state in the graph and ending up in another specified state has a score calculated as the combination of the consecutive weights of the arcs used to traverse it. It is equivalent to the sequence and initial score of the N-best approach discussed above, but much more compact and efficient. Furthermore, each weight can be decomposed into its acoustic and language model components, which makes the re-scoring even better, by saving the acoustic model information in the final score. A lattice oracle is the path within the lattice that is closest to the reference output. A lattice will not contain every possible word sequence, so the lattice oracle error rate is usually $> 0\%$.

2. Task definition

The goal of this task was to create a system for converting a sequence of words from a specific ASR system into another sequence of words that more accurately describes the actual spoken utterance. The training data consisted of utterance pairs – each pair containing an utterance generated by the output of the ASR and an utterance containing the correct transcription of the utterance.

The ASR system used to create the training data was the same one used during the evaluation procedure. That means that the solution to this task had to be able to correct the errors that particular one specific system makes. It didn't need to fix errors in any system that it hasn't seen.

Table 1: Contents of the training data

Data set	WER %	Lattice oracle WER %
Clarin-PL training set	9.59	3.75
Clarin-PL test set	12.08	4.72
Clarin-PL dev set	12.39	4.93
Polish Parliamentary Corpus	45.57	30.71

For simplicity, the output of both the ASR and the required reference was in normalized form: no punctuation, no capitalization and no digits, symbols or abbreviations. Just a simple sequence of words.

3. Data

The ASR system used to generate the transcripts was trained on the Clarin-PL studio corpus using the *tri3b* model from the ClarinStudioKaldi setup available online and in (Koržinek et al. 2017). The same system was applied to a larger Polish Parliamentary Corpus[1] from the last year's competition described in (Koržinek 2019).

Each provided set contained the following files:

— 1-best output – each utterance containing a single best transcript of the ASR output

— n-best output – each utterance containing up to 100 best alternative hypotheses of the ASR output

— lattice output – each utterance containing a list of arcs forming a lattice of the ASR output; each line contains the following fields: start node, end node, words, language weight, acoustic weight, list of phonetic-level states

— reference – file similar to the 1-best output, but containing the actual reference transcript

The contestants were encouraged to use other resources to train their systems such as the Polish Sejm Corpus (Ogrodniczuk 2012, 2018) for language modeling. There was also a similar task completed a year prior by a different research team described by Kubis et al. (2020).

4. Evaluation

The evaluation was carried out on a newly created dataset of sessions of the European Parliament in Polish, unavailable to the participants, but soon to be published online and described by Chmiel et al. (2020). Participants were provided with a file containing a collection of ASR outputs in the format described in the training section. They had to create a

[1] `http://clip.ipipan.waw.pl/PPC`

file containing the final corrected output, one line per utterance in the similar format as the 1-best output. The participants were allowed any number of submissions and the best one was counted to the final ordering.

The evaluation was carried out using the NIST SCLITE package to determine the Word Error Rate (WER) of the individual submissions. Word error rate is defined as the number of edit-distance errors (deletions, substitutions, insertions) divided by the length of the reference:

$$WER = \frac{N_{del} + N_{sub} + N_{ins}}{N_{ref}}$$

The task of creating an interesting competition turned out to be more difficult than anticipated. If the ASR system used was too good, the post-editing problem would be too minor and unimpressive. A very bad system, on the other hand, would make the language too difficult to analyze. A compromise was made and the chosen system had a somewhat average word-error-rate (from the tested systems) amounting to 27.6%. For those that decided to use the lattice as their input, they could count on the oracle word-error-rate of 17.7%, presenting a sort of floor for the error rate of the given ASR system. That means that the system likely made many errors which were unrecoverable from the post-editing perspective, therefore an improvement of even a few absolute percentage points made a significant difference.

5. Participating systems and results

There were 20 submissions from 8 different teams. The results were very close in a few cases. To make the assessment a bit more interesting we calculated both the error rate compared to the reference (which was not known to the submitters), as well as the single best output (which was known to the submitters). The latter shows the number of changes the submission made to the original.

The submission titled “MLM+bert_base_polish” only had 171 out of 462 files submitted and cannot be directly compared with others. The result for only the present files was 26.8%. If we extract these 171 files from the winning submission, its result would yield 23.4%, so this submission would have lost in this comparison as well.

The winning submission by the Wrocław University team titled “flair-bigsmall” was submitted twice (both submissions were identical) and it also lacked the output for 2 files. The result in the table is calculated assuming these two files were completely incorrect. If we didn’t account for these files, the result would be 24.0%. It was a close call with between the Wrocław University and Adam Mickiewicz University teams, but ultimately the former team won

Acknowledgements

The creation of the training corpora was funded by the Clarin-PL project. The test corpus was funded by the UMO-2018/30/E/HS2/00035 (SONATA BIS 8) research grant funded by the National Science Centre (2019-2024).

Table 2: Results of the submissions to Task 1

Submitter	Submission	Affiliation	WER %	Changes
Krzysztof Wróbel	KRS + spaces	UJ, AGH	25.9	3.6%
Krzysztof Wróbel	KRS	UJ, AGH	26.9	1.6%
Dariusz Kłeczek	Polbert	skok.ai	26.9	2.1%
Tomasz Ziętkiewicz	BiLSTM-CRF edit-operations tagger	AMU	24.7	6.2%
Kornel Jankowski	base-4g-rr	Samsung	27.7	2.0%
Adam Kaczmarek	t-REx_k10	UWr	24.9	14.2%
Adam Kaczmarek et al.	t-REx_k5	UWr	25.0	14.2%
Adam Kaczmarek et al.	t-REx_fbs	UWr	24.0	17.2%
Krzysztof Wolk	PJA_CLARIN_1k	PJAIT	33.5	9.1%
Krzysztof Wołk	PJA_CLARIN_10k	PJAIT	32.0	9.6%
Krzysztof Wołk	PJA_CLARIN_20k	PJAIT	31.8	9.9%
Krzysztof Wołk	PJA_CLARIN_40k	PJAIT	31.8	10.3%
Krzysztof Wołk	PJA_CLARIN_50k	PJAIT	31.8	10.2%
Krzysztof Wołk	CLARIN_SEJM_40k	PJAIT	33.7	19.1%
Krzysztof Wołk	CLARIN_SEJM_50k	PJAIT	32.5	17.7%
Jim O'Regan	MLM+bert_base_polish	n/a	26.8	2.1%
Tomasz Syposz et al.	tR-Ex_xk	UWr	25.7	18.1%
Tomasz Syposz et al.	tR-Ex_fbs	UWr	24.0	17.2%
Tomasz Syposz et al.	tR-Ex_fx	UWr	25.0	23.3%
Tomasz Syposz et al.	tR-Ex_kxv2	UWr	25.5	17.1%

References

Bassil Y. and Alwani M. (2012). *Post-Editing Error Correction Algorithm for Speech Recognition using Bing Spelling Suggestion*. arXiv:1203.5255.

Chmiel A., Kajzer-Wietrzny M., Koržinek D., Janikowski P., Jakubowski D. and Polakowska D. (2020). *Fluency Parameters in the Polish Interpreting Corpus (PINC)*. In *Empirical Investigations into the Forms of Mediated Discourse at the European Parliament*. Language Science Press.

Koržinek D. (2019). *Results of the PolEval 2019 Shared Task 5: Automatic Speech Recognition Task*. In Ogrodniczuk M. and Kobyliński Ł. (eds.), *Proceedings of the PolEval 2019 Workshop*, pp. 73–78. Institute of Computer Science, Polish Academy of Sciences.

Koržinek D., Marasek K., Brocki Ł. and Wołk K. (2017). *Polish Read Speech Corpus for Speech Tools and Services*. arXiv:1706.00245.

Kubis M., Vetulani Z., Wypych M. and Ziętkiewicz T. (2020). *Open Challenge for Correcting Errors of Speech Recognition Systems*. arXiv:2001.03041.

Lee W., Shin J. and Lee J.-H. (2019). *Transformer-based Automatic Post-Editing Model with Joint Encoder and Multi-source Attention of Decoder*. In *Proceedings of the 4th Conference on Machine Translation (Volume 3: Shared Task Papers, Day 2)*, pp. 112–117. Association for Computational Linguistics.

Lopes A. V., Farajian M. A., Correia G. M., Trenous J. and Martins A. F. (2019). *Unbabel's Submission to the WMT2019 APE Shared Task: BERT-based Encoder-Decoder for Automatic Post-Editing*. arXiv:1905.13068.

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

Ogrodniczuk M. (2018). *Polish Parliamentary Corpus*. In Fišer D., Eskevich M. and de Jong F. (eds.), *Proceedings of the LREC 2018 Workshop* ParlaCLARIN: Creating and Using Parliamentary Corpora, pp. 15–19. European Language Resources Association.

Shterionov D., Wagner J. and do Carmo F. (2019). *APE through Neural and Statistical MT with Augmented Data. ADAPT/DCU Submission to the WMT 2019 APE Shared Task*. In *Proceedings of the 4th Conference on Machine Translation (Volume 3: Shared Task Papers, Day 2)*, pp. 132–138. Association for Computational Linguistics.

Vu T.-T. and Haffari G. (2018). *Automatic Post-Editing of Machine Translation: A Neural Programmer-Interpreter Approach*. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 3048–3053. Association for Computational Linguistics.

Xiong W., Wu L., Alleva F., Droppo J., Huang X. and Stolcke A. (2018). *The Microsoft 2017 Conversational Speech Recognition System*. In *Proceedings of 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5934–5938. IEEE.

Xu H., Chen T., Gao D., Wang Y., Li K., Goel N., Carmiel Y., Povey D. and Khudanpur S. (2018). *A Pruned RNNLM Lattice-Rescoring Algorithm for Automatic Speech Recognition*. In *Proceedings of 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5929–5933. IEEE.

t-REx: The Rescorer-Extender Approach to ASR Improvement

Adam Kaczmarek (VoiceLab.AI)

Tomasz Syposz, Michał Martusewicz, Jan Chorowski, Paweł Rychlikowski (Institute of Computer Science, University of Wrocław)

Abstract

This paper presents our contribution[1] to PolEval 2020[2] Task 1: Post-editing and rescoring of automatic speech recognition results. Our system was scored first and achieved WER 24.3% (compared to 27.6% for original ASR system).

Keywords

natural language processing, computational linguistics, linguistic engineering, lattice, flair, word2vec, n-grams

1. Introduction

Automatic Speech Recognition is one of the standard natural language processing tasks for which the input is a recording and the output is a transcription of that recording. Most often, first the input waveform is transformed to feature vectors sequence (MFCC, or spectrogram), and then the ASR system tries to find the most likely sequence of words for these features. Nevertheless, it is interesting to ask whether, for a specific speech recognition system, looking only at its results (presented as the most likely sequence of words, a set of top-N sequences, or a lattice describing many variants of the recognition together with their costs), we will be able to significantly improve the performance of the original system.

[1] Code and models are available at: `https://github.com/adamjankaczmarek/poleval2020`

[2] `http://2020.poleval.pl`

2. Task description

The goal of PolEval 2020 *Task 1. Post-editing and rescoring of automatic speech recognition results* is to create a system for converting a sequence of words from a specific automatic speech recognition (ASR) system into another sequence of words that more accurately describes the actual spoken utterance. In the training data for every utterance we have its correct transcription, as well as the output of an ASR system (top N word sequences, and the full lattice containing acoustic and language model costs).

3. Our system

We have decided to implement our system based on the following ideas:

1. Use several Language Models trained on various text corpora including large parts of the Polish Parliamentary Corpus (Ogrodniczuk 2012, 2018) and the reference transcriptions given as training data.
2. Experiment both with standard n-gram models, and with modern deep neural networks architectures (including recurrent models defined in Flair library, and BERT transformer model).
3. Implement our own beam search on lattices which can handle the above mentioned language models.
4. Extend the original lattice with extra words, by adding new nodes and edges.

3.1. Lattice extending

The idea behind lattice extensions is fairly simple: when an ASR creates a lattice it may miss some words from the real sentence. These words cannot be recovered with regular lattice rescoring. We can counteract it by adding new edges to the lattice. The first step was to find for each word in the lattice a set of possible extensions which are words within a small edit (Levenshtein) distance. For this we used a structure called "BK-tree" filled with unigrams[3] from the NKJP corpus (Przepiórkowski et al. 2012). The search range was dependent on the chosen word's length. There were a few assumptions and restrictions. We did not want to extend short words, because almost all of them are in the shortest range. We also knew the longer the word, the smaller the set of words in a given edit distance. For computational reasons, the set of possible extending words and the search radius had to be rather small. We experimentally established that it's better to use the distance between words converted to phonemes.

[3] `http://zil.ipipan.waw.pl/NKJPNGrams`

The final search radius was given by the formula:

$$x = \text{length of phonemes form of a word}$$

$$\text{search radius } (x) = \begin{cases} 0, & \text{if } x < 4 \\ 1, & \text{if } 4 \leq x < 8 \\ 2, & \text{if } x \geq 8 \end{cases}$$

After this we need to cut down the lattice because it is much too big — 30 times bigger than the original one. Our idea is loosely inspired by graph-based text summarization (Mihalcea and Tarau 2004). We create a directed graph describing the relationship between words in the lattice. The intuition is as follows: nodes are connected by an edge if it is natural that they can appear in the same utterance.

We create edges between two nodes if the following score is bigger than 1:

$$\left[v * \text{cosine\_distance}(\text{parent}, \text{child}) + b * \log_{10} \frac{\text{bi-gram}[\text{parent}, \text{child}]}{\text{uni-gram}[\text{parent}]} \right]^{-1}$$

where cosine_distance is equal to one minus cosine similarity between word2vec vectors for given words, uni-gram$[w]$ is the unigram count for word w, bi-gram$[w1, w2]$ is the 2-gram count for $w1$ and $w2$, and b and v are constants. We, again, used n-grams with their respective counts collected on NKJP. During experiments, we set b to -40 and v to 333. These constant have been chosen such that only about 30% of edges have a score above 1, which results in a sparse graph. The next step is to perform a PageRank algorithm (Brin and Page 1998) on the resulting graph. When the Page Rank iterations are done, we keep up to 15 possible extensions of each word by selecting the candidates with the biggest page rank value.

Lattice extending results

The average oracle WER on extended lattices from our train set was 9.8%. After reducing it with PageRank algorithm by more than 3 times we achieved 11.3% (while oracle WER on original lattice set is equal to 15.4%). Of course bigger lattices (with smaller Oracle-WER) give a bigger chance of substantial WER decrease, however, they also make finding the correct path much more difficult. We believe that the PageRank reduced version of the lattice is a good trade-off.

3.2. Lattice rescoring

Our system performs two lattice rescoring runs. First, we find a set of paths using a n-gram language model. We then rescore the top paths from the second stage using neural language models.

Beam search

The first lattice rescoring run is a beam search-based rescorer. We investigate two variants of language models to calculate the total rescored cost according to the equation:

$$\text{cost} = \text{cost}_{\text{language}} + 0.02 \times \text{cost}_{\text{acoustic}}$$

The first variant uses beams of size 5 000 and 10 000 with KenLM (Heafield 2011) language model trained on a large part of the Polish Parliamentary Corpus. The second beam search variant uses a beam of size 1 000 with Flair (Akbik et al. 2018) language model. We used a pre-trained model `pl-opus` primarily based on the Wikipedia and OPUS corpus (Tiedemann 2012). The model was then fine-tuned on the Polish Parliamentary Corpus and its subcorpus of transcriptions provided in training data.

Beam rescoring

Second lattice rescoring run is a pure language model based rescoring using PolBert[4] model. Language model score is computed as the sum of per-token scores of BERT model. Total score of an utterance is calculated according to the equation above.

3.3. Results

We present results of our system, along with other submissions to the PolEval 2020 Task 1 in Table 4. Systems named as *t-REx_k5* and *t-REx_k10* show results from beam search decoder with KenLM language model and beams of respective size 5 000 and 10 000. System *t-REx_fbs* denotes the winning solution with beam search decoder with beam size 1 000 and the fine-tuned Flair language model. Additionally, we provide scores for systems evaluated out of the competition. Due to a technical issue, the winning solution did not contain results for all utterances. We have corrected it, and here we provide the score for the winning system with missing transcripts taken from original `1best` file from the test dataset. Next systems are: *t-REx_bert* — results from *t-REx_fbs* additionally rescored with PolBert model and two variants of the system with lattice extending — *t-REx_fbsx*.

3.4. Future works

There are many possible future directions to extend our work. We enumerate a few of them.

Text corpora

To train a language model we have used two corpora: a large part of Polish Parliamentary Corpus and a reference transcription from PolEval training data. These two corpora have several drawbacks when treated as a base for training language models: the small one is

[4] `https://github.com/kldarek/polbert`

Table 1: Results of PolEval 2020 Task 1

System name	Affiliation	WER %	Changes %
KRS + spaces	UJ. AGH	25.9	3.6
KRS	UJ. AGH	26.9	1.6
Polbert	https://skok.ai/	26.9	2.1
BiLSTM-CRF edit-operations tagger	Adam Mickiewicz University	24.7	6.2
base-4g-rr	Samsung R&D Institute Poland	27.7	2.0
t-REx_k10	University of Wrocław	24.9	14.2
t-REx_k5	University of Wrocław	25.0	14.2
t-REx_fbs	**University of Wrocław**	**24.31**	**17.2**
PJA_CLARIN_1k	Polish-Japanese Academy of IT	33.5	9.1
PJA_CLARIN_10k	Polish-Japanese Academy of IT	32.0	9.6
PJA_CLARIN_20k	Polish-Japanese Academy of IT	31.8	9.9
PJA_CLARIN_40k	Polish-Japanese Academy of IT	31.8	10.3
PJA_CLARIN_50k	Polish-Japanese Academy of IT	31.8	10.2
CLARIN_SEJM_40k	Polish-Japanese Academy of IT	33.7	19.1
CLARIN_SEJM_50k	Polish-Japanese Academy of IT	32.5	17.7
MLM+bert_base_polish	—	73.9	2.1
Non-competitive results			
t-REx_fbs (with 2 from 1best)	University of Wrocław	23.93	16.6
t-REx_bert (with 8 from 1best)	**University of Wrocław**	**23.4**	**17.0**
t-REx_fbsx_50	University of Wrocław	25.09	23.3
t-REx_fbsx_150	University of Wrocław	23.67	19.1

simply too small (it has only 5 MB of text), while the large one is not a real transcription: text contains some abbreviations, numbers are written with digits not words (so we lose the actual form of them), maybe some times text is corrected. It is possible to create a version of these corpora which is more suitable for speech recognition. It can be done by applying the text normalization procedures similar to the first pass of a TTS system. Of course it is quite easy to train a neural network solving this task, and one can produce the training data from unannotated corpora.

Acoustic information

In this task we had access only to the limited acoustic information: in lattices for every edge we had an LM cost and acoustic cost (which was a mixture of HMM cost and emission cost). Moreover, this information was restricted to phonemes occurring in words present in a lattice. It makes it difficult to expand a lattice (because for many phonemes we don't have any information about their acoustic costs). Our assumption, that the cost of new edges is a sum of an acoustic cost from the closest path in the lattice, and the cost connected with phonemes insertions, deletions and replacement is in general false: the new version in fact can have smaller acoustic costs. One can overcome these problems by slight modification of the task. There are two possibilities for doing this. First one is to record raw acoustic costs,

ie. probabilities of every phoneme in every 10 ms of speech signal. The second option is to add to the input data the lattices containing phonemes (the result of the recognition with a vocabulary containting only phonemes, with a very simple phoneme bigram language model).

More careful lattice extensions

When expanding the lattice we can see the trade-off between bigger lattices which can substantially reduce the Oracle WER and the ease of beam search: with larger lattices there are more options to consider. Maybe it is worth concentrating only on very typical errors (as changing the word into the other word with the same lemma, joining some very short words with their neighbours, and so on).

Language models with holes

Standard language models give a probability distribution over the word w_T given its predecessors $w_1 \dots w_{T-1}$, and they are trained only with the correct prefixes. When the ASR system results are of a low quality, then, even in wide lattices, there are no perfect paths. Nevertheless, we use the pretrained LM for the data which is substantially different from the data used during training. We believe that careful consideration of this issue can yield further WER reduction.

Acknowledgements

Jan Chorowski and Paweł Rychlikowski were partially supported from the NCN OPUS 2019/35/B/ST6/04379 grant. The authors also thank the PL-Grid consortium for computational resources.

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.

Brin S. and Page L. (1998). *The Anatomy of a Large-Scale Hypertextual Web Search Engine*. In *Proceedings of the 7th International World-Wide Web Conference (WWW 1998)*. `http://ilpubs.stanford.edu:8090/361/`.

Heafield K. (2011). *KenLM: Faster and Smaller Language Model Queries*. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pp. 187–197. Association for Computational Linguistics.

Mihalcea R. and Tarau P. (2004). *TextRank: Bringing Order into Text*. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pp. 404–411. Association for Computational Linguistics.

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

Ogrodniczuk M. (2018). *Polish Parliamentary Corpus*. In Fišer D., Eskevich M. and de Jong F. (eds.), *Proceedings of the LREC 2018 Workshop* ParlaCLARIN: Creating and Using Parliamentary Corpora, pp. 15–19. European Language Resources Association.

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

Tiedemann J. (2012). *Parallel Data, Tools and Interfaces in OPUS*. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC 2012)*, pp. 2214–2218. European Language Resources Association.

Post-editing and Rescoring of ASR Results with Edit Operations Tagging

Tomasz Ziętkiewicz
(Adam Mickiewicz University, Samsung Research Poland)

Abstract

This paper presents a system developed at Adam Mickiewicz University and Samsung Research Poland for submission to PolEval 2020 Task 1: Post-editing and rescoring of ASR results.[1] The goal of the task was to "create a system for converting a sequence of words from a specific automatic speech recognition (ASR) system into another sequence of words that more accurately describes the actual spoken utterance." The paper describes a novel approach to the problem of correcting speech recognition errors, by tagging with edit operations tags. We show that the proposed system can achieve results comparative with other approaches while providing a high level of control over how the system works, which make it suitable for production settings. Beside the detailed description of the method, the paper presents related works, task and data description, data augmentation techniques and results analysis.

Keywords

natural language processing, speech processing, ASR, speech recognition, error correction, rescoring, error correction

1. Introduction

Automatic speech recognition is a computer science problem which both academia and industry have been working on for more than six decades now. Thanks to constant progress in the field, automatic speech recognition performance in some use cases approaches human level (Spille et al. 2018).

Still, even state-of-the-art speech recognition systems make mistakes, especially in difficult conditions or in unknown domains. These mistakes can be addressed in the ASR system itself, by training or adapting the ASR model with additional data or modifying its parameters.

[1] `http://2020.poleval.pl/tasks/task1/`

This approach, however, is not always possible – it requires large amounts of training data and computing power. In the case of cloud-based speech recognition services, it can be just impossible to modify the model used by the ASR at all. Even if one can modify the ASR model itself and have sufficient resources to do so, the learning capacity of the chosen ASR system architecture can limit performance achieved by the system in some specific cases (Guo et al. 2019).

To further improve speech recognition results in the above scenarios we can try to fix mistakes made by an ASR system on its output, in a post-processing step. In this approach hypothesis returned by the speech recognition system is processed by an error correction model. The model is trained on previous mistakes of the ASR system and tries to detect and correct mistakes present in ASR hypotheses.

2. Related work

Errattahi et al. (2018) present a comprehensive overview of speech recognition errors correction methods. In a more recent paper Guo et al. (2019) propose machine translation-inspired sequence-to-sequence approach which learns to "translate" hypotheses to reference transcripts. To augment training data authors use all N-best hypotheses to form pairs with reference sentences, generate audio data using speech synthesis and add noise to the source recordings. The resulting training set consist of 640M reference-hypothesis pairs. The proposed system achieves 18.6% relative WER (Word Error Rate) reduction. Zhang et al. (2019) use a similar approach for Mandarin speech recognition, but propose Transformer model for spelling correction. Authors report the result of 22.9% relative CER (Character Error Rate) improvement. Hrinchuk et al. (2019) developed a transformer-based sequence to sequence model, trained on a training set consisting of 2.5M examples which achieved relative WER reduction of around 12% on English datasets. More recently, Mani et al. (2020) applied machine translation techniques for the same problem but for a specific domain of medical reports. Depending on the ASR system used they achieved relative WER reduction of around 16% (41 to 34) and 3% (35,8 to 34,5).

To our best knowledge, there was only one shared task on ASR error correction before PolEval 2020: "Open Challenge for Correcting Errors of Speech Recognition Systems" (Kubis et al. 2020). This challenge also focused on ASR error correction for Polish speech recognition but in a slightly different setting. Dataset consisted of just 1-best hypotheses and reference sentences and the ASR system being used was not open-sourced.

Malmi et al. (2019) use edit operation tagging for sentence fusion, sentence splitting, abstractive summarization, and grammar correction tasks. Transformer architecture is used for the tagger. For the grammatical error correction task which is most similar to the problem discussed in this paper, authors report competitive results for small training datasets and very short inference times compared with sequence to sequence approach.

One can also find research on a closely related topic of grammatical error correction: see e.g. (Grundkiewicz and Junczys-Dowmunt 2018) for sequence-to-sequence approach or (Omelianchuk et al. 2020) for tagging approach. There are also shared tasks for grammatical correction, like (Bryant et al. 2019) or (Ng et al. 2014).

3. Data

Data provided by organizers consisted of:

— reference transcriptions of utterances (except for evaluation dataset)

— hypotheses returned by speech recognition system in 3 different forms:
  — 1-best output – each utterance containing a single best transcript of the ASR output
  — n-best output – each utterance containing up to 100 best alternative hypotheses of the ASR output
  — lattice output – each utterance containing a list of arcs forming a lattice of the ASR output

The speech recognition system used to produce the hypotheses was trained using Kaldi (Povey et al. 2011) ASR toolkit on CLARIN-PL studio corpus[2] (Marasek et al. 2015). Recordings and references used to create data came from 3 different corpora:

— CLARIN-PL studio corpus (Marasek et al. 2015) (the same used to train the ASR model)

— Polish Parliamentary Corpus (PPC)[3]

— subset of Polish interpreting corpus (PINC)[4]

Sizes of the resulting datasets are shown in Table 1.

Table 1: Statistics of datasets provided by organizers

	CLARIN-PL studio			**PPC**	**PELCRA-PARL**	**PINC**
	Train	**Test**	**Dev**	**Train**	**Train**	**Eval**
Sentences	11 222	1362	1229	6752	8066	462
WER	9.59	12.08	12.39	45.57	59.95	27.6
Oracle WER	3.75	4.72	4.93	30.71	–	17.7
Avg. length	22	21	21	104	12	169
Min. length	3	6	7	1	2	70
Max. length	55	53	49	341	47	435

3.1. Data augmentation

Because task description allowed using external data, we decided to prepare an additional corpus of ASR hypotheses to augment the training set. Aim of the task was to create ASR error correction model tailored for specific ASR system. What follows – train, test and development datasets provided by organizers were produced by the same ASR system as the evaluation data used for the final assessment of submissions. Therefore, to obtain optimal results, the

[2] `https://clarin-pl.eu/dspace/handle/11321/236`

[3] `http://mowa.clarin-pl.eu/korpusy/parlament/parlament.tar.gz` (Ogrodniczuk 2018)

[4] `https://pincproject2020.wordpress.com/`

additional training data should also be prepared using the same speech recognition model. To achieve it, we trained the model using Kaldi recipe mentioned in task description[5] (Korzinek et al. 2017) and the same training data as used by organizers (CLARIN-PL studio corpus). We used the ASR to decode PELCRA-PARL corpus (Pęzik 2018), which similarly to CLARIN-PL studio corpus, contains recordings of Polish Parliament speeches. We chose this corpus because of a similar domain to the one used in the training and evaluation datasets. By adding the corpus, which consisted of 8066 sentences, we extended the number of training examples from 11 222 to 19 288. For statistics of the corpus see Table 1. To achieve a larger data set covering wider variety of errors generated by the ASR system, we used up to 10-best hypotheses to generate more reference-hypothesis pairs. As a result, the training set finally consisted of 110 059 hypothesis-reference pairs. This is still quite a small dataset in comparison with training sets used by Guo et al. (2019) (640M) or Hrinchuk et al. (2019) (2.4M), but as shown by Malmi et al. (2019), tag-apply models (see Section 2) requires much smaller datasets then sequence-to-sequence models used for the same tasks.

4. Method

Contrary to the sequence-to-sequence approach which learns the hypothesis-reference mapping directly, we propose a tag-apply approach. Training of a model in this approach consists of the following steps: input and expected output sequences are compared. Based on differences found, edit operation tags are added to the input, describing how to transform it into expected output. This way a corpus of input sequences tagged with operation tags is created. The corpus is then used to train a tagging model. On inference time, the input sequences are tagged with edit operations by the model. The edit operations are then applied to the input sentence.

Below, we describe in detail how training of the model and correction of errors are performed. Given a corpus of pairs of (potentially incorrect) ASR hypotheses and corresponding reference transcriptions, a corpus of ASR hypotheses tagged with edit operation tags is prepared (see *Labeler* block in Figure 2) Every pair of hypothesis and reference sentences is compared using Ratcliff-Obershelp algorithm (Ratcliff and Metzener 1988)[6] for approximate string matching. As a result, for every such pair a list of operation codes ("insert", "delete", "replace" or "equal") is obtained, describing how to transform sequences of tokens from the hypothesis into corresponding sequences of reference tokens.

Edit operations codes "replace" and "insert" returned by string matching algorithm need to be combined with tokens from the reference sentence to contain enough information for transforming hypothesis into a corresponding reference sentence. For example, given reference sentence: *cats are cute* and hypothesis: *cat are cute* the returned opcodes would be *replace equal equal*. To make edit operation tag from the operation code "replace" and reference token *cats* we concatenate them with _ separator to get *replace_cats*. Resulting operation tags for the given example would be: *replace_cats equal equal*.

[5] `https://github.com/danijel3/ClarinStudioKaldi`

[6] `https://docs.python.org/3/library/difflib.html`

Because we want the trained tagging model to be able to generalize into unseen words, we decided to extend the set of available edit operations to include more fine-grained operations. To do so, we analyze differences within the mismatched tokens and whenever possible we try to use more fine-grained operations. In the above example, *replace_cats* would be changed into *append_s* – an edit operation that appends letter *s* to the word being tagged. For complete example of processing hypothesis-reference pair, see Figure 1.

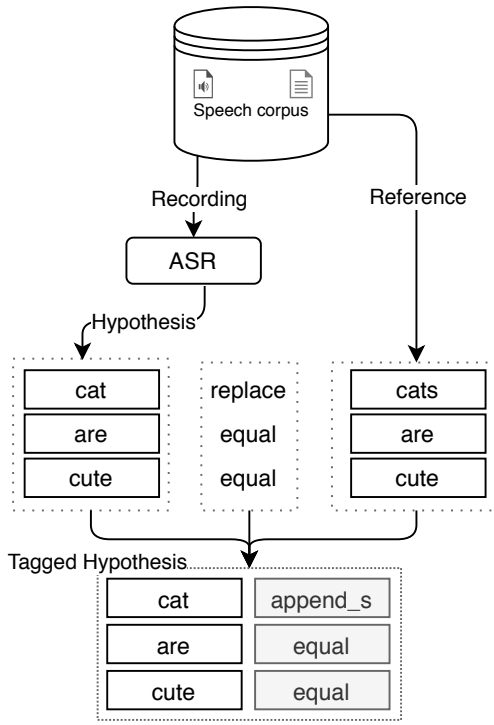


Figure 1: Example of creating a tagged hypothesis sentence from hypothesis-reference pair

Edit operations are chosen in a deterministic manner from a predefined set of edit operation classes. Examples of edit operations classes can be found in Table 2.

Having prepared a corpus of hypothesis sentences labeled with edit operations, a tagger model can be learned (see. *Tagger trainer* block in Figure 2). We train the tagger using Flair[7] Sequence Tagger (Akbik et al. 2019) with Polish Flair word embeddings (Borchmann et al. 2018).

During the inference stage, When the system is used to correct ASR output, without knowing the corresponding reference sentence, the tagging model is used to insert edit operation tags (see. *Tagger* block in Figure 2). Edit operations are then applied to words in hypothesis (see *Editor* block in Figure 2) to correct potential errors. Using taggers score values returned

[7] `https://github.com/flairNLP/flair`

Table 2: Examples of edit operations

Name	Description	Example
`del`	deletes a token	"a" → ""
`append_s`	appends given suffix to the token	"cat" → "cats"
`add_prefix_`	prepends given prefix to the token	"owl" → "howl"
`remove_suffix_1`	removes 1 character from the end of the token	"cats" → "cat"
`remove_prefix_1`	removes 1 character from the beginning of the token	"howl" → "owl"
`join`	joins token with previous one	"book store" → "bookstore"
`join_-`	joins token with previous one using given separator	"long term" → "long-term"
`replace_`	replaces token with given string	"cat" → "hat"

together with tag labels, one can also control precision of the system by performing only these edit operations which were returned with score value above some threshold.

5. Evaluation

The only metric used to officially compare submissions to the task was average Word Error Rate (WER) of hypotheses corrected by the proposed system. WER is given by following formula:

$$WER = \frac{N_{del} + N_{sub} + N_{ins}}{N_{ref}}$$

where N_{sub} – number of substitutions, N_{del} – number of deletions, N_{ins} – number of insertions, N_{ref} – length of reference sentence.

Results achieved by the proposed system are shown in Table 4.

Table 3: Word Error Rates for input data and the proposed system

	CLARIN			PPC	PINC
	Train	**Test**	**Dev**	–	**Eval**
Raw ASR 1best	9.59	12.08	12.39	45.64	27.6
Lattice oracle WER	3.75	4.72	4.93	30.71	17.7
Edit operation tagger (from 1best)	–	10.7	–	–	24.7
Absolute WER Reduction	–	11.42%	–	–	10.50%
Relative WER Reduction	–	11.42%	–	–	10.50%

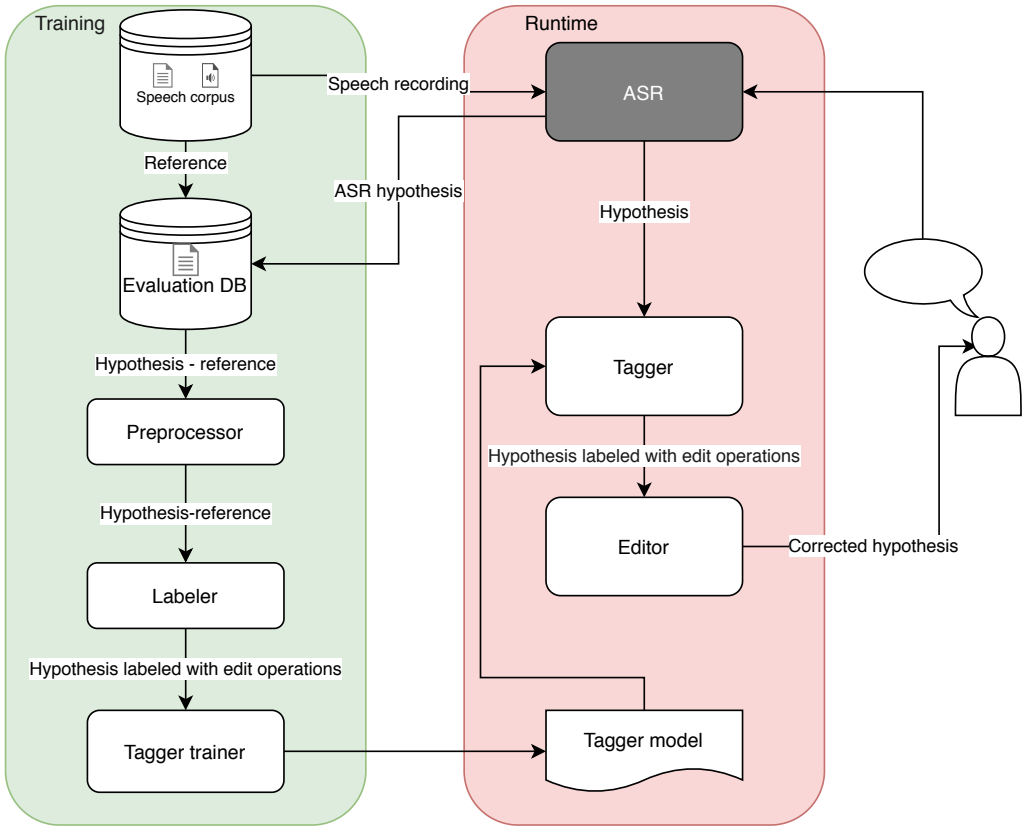


Figure 2: Dataflow in the system

According to official Poleval 2020 results, our submission to the shared task scored second best result in terms of WER level.[8] We believe that by further increasing the training dataset and experimenting with tagger architecture the result could be even further improved. Organizers also report percentages of changes introduced by proposed systems into input sentences. In this respect, our submission (6.2%) is fairly more "conservative" then the winning one (17.2%) while achieving only slightly higher WER (24.70 vs 24.31) This is especially important in a commercial setting where error correction systems are expected to prefer precision over recall.

6. Conclusion

We presented a novel approach to correcting speech recognition errors in a post-processing step. Instead of learning transformation from hypothesis to reference directly, we propose to

[8] http://poleval.pl/results/

learn tags that describe how to turn the hypothesis into reference. A similar approach has been proved to be valuable in other, yet similar tasks. Achieved results suggest its applicability to the ASR error correction problem. We showed that the proposed solution has advantages over sequence-to-sequence systems inspired by machine translation. It offers precise control over how the error correction model works and it requires smaller training sets. When compared with competitive PolEval submissions our approach shows a low percentage of introduced changes. These features suggest applicability of the solution in a production setting.

References

Akbik A., Bergmann T., Blythe D., Rasul K., Schweter S. and Vollgraf R. (2019). *FLAIR: An Easy-to-Use Framework for State-of-the-Art NLP*. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pp. 54–59. Association for Computational Linguistics.

Borchmann Ł., Gretkowski A. and Graliński F. (2018). *Approaching Nested Named Entity Recognition with Parallel LSTM-CRFs*. In Ogrodniczuk M. and Kobyliński Ł. (eds.), *Proceedings of the PolEval 2018 Workshop*, pp. 63–73. Institute of Computer Science, Polish Academy of Sciences.

Bryant C., Felice M., Andersen Ø. E. and Briscoe T. (2019). *The BEA-2019 Shared Task on Grammatical Error Correction*. In *Proceedings of the 14th Workshop on Innovative Use of NLP for Building Educational Applications*, pp. 52–75. Association for Computational Linguistics.

Errattahi R., El Hannani A. and Ouahmane H. (2018). *Automatic Speech Recognition Errors Detection and Correction: A Review*. „Procedia Computer Science", 128, p. 32–37.

Grundkiewicz R. and Junczys-Dowmunt M. (2018). *Near Human-Level Performance in Grammatical Error Correction with Hybrid Machine Translation*. arXiv:1804.05945.

Guo J., Sainath T. N. and Weiss R. J. (2019). *A Spelling Correction Model for End-to-end Speech Recognition*. In *Proceedings of the 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2019)*, pp. 5651–5655.

Hrinchuk O., Popova M. and Ginsburg B. (2019). *Correction of Automatic Speech Recognition with Transformer Sequence-to-sequence Model*. arXiv:1910.10697.

Koržinek D., Marasek K., Łukasz Brocki and Wołk K. (2017). *Polish Read Speech Corpus for Speech Tools and Services*. arXiv:1706.00245.

Kubis M., Vetulani Z., Wypych M. and Ziętkiewicz T. (2020). *Open Challenge for Correcting Errors of Speech Recognition Systems*. arXiv:2001.03041.

Malmi E., Krause S., Rothe S., Mirylenka D. and Severyn A. (2019). *Encode, Tag, Realize: High-Precision Text Editing*. arXiv:1909.01187.

Mani A., Palaskar S., Meripo N. V., Konam S. and Metze F. (2020). *ASR Error Correction and Domain Adaptation Using Machine Translation*. arXiv:2003.07692.

Marasek K., Koržinek D., Brocki Ł. and Jankowska-Lorek K. (2015). *Clarin-PL Studio Corpus (EMU)*. CLARIN-PL digital repository, http://hdl.handle.net/11321/236.

Ng H. T., Wu S. M., Briscoe T., Hadiwinoto C., Susanto R. H. and Bryant C. (2014). *The CoNLL-2014 Shared Task on Grammatical Error Correction*. In *Proceedings of the 18th Conference on Computational Natural Language Learning: Shared Task*, pp. 1–14. Association for Computational Linguistics.

Ogrodniczuk M. (2018). *Polish Parliamentary Corpus*. In Fišer D., Eskevich M. and de Jong F. (eds.), *Proceedings of the LREC 2018 Workshop* ParlaCLARIN: Creating and Using Parliamentary Corpora, pp. 15–19. European Language Resources Association.

Omelianchuk K., Atrasevych V., Chernodub A. and Skurzhanskyi O. (2020). *Gector – Grammatical Error Correction: Tag, Not Rewrite*. arXiv:2005.12592.

Povey D., Ghoshal A., Boulianne G., Burget L., Glembek O., Goel N., Hannemann M., Motlicek P., Qian Y., Schwarz P., Silovsky J., Stemmer G. and Vesely K. (2011). *The Kaldi Speech Recognition Toolkit*. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society. IEEE Catalog No.: CFP11SRW-USB.

Pęzik P. (2018). *PELCRA PARL corpus*. CLARIN-PL digital repository, http://hdl.handle.net/11321/545.

Ratcliff J. W. and Metzener D. (1988). *Pattern Matching: the Gestalt Approach*. https://www.drdobbs.com/database/pattern-matching-the-gestalt-approach/184407970?pgno=5. Dr. Dobb's Journal.

Spille C., Kollmeier B. and Meyer B. T. (2018). *Comparing Human and Automatic Speech Recognition in Simple and Complex Acoustic Scenes*. „Computer Speech & Language", 52, p. 123–140.

Zhang S., Lei M. and Yan Z. (2019). *Automatic Spelling Correction with Transformer for CTC-based End-to-End Speech Recognition*. arXiv:1904.10045.

Post-editing and Rescoring of Automatic Speech Recognition Results with OpenNMT-APE

Dominika Wnuk, Krzysztof Wołk
(Polish-Japanese Academy of Information Technology)

Abstract

This paper presents our submission to Task 1 of PolEval 2020 contest. OpenNMT-APE system was applied in post-editing and rescoring of ASR output for Polish language. The system trained on Clarin-PL and Polish Parliament corpora was then evaluated with Word Error Rate metric. The experiments performed differed in dataset size and training steps, and even though the results fell behind the average WER, the proposed system is novel and worth further experimenting.

Keywords

Polish, natural language processing, automatic post-editing, APE, Word Error Rate, WER, speech recognition

1. Introduction

Automatic post-editing (APE) is a supervised learning method of automatically correcting errors in the output of a machine translation (MT) algorithm (Negri et al. 2018). Post-editing was traditionally performed by a human editor correcting mistakes in machine translation (Correia and Martins 2019). The present paper describes the solution proposed to Task 1 of PolEval 2020, an annual contest regarding natural language processing of Polish. In the submitted solution, the automatic post-editing method was applied to the results of an automatic speech recognition system. Hence, both the source and the target data being the same language, and not a language pair.

2. Task description and data

Task 1 was meant to refine the results of automatic speech recognition (ASR; Levis and Suvorov 2012) through post-editing and rescoring. The created system was supposed to convert utterances generated by an ASR (transcriptions of oral utterances) into the new ones which would better reflect the actually spoken phrases. As the ASR used to generate the training data was also used in the evaluation stage, the created system had to be tailored to the meet the specifics of this particular ASR only, and not create a global solution. In order to simplify the task, all reference and output files were simple sequences of words, without capitalization, digits, punctuation, symbols nor abbreviations.

There were four data sets provided, each consisting of four files: the actual record transcript for reference, 1-best output with a sole best transcript of the ASR system, n-best output with up to 100 optimal transcripts, and lattice output in form of a graph representing different hypotheses for the utterance. Three data sets: training, test and development came from on from the Clarin-PL studio corpus (Marasek et al. 2015), and the fourth – the Polish parliament corpus (Ogrodniczuk 2012, 2018).

For the purpose of training, two data sets were prepared: one consisting only of Clarin-PL sets, and the second combining the Clarin-PL and the Polish parliament corpora.

3. Post-editing system

For the training OpenNMT-APE, part of the open source toolkit for neural machine translation system (Klein et al. 2017) was selected. Full code is available on Github.[1] OpenNMT-APE applies transfer learning by implementing the pretrained BERT model (Devlin et al. 2019). The novelty of this approach is the use of BERT both as encoder and decoder in a seq2seq (sequence to sequence) model (Correia and Martins 2019, Lopes et al. 2019).

As per OpenNMT-APE documentation (available on Github), the following parameters had their optimal values set:

— Validation steps: 1000

— Checkpoint: 30

— Warmup steps: 5000

— Learning rate: 0.00005

— Average decay: 0.0001

Source and target sequence length was defined as 200, while train steps and start decay steps were adjusted per each experiment (as shown in Table 1).

Additionally, as advised by Correia and Martins (2019), the self-attention was shared between encoder and decoder, and the context attention had the same weights as the self-attention. The dropout rate and the label soothing were both set as 0.1.

[1] `https://github.com/deep-spin/OpenNMT-APE`

Table 1: Task 1 corpora and training iterations

Submission	Corpora	Iterations
PJA_CLARIN_1k	Clarin-PL	1 thousand
PJA_CLARIN_10k	Clarin-PL	10 thousand
PJA_CLARIN_20k	Clarin-PL	20 thousand
PJA_CLARIN_40k	Clarin-PL	40 thousand
PJA_CLARIN_50k	Clarin-PL	50 thousand
CLARIN_SEJM_40k	Clarin-PL and Polish parliament	40 thousand
CLARIN_SEJM_50k	Clarin-PL and Polish parliament	50 thousand

The whole process had three principal steps: pre-processing, training and translation, and two additional steps including the clean-up of missing lines and reprocessing of missed lines. In the pre-processing step model dimensionality was configured with 12 self-attention layers, 12 attention head, the RNN and word vector size of 768, and feed-forward inner layer of 3072.

Correia and Martins (2019) also suggest the use of Moses tokenizer in the pre-processing stage; however, this method was not used.

Interestingly, unlike in the previous approaches APE was not used for machine translation post-editing, but it was applied to refine the transcriptions of Polish language, hence improving the quality of ASR dedicated for Polish language (Ziółko et al. 2011).

4. Evaluation

The evaluation process has been performed with the use of Word Error Rate (WER), which measures the number of substitutions, deletions and insertions in the hypothesis utterance, divided by the length of the reference utterance (Popović and Ney 2007).

$$WER = \frac{N_{del} + N_{sub} + N_{ins}}{N_{ref}}$$

N_{del} refers to the number of deletions, N_{sub} refers to the number of substitutions, N_{ins} refers to the number of insertions, and N_{ref} refers to the reference length.

WER was calculated using NIST SCLITE (Score lite) package which is part of the Scoring Toolkit developed by the National Institute of Standards and Technology (NIST).[2] Reference data was generated by the same ASR system as the training data.

5. Results

Table 2 below presents the results obtained by each of the submissions made. Word Error Rate score of the ASR system used was 27.6%, hence any result below this score would indicate

[2] `https://www.nist.gov/about-nist`

a successful solution to the question of refining ASR output. Additionally, *Changes* column shows the percentage of changes made by the submission to the original transcription.

Table 2: Submission results

Submission	**WER %**	**Changes %**
PJA_CLARIN_1k	33.5	9.1
PJA_CLARIN_10k	32.0	9.6
PJA_CLARIN_20k	31.8	9.9
PJA_CLARIN_40k	31.8	10.3
PJA_CLARIN_50k	31.8	10.2
CLARIN_SEJM_40k	33.7	19.1
CLARIN_SEJM_50k	32.5	17.7

As can be observed in Table 2, the best score was achieved with the use of Clarin-PL dataset and 20 thousand iterations – 31.8 WER. The use of additional iterations (respectively 40 and 50 thousand) did change the original output yet did not improve the WER score. The use of combined dataset (Clarin-PL and Polish Parliament corpora) had an even higher percentage of change made to the original, yet again the WER score was not satisfactory. Although, it is worth noting that 50 thousand iterations improved the result, compared to 40 thousand iterations.

6. Conclusions

We have presented our submission to the PolEval 2020 contest for Task 1. With the use of open source APE system, the optimal score of 31.8% WER was achieved. It falls behind the average word error rate; nevertheless, the system applied has great potential thanks to the novelty application of encoder-decoder architecture with BERT language model (Lopes et al. 2019), and it would be worth experimenting further to optimize the parameters, in order to refine the results. Nonetheless, it adds a new perspective to the question of automatic post-editing models applied to the natural language understanding systems and sets an interesting path for further research.

References

Correia G. M. and Martins A. F. T. (2019). *A Simple and Effective Approach to Automatic Post-Editing with Transfer Learning*. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 3050–3056. Association for Computational Linguistics.

Devlin J., Chang M.-W., Lee K. and Toutanova K. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. arXiv:1810.04805.

Klein G., Kim Y., Deng Y., Senellart J. and Rush A. (2017). *OpenNMT: Open-Source Toolkit for Neural Machine Translation*. In *Proceedings of ACL 2017, System Demonstrations*, pp. 67–72. Association for Computational Linguistics.

Levis J. and Suvorov R. (2012). *Automatic Speech Recognition*. In Chapelle C. A. (ed.), *The Encyclopedia of Applied Linguistics*. Blackwell Publishing Ltd.

Lopes A. V., Farajian M. A., Correia G. M., Trenous J. and Martins A. F. (2019). *Unbabel's Submission to the WMT2019 APE Shared Task: BERT-based Encoder-Decoder for Automatic Post-Editing*. arXiv:1905.13068.

Marasek K., Koržinek D., Brocki Ł. and Jankowska-Lorek K. (2015). *Clarin-PL Studio Corpus (EMU)*. CLARIN-PL digital repository, `http://hdl.handle.net/11321/236`.

Negri M., Turchi M., Bertoldi N. and Federico M. (2018). *Online Neural Automatic Post-editing for Neural Machine Translation*. In Cabrio E., Mazzei A. and Tamburini F. (eds.), *Proceedings of the 5th Italian Conference on Computational Linguistics (CLiC-it 2018)*, pp. 288–293.

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

Ogrodniczuk M. (2018). *Polish Parliamentary Corpus*. In Fišer D., Eskevich M. and de Jong F. (eds.), *Proceedings of the LREC 2018 Workshop* ParlaCLARIN: Creating and Using Parliamentary Corpora, pp. 15–19. European Language Resources Association.

Popović M. and Ney H. (2007). *Word Error Rates: Decomposition over POS Classes and Applications for Error Analysis*. In *Proceedings of the 2nd Workshop on Statistical Machine Translation (StatMT 2007)*, pp. 48–55. Association for Computational Linguistics.

Ziółko M., Gałka J., Ziółko B., Jadczyk T., Skurzok D. and Mąsior M. (2011). *Automatic Speech Recognition System Dedicated for Polish*. In *Proceedings of the 12th Annual Conference of the International Speech Communication Association (INTERSPEECH 2011)*, pp. 3315–3316.

Results of the PolEval 2020 Shared Task 2: Morphosyntactic Tagging of Middle, New and Modern Polish

Marcin Woliński, Witold Kieraś, Dorota Komosińska
(Institute of Computer Science, Polish Academy of Sciences)

Włodzimierz Gruszczyński
(Institute of Polish Language, Polish Academy of Sciences)

Abstract

In the paper we present the objectives, dataset, evaluation and results of the PolEval 2020 Shared Task 2 devoted to the morphosyntactic tagging of historical texts representing three periods of the development of Polish: Middle, New and Modern. Our dataset is spanned between early 17th and early 21st c., covering grammatical and lexical changes of the last four centuries of the history of Polish. Thus the data are much more diverse and allow for testing morphosyntactic taggers in a slightly new environment.

Keywords

morphosyntactic disambiguation, tagging, historical language processing

1. Introduction

Morphosyntactic disambiguation is one of the most classic NLP problems. For nearly ten years the development and evaluation of morphosyntactic taggers for Polish were focused on the same dataset, namely NKJP1M. Our shared task was aimed at providing an opportunity to build new systems or tune existing ones to test them in a slightly different environment of more diverse and less standardised historical data. Although the data may seem to be unusual and atypical for everyday applications of NLP tools, the best performing solutions may be deployed in a growing number of projects aimed at building historical corpora of various periods of Polish.

2. Task definition

The data for this year's task covers 400 years of the development of the Polish language. Text samples were drawn from three manually annotated corpora: KorBa — a corpus of the 17th and 18th century (Gruszczyński et al. 2013, Kieraś et al. 2017, Gruszczyński et al. 2020), a corpus of the 19th century (Kieraś and Woliński 2018), and 1M subcorpus of the National Corpus of Polish NKJP (Przepiórkowski et al. 2012). The corpora represent three different periods of development of Polish: Middle, New and Modern.

All the texts were marked using a historical tagset, which is similar to the tagset of Morfeusz SGJP (Woliński 2019) with some differences, e.g.:

— the gender system is reduced to three basic genders: masculine, feminine and neuter, however some masculine forms are marked for animacy distinction (*manim1* and *manim2*);

— there are three values for the number category: singular, dual (*Dwie żabie upragnione po polach biegały*), and plural;

— there are special flexemes *adjb*, *ppasb* and *pactb* for historical "short" (non-compound) forms of adjectives and participles (*równien*, *pogrzebion*, *pięknę*, *swoję*, *chcęcy*…);

— additional flexeme for past participle (*ppraet*) was introduced (*wyłysiały*, *przybyły*);

— separate flexeme classes of so called adjectival and adverbial numerals (*adjnum* and *advnum*) were introduced;

— there are two flexemes *fut* and *plusq* for auxiliary forms of BYĆ for future and pluperfect tenses.

The tagset used in the task isn't native for any of the source corpora sampled for the shared task. It is rather a blend of all source tagsets, which was primarily developed for the purpose of Chronofleks web application (Woliński and Kieraś 2020) aimed at visualization of Polish inflectional phenomena over time.[1]

The goal of Task 2 is to disambiguate morphosyntactic interpretations and to guess the interpretation for unknown words — exactly as in Subtask 1A of PolEval 2017 (Kobyliński and Ogrodniczuk 2017). The text provided as input for taggers is represented as a directed acyclic graph of morphosyntactic interpretations, as returned by Morfeusz.

What we find interesting in this task is that the texts are not homogenous since the language changes. In fact, 17th century texts can be considered to represent a different (yet closely related) language than contemporary Polish. Thus, information on the date of creation was provided for each text.

[1] `http://chronofleks.nlp.ipipan.waw.pl`

3. The data

Proportions of the data sampled from the source corpora vary in training, development and testing data. In training data the proportions are as follows: aprox. 28% represent the Baroque corpus, 63% represent the 19th century corpus and 29% represent contemporary data (NKJP1M). However, the development and testing sets are more biased towards the oldest data in the overall dataset with 50% from the Baroque corpus, 30% from the 19th century corpus and only 20% from NKJP1M. The test set was used for scoring presented taggers, while the development set is meant to provide a preview of what to expect from the test set. Both are guaranteed to have similar distribution of texts in time. The training set contains the data intended for learning. In this set we provide as much data from each period as we have available. The details of the data split are presented in Table 1.

Table 1: Data split between three periods of time for train, devel and test datasets

	train		**devel**		**test**	
	# segm.	**%**	**# segm.**	**%**	**# segm.**	**%**
Baroque	408 248	28.32%	20 005	49.99%	20 026	50.00%
19th century	613 914	42.59%	12 001	29.99%	12 001	29.97%
Contemporary	419 346	29.08%	8 010	20.02%	8 018	20.03%
Total	1 441 508	100.00%	40 016	100.00%	40 045	100.00%

Motivations for such data split is straightforward – annotation of historical data is much more time and labour consuming and requires rarer skills from the annotators comparing to the contemporary Polish. The same applies to automatic morphosyntactic tagging (or disambiguation). Yet the growing number of projects aimed at building historical corpora cause a constant need for more accurate tagging systems. Thus in the shared task our goal was to encourage researchers to improve their systems towards tagging more diverse historical language while taking advantage of the more resource-rich contemporary Polish.

Each file available at the task page[2] corresponds to a particular text from one of the corpora. The first line of the file contains a time marker for the text. It may be a single number denoting the year on which the text was written (e.g. `#1651`) or a range (e.g. `#1651:1675`, meaning that the text was written between 1651 and 1675). Sometimes only the lower limit of the range is known (e.g. `#1651:` meaning after 1651). A file may contain several text samples separated by empty lines.

Each line contains one interpretation for a segment in 7 column format:

— start position for the segment,

— end position for the segment,

— the segment,

— lemma for the corresponding lexeme,

[2] `http://2020.poleval.pl/tasks/task2/`

— morphosyntactic tag,

— the string `nps` if there is no preceding space,

— the string `disamb` if this is the correct interpretation selected among variants provided by the morphological analyzer or `disamb_manual` if the corrected interpretation was added by a human. In the case of manually added segmentation variants all added segments are marked as "manual" even if some of them could be recognised by the analyser in other contexts.

Each dataset exists in two variants. In the "disamb" variant exactly one interpretation for each segment is marked as correct (in 7th column). The "plain" variant has this column removed together with all manual interpretations and segmentation variants. The train and devel data sets are provided in both variants. The test set was provided only in the plain variant during the competition and participants were expected to send the results of processing these files. Evaluation against an undisclosed "disamb" variant was performed by the organisers.[3]

Example of a manual interpretation in both variants:

```
--- in disamb variant ---
 36  37 inaczy inaczy  adv                disamb_manual
 36  37 inaczy inaczyć fin:sg:ter:imperf

--- in plain variant ---
 36  37 inaczy inaczyć fin:sg:ter:imperf
```

Example of a manual segmentation in both variants:

```
--- in disamb variant ---
271 272 więtszy  więtszy  adj:sg:nom:m:pos
271 272 więtszy  więtszy  adj:sg:voc:m:pos
271 273 więtszym więtszy  adj:sg:inst:m:pos           disamb_manual
271 273 więtszym więtszym ign
272 273 m        być      aglt:sg:pri:imperf:nwok nps

--- in plain variant ---
271 272 więtszy  więtszy  adj:sg:nom:m:pos
271 272 więtszy  więtszy  adj:sg:voc:m:pos
272 273 m        być      aglt:sg:pri:imperf:nwok nps
```

Besides the data provided by us, the participants were free to use any auxiliary data available to the public and released on an open license.

[3] The gold standard for test part was made public after the competition.

4. Scoring

A tagger is expected to split the text into a continuous stream of tokens and provide single morphological interpretation for each token. The solutions were scored against a gold standard corpus, which provides its own stream of tokens and interpretations. The tagger's answer for a given gold standard token is considered correct only if it contains a token with the same span in the text. The interpretation for this token will be assessed correct if segment and tag (columns 3, 5) are the same as in the corresponding interpretation marked as gold standard. This means that the choice of the lemma is not scored (and in fact most taggers did not produce any meaningful lemmas).

Please note that this setup means that the tagger is expected to provide exactly one interpretation for each token, otherwise the solution is rejected by the evaluation script. In other words, the tagger is not allowed to leave any tokens without an interpretation nor to allow tokens to remain ambiguous by marking more than one possibility (some contemporary taggers do that).

Overall accuracy against gold standard corpus was assumed as the evaluation metric for the contest. In the following section we show also some interesting partial marks: accuracy achieved on tokens which are known to the morphological analyser, accuracy on unknown ("ign") tokens, and accuracy on tokens which are known to the analyser, yet annotators decided to provide a different interpretation in the gold standard corpus. This last case is often not taken into account by contemporary taggers as marginal (0.25% in NKJP1M). In historical texts this situation is more common (1.45% of test data in this task).

Some of the taggers do not use the provided morphological graph at all nor do they work by choosing from provided interpretations. Yet, differences between these dictionary based categories are clearly visible.

5. Results

Eight solutions for this task were submitted by four teams representing both academic and industrial research groups. Three variants of the KFTT tagger were construed by Krzysztof Wróbel (Jagiellonian University). Two models disguised under the name of „Simple baselines" are a work of Piotr Rybak and Agnieszka Ciepielewska (Allegro.pl, Melodice.org). The author of "CMC Graph Heuristics" is Wiktor Walentynowicz (Wrocław University of Science and Technology). Alium was submitted by Marcin Bodziak (no affiliation given).

The results achieved (Table 2) are far better than we anticipated. Tagging of historical Polish can be expected to be more difficult than tagging contemporary language: the tagset includes more features, some of them describing very rare phenomena; the number of tokens unknown to the morphological analyser is larger (2.25% vs. 1.26%); the word order is less stable (with many discontinuous constructions). Yet, the results compare favourably to the results of PolEval 2017 Task 1(A) for contemporary language.[4] The best overall accuracy is 95.7%

[4] `http://2017.poleval.pl/index.php/results/`

Table 2: Competition results for Task 2 sorted by overall accuracy

System	Accuracy			
	overall	**on known**	**on unknown**	**manual not ign**
KFTT train+devel	**0.9573**	**0.9607**	0.8102	0.6781
KFTT train	0.9564	0.9600	0.7991	0.6661
KFTT train+devel wo_morf	0.9563	0.9595	**0.8191**	0.6730
Simple Baselines: XLM-R	0.9499	0.9562	0.6770	**0.6850**
Simple Baselines: COMBO	0.9284	0.9363	0.5838	0.5232
CMC Graph Heuristics	0.9121	0.9214	0.5072	0.1670
Alium-1000	0.8880	0.8985	0.4306	0.2427
Alium-1.25	0.8880	0.8985	0.4295	0.2427

compared to 94.6% of PolEval 2017. The most striking improvement lays in tagging tokens unknown to the morphological analyser: 81.9% compared to 67% in PolEval 2017.

Table 3 shows accuracies of each solution on parts of the test set drawn from respective historical periods. As can be seen, variants of KFTT win in most categories. In overall accuracy "KFTT train+devel" wins on texts from all periods. However, the variant not using tokenisation provided by Morfeusz (`wo_morf`) has better results in guessing tags for unknown tokens of historical text.

KFTT and "Simple baselines: XLM-R" show rather similar results, which is understandable, since both are based on the XLM-R model. The two solutions differ mainly in the method for tokenisation (see authors' texts in this volume). The difference in results is the largest for oldest texts. KFTT is also much better at guessing unknown tags. Strikingly, "Simple baselines: XLM-R" is better at guessing out-of-dictionary tags for known tokens. There is one category in which "XML-R" actually won: accuracy on known tokens of contemporary text (but the difference is just one more segment done correctly by "XML-R").

A very thrilling element in Table 3 is the accuracy over 0.97 for contemporary texts both in the case of KFTT and "Simple baselines: XLM-R". The contemporary part of the test set is admittedly tiny, yet we see here definite progress in tagging contemporary Polish.

6. Conclusions

These results require a further study, which will hopefully lead to interesting discussions during the PolEval 2020 conference session, but generally we can conclude that the presented systems not only improve on tagging historical texts, but provide better taggers also for contemporary Polish, which is a substantial achievement. The best performing systems have crossed the barrier of 97% accuracy for contemporary data which leaves very little (if any) room for further improvement and leads us to conclusion that the problem of morphosyntactic tagging for the contemporary Polish language may be in fact successfully solved at last.

Table 3: Results by historical period

System	Period	Accuracy			
		overall	**known**	**unknown**	**manual not ign**
KFTT train+devel	Baroque	**0.9435**	**0.9483**	0.7943	0.6574
	19th c.	**0.9694**	0.9715	0.8324	0.6316
	Contemp.	**0.9737**	0.9748	**0.8778**	**0.8043**
KFTT train	Baroque	0.9422	0.947	0.7943	0.6574
	19th c.	0.9692	**0.9718**	0.7933	0.5789
	Contemp.	0.9727	0.9741	0.8444	0.7609
KFTT train+devel wo_morf	Baroque	0.942	0.9464	**0.8054**	0.6505
	19th c.	0.969	0.9709	**0.8436**	0.6316
	Contemp.	0.9733	0.9745	0.8667	**0.8043**
Simple baselines: XLM-R	Baroque	0.9302	0.9392	0.6535	**0.6620**
	19th c.	0.9672	0.9715	0.6816	**0.6667**
	Contemp.	0.9733	**0.9749**	0.8333	**0.8043**
Simple baselines: COMBO	Baroque	0.9091	0.92	0.5744	0.5139
	19th c.	0.9452	0.951	0.5587	0.4561
	Contemp.	0.9514	0.9542	0.7	0.6087
CMC Graph Heuristics	Baroque	0.8837	0.8973	0.4636	0.1806
	19th c.	0.9354	0.9412	0.5531	0.01754
	Contemp.	0.9482	0.9508	0.7222	0.1957
Alium-1000	Baroque	0.8659	0.8804	0.4209	0.2616
	19th c.	0.9087	0.9159	0.4302	0.1404
	Contemp.	0.9123	0.917	0.5	0.2174
Alium-1.25	Baroque	0.8662	0.8804	0.4304	0.2616
	19th c.	0.9085	0.9159	0.419	0.1404
	Contemp.	0.9117	0.917	0.4444	0.2174

Acknowledgements

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

References

Gruszczyński W., Adamiec D. and Ogrodniczuk M. (2013). *Elektroniczny korpus tekstów polskich z XVII i XVIII w. (do 1772 r.) — prezentacja projektu badawczego*. „Polonica", XXXIII, p. 309–316.

Gruszczyński W., Adamiec D., Bronikowska R. and Wieczorek A. (2020). *Elektroniczny Korpus Tekstów Polskich z XVII i XVIII w. – problemy teoretyczne i warsztatowe*. „Poradnik Językowy", 8, p. 32–51.

Kieraś W. and Woliński M. (2018). *Manually Annotated Corpus of Polish Texts Published between 1830 and 1918*. 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. 3854–3859. European Language Resources Association.

Kieraś W., Komosińska D., Modrzejewski E. and Woliński M. (2017). *Morphosyntactic Annotation of Historical Texts. The Making of the Baroque Corpus of Polish*. In Ekštein K. and Matoušek V. (eds.), *Proceedings of the 20th International Conference Text, Speech, and Dialogue (TSD 2017)*, number 10415 in Lecture Notes in Computer Science, pp. 308–316. Springer International Publishing.

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.

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

Woliński M. (2019). *Automatyczna analiza składnikowa języka polskiego*. Wydawnictwa Uniwersytetu Warszawskiego.

Woliński M. and Kieraś W. (2020). *Analiza fleksyjna tekstów historycznych i zmienność fleksji polskiej z perspektywy danych korpusowych*. „Poradnik Językowy", 8, p. 66–80.

KFTT: Polish Full Neural Morphosyntactic Tagger

Krzysztof Wróbel
(Department of Computational Linguistics, Jagiellonian University)

Abstract

This paper presents winning solution to PolEval 2020[1] morphosyntactic tagging of Middle, New and Modern Polish task. The goal of the task is to disambiguate morphologic analysis. The solution has a full neural network pipeline (tokenization and morphosyntactic tagging) from raw text to annotated text. It does not require any external dependencies. However, the output from morphological analyzer can be exploited to increase the scores. Finally, the tagger exceeds the threshold of 97% obtaining the score of 97.3% for contemporary texts.

Keywords

natural language processing, part-of-speech tagging, tokenization, Polish

1. Introduction

Manning (2011) reports that the state-of-the-art part-of-speech taggers for English have obtained 97.3% of accuracy, which is similar to a human inter-annotator agreement (97%). In 2016, for Polish taggers "reaching the goal of 97% seemed very distant" (Kobyliński and Kieraś 2016). However, the next year thanks to PolEval competition (Kobyliński and Ogrodniczuk 2017) new deep learning approaches arose reaching 94% of accuracy (Krasnowska-Kieraś 2017, Wróbel 2017). Kobyliński et al. (2018) used meta-algorithm to achieve 94.7%.

The above scores are not directly comparable because of different procedures of preparing training and test data and the corpus itself. English POS tagging problem is simpler because there are several dozen tags and in Polish about 1000 is used.

The lemmatization problem was not properly tackled so far. The most common solution is to randomly pick a lemma from interpretations from a morphological analyzer consistent with the predicted tag. KRNNT (Wróbel 2017) improves this process by learning the most common lemma for text form and tag pair.

[1] `http://2020.poleval.pl`

Most of the taggers tokenize text by finding the shortest path in the output of the morphological analyzer. Waszczuk et al. (2018) solve this problem by using a conditional random field directly on directed acyclic graphs (DAG) representing possible tokenizations.

In this work, finally, we exceeded the score of 97% accuracy for modern texts. The tokenization is tackled by a character language model and recurrent neural networks (RNN) allowing scores higher than oracle on DAGs. The tagging process is performed by a multilingual transformer model XLM-RoBERTa (Conneau et al. 2020).

Trained models and code needed for result reproduction are available online.[2]

2. Data

The manually annotated corpus was created by sampling three corpora representing three periods of development of Polish (Middle, New, and Modern): KorBa — a corpus of 17th and 18th century, a corpus of 19th century, and 1M subcorpus of the National Corpus of Polish.

The texts were annotated using a historical tagset similar to Morfeusz SGJP (Woliński 2014, Kieraś et al. 2017). The texts are represented as directed acyclic graphs of interpretations, as returned by Morfeusz. Each text is also annotated by the date of creation.

In comparison to other corpora, e.g. National Corpus of Polish (NKJP), texts are not split into sentences. Table 1 presents the number of texts, tokens, the average number of tokens in texts, and the number of unique tags. NKJP has a similar number of unique tags: 926. Table 2 shows the distribution of tokens regarding time in training, development, and testing dataset. The development and testing datasets have a similar distribution of texts in time.

Table 1: Number of texts, tokens, the average number of tokens in texts, and the number of unique tags for training, development, and test data

	train	**dev**	**test**
number of texts	10 755	244	280
number of tokens	1 441 508	40 016	40 045
average number of tokens in text	134	164	143
unique tags	994	571	582

Table 2: Distribution of texts by time in training, development, and test data

Subcorpus	**Period**	**train**	**dev**	**test**
KorBa — a corpus of 17th and 18th century	Middle	28.3%	50.0%	50.0%
A corpus of 19th century	New	42.6%	30.0%	30.0%
1M subcorpus of the National Corpus of Polish	Modern	29.1%	20.0%	20.0%

[2] `https://github.com/kwrobel-nlp/kftt/`

3. Methods

The solution consists of two separate modules: tokenization and tagging. Tokenization can be performed on raw texts (without information from morphological analysis) or using a graph of interpretations from Morfeusz.

The tokenization module has been implemented using recurrent neural networks operating on characters. The first layer is a character-based language model working forward and backward using RNNs. The second layer is bidirectional RNN with a conditional random field (CRF) on top. The network answers a question if after every character should be the end of the token.

In the first version, on the input of the tokenization network are only characters. The second version exploits information from Morfeusz by appending to each character additional information, i.e. potential end of token, potential tags, and time of creation.

Table 3 presents output from Morfeusz with Baroque dictionary for word *zaś*. It can be tokenized as one token *zaś* or two tokens *za* and *ś*. Table 4 shows features generated for each character of this word. For example for character *ś* the `joined tags` feature is constructed from tags of every segment ending with this character: *zaś* and *ś*.

Table 3: Output from Morfeusz with Baroque dictionary for word *zaś*

start	end	segment	lemma	tag
1	2	za	za	`part`
1	3	zaś	zaś	`conj`
1	3	zaś	zaś	`part`
2	3	ś	być	`aglt:sg:sec:imperf:nwok nps`

Table 4: Additional features generated for characters in word *zaś*

Features	z	a	ś
is space before	True	False	False
joined tags	—	part	`aglt:sg:sec:imperf:nwok_conj_part`
joined POS	—	part	`aglt_conj_part`
century	17	17	17
is ambiguous	False	True	False

The tagging module operates on tokenized texts. It is a transformer model with a token classification head on top. The transformer returns contextual embedding of each token, then a linear layer with softmax activation returns normalized scores for each tag seen in training.

4. Evaluation

Tokenization is evaluated using similar metrics as tagging. E.g. not splitting *zaś* to two tokens *za* and *ś* generates one false positive and two false negatives. Recall here is accuracy in tagging, so tagger accuracy cannot be higher than tokenization recall.

The metrics for tagging are consistent with Radziszewski and Acedański (2012) with Accuracy (accuracy lower bound), Acc on known (accuracy lower bound for known words), Acc on unknown (accuracy lower bound for unknown words). The main metric in the competition is accuracy – a percentage of all tokens that match tagger segmentation with the correct tag. The accuracy is also provided for known and unknown tokens for a morphological analyzer. Additionally, the organizers report Acc on manual – accuracy for manually tokenized words and manually appended correct interpretations to interpretations from the analyzer.

5. Experiments

The training was performed using only data provided by organizers.

The tokenization module uses Flair embeddings (Akbik et al. 2018). The training lasts 24 hours on GPU Tesla V100 with a learning rate of 0.1 and a hidden size of RNN 256.

For the tagging module, the transformer model has been chosen as a multi-language XLM-RoBERTa large version as it is one of the best models as stated in a leaderboard of KLEJ Benchmark[3] (Rybak et al. 2020) – a set of nine evaluation tasks for the Polish language understanding. The model was fine-tuned for 20 epochs using learning rate 5e-5, maximum sequence length 512, max gradient norm 1.0, without warmup steps. The training takes 4 hours using GPU Tesla V100. Two versions were trained: using only training data (`train`) and using training and development data (`train+devel`).

Table 5 presents precision, recall and F_1 for tokenization. The proposed solution obtains better scores than the shortest path method, which is used in most of the taggers. In terms of recall and F_1, the solution is also better than oracle on Morfeusz analysis. The oracle is the best path in a DAG. From a practical point of view, the new solution deals correctly with frequent word *miałem* (eng. *I had*).

Table 5: Scores of two tokenization modules compared with shortest path strategy and oracle (the best path)

Method	Precision	Recall	F_1
with morf	99.74%	**99.76%**	**99.75%**
without morf	99.72%	99.67%	99.70%
shortest path	99.48%	99.23%	99.35%
oracle	**99.83%**	99.63%	99.73%

[3] `https://klejbenchmark.com/leaderboard/`

Table 6 shows possible interpretations where shortest path tokenization method forces rare interpretation *miałem* (eng. *coal dust*).

Table 6: Output from Morfeusz for word *miałem*

start	end	segment	lemma	tag
1	2	miał	mieć	`praet:sg:m1.m2.m3:imperf`
1	3	miałem	miał	`subst:sg:inst:m3`
2	3	em	być	`aglt:sg:pri:imperf:wok`

The test data is tokenized in 14s and tagged in 16s using GPU Tesla V100 (time for tagging is measured using batch size 1, so can be easily optimized). The full pipeline for KRNNT takes 18.4s using only CPU i7-7700HQ.

Table 7 shows official results for top 5 submissions. KFTT obtains the highest accuracy 95.7%. By using more training data the score increases by 0.1 percentage point (`train+devel` versus `train`). KFTT version that works on raw text without using morphological analysis (`wo_morf`) decrease accuracy by 0.1 percentage point. This version has the highest score on unknown words – because the tokenizer has no information whether a segment is known for analyzer and model did not focus on them.

Table 7: Official results for the top 5 submissions

System	Accuracy			
	overall	known	unknown	manual not ign
KFTT train+devel	**95.73%**	**96.07%**	81.02%	67.81%
KFTT train	95.64%	96.00%	79.91%	66.61%
KFTT train+devel wo_morf	95.63%	95.95%	**81.91%**	67.30%
Simple Baselines: XLM-R	94.99%	95.62%	67.70%	**68.50%**
Simple Baseline: COMBO	92.84%	93.63%	58.38%	52.32%

Table 8 presents the scores by the period of text creations. We can finally announce that there is a tagger that exceeded the threshold of 97% for contemporary Polish. The worst results are for Middle period Polish development. Unfortunately, the feature related to the time of text creation has not improved the tagging module.

Table 8: KFTT train+devel scores for each period

Period	Accuracy			
	overall	known	unknown	manual
Middle	94.35%	94.83%	79.43%	73.87%
New	96.94%	97.15%	83.24%	78.39%
Modern	97.37%	97.48%	87.78%	84.07%

6. Conclusions

The paper presents state-of-the-art morphosyntactic tagger for Polish, the winner of PolEval 2020. It exceeds the threshold of 97% for contemporary Polish. The full neural network version (which accuracy drops by 0.1 percentage point) does not require any dependencies so it is easy to run on computer clusters.

In comparison to KRNNT, KFTT uses contextual embeddings, training and prediction are made not on separate sentences, so the context is much wider, tokenization is trainable, there are no dependencies to external libraries.

The future works may focus on using other transformer models, e.g. Polish RoBERTa (Dadas et al. 2020), which is adjusted to the Polish language and should perform faster. A transformer model may be further tuned on historical texts. Information from a morphological analyzer could be exploited also in the tagging module. A transformer model could be fine-tuned using other annotated corpora.

Acknowledgements

This research was supported in part by PLGrid Infrastructure.

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. Association for Computational Linguistics.

Conneau A., Khandelwal K., Goyal N., Chaudhary V., Wenzek G., Guzmán F., Grave E., Ott M., Zettlemoyer L. and Stoyanov V. (2020). *Unsupervised Cross-lingual Representation Learning at Scale*. „Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics".

Dadas S., Perełkiewicz M. and Poświata R. (2020). *Pre-training Polish Transformer-based Language Models at Scale*. arXiv:2006.04229.

Kieraś W., Komosińska D., Modrzejewski E. and Woliński M. (2017). *Morphosyntactic Annotation of Historical Texts. The Making of the Baroque Corpus of Polish*. In Ekštein K. and Matoušek V. (eds.), *Proceedings of the 20th International Conference Text, Speech, and Dialogue (TSD 2017)*, vol. 10415 of *Lecture Notes in Computer Science*, pp. 308–316. Springer International Publishing.

Kobyliński Ł. and Kieraś W. (2016). *Part of Speech Tagging for Polish: State of the Art and Future Perspectives*. In *Proceedings of the 17th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing 2016)*, pp. 307–319.

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.

Kobyliński Ł., Wasiluk M. and Wojdyga G. (2018). *Improving Part-of-Speech Tagging by Meta-Learning*. In Sojka P., Horák A., Kopeček I. and Pala K. (eds.), *Proceedings of 21st International Conference Text, Speech, and Dialogue (TSD 2018)*, vol. 11107 of *Lecture Notes in Artificial Intelligence*, pp. 144–152. Springer-Verlag.

Krasnowska-Kieraś K. (2017). *Morphosyntactic Disambiguation for Polish with Bi-LSTM Neural Networks*. 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. 367–371. Fundacja Uniwersytetu im. Adama Mickiewicza w Poznaniu.

Manning C. D. (2011). *Part-of-Speech Tagging from 97% to 100%: Is It Time for Some Linguistics?* In Gelbukh A. F. (ed.), *Computational Linguistics and Intelligent Text Processing*, pp. 171–189. Springer Berlin Heidelberg.

Radziszewski A. and Acedański S. (2012). *Taggers Gonna Tag: An Argument against Evaluating Disambiguation Capacities of Morphosyntactic Taggers*. In Sojka P., Horák A., Kopeček I. and Pala K. (eds.), *Proceedings of the 15th International Conference on Text, Speech and Dialogue (TSD 2012)*, pp. 81–87. Springer Berlin Heidelberg.

Rybak P., Mroczkowski R., Tracz J. and Gawlik I. (2020). *KLEJ: Comprehensive Benchmark for Polish Language Understanding*. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 1191–1201. Association for Computational Linguistics.

Waszczuk J., Kieraś W. and Woliński M. (2018). *Morphosyntactic Disambiguation and Segmentation for Historical Polish with Graph-Based Conditional Random Fields*. In Sojka P., Horák A., Kopeček I. and Pala K. (eds.), *Proceedings of 21st International Conference Text, Speech, and Dialogue (TSD 2018)*, vol. 11107 of *Lecture Notes in Artificial Intelligence*, pp. 188–196. Springer-Verlag.

Woliński M. (2014). *Morfeusz reloaded*. In Calzolari N., Choukri K., Declerck T., Loftsson H., Maegaard B., Mariani J., Moreno A., Odijk J. and Piperidis S. (eds.), *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC 2014)*, pp. 1106–1111. 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.

Simple yet Effective Baseline for Morphosyntactic Tagging of Middle, New and Modern Polish

Piotr Rybak (ML Research at Allegro.pl)

Agnieszka Rybak (Melodice.org)

Abstract

This paper presents our solution of the PolEval 2020 task on Morphosyntactic Tagging of Middle, New and Modern Polish. The unique challenge posed by this task was to use the additional information about the year in which given text was written. We evaluated two approaches on how to tackle this problem. Additionally, we trained a BERT-based classifier which proved to perform much better than a solution based on LSTM networks and it ranked second in the official evaluation obtaining accuracy of 94.99%.

Keywords

natural language processing, tagging, pretrained language models, XLM-RoBERTa, COMBO

1. Introduction

The morphosyntactic tagging is one of the most classic and extensively researched tasks in Natural Language Processing (NLP). Over the years many different methods were used to solve it. Currently, the best results for English are achieved by the BERT-based (Devlin et al. 2019) models which use extensive pretraining to improve their quality.

In this work, we present our solution to the PolEval 2020 task on Morphosyntactic Tagging of Middle, New and Modern Polish. The rest of the paper is organized as follows. In Section 2, we describe our approach to solving the task. In Section 3, we conduct the evaluation of our method and finally we conclude our work in Section 7.

2. Method

The proposed task differs from a typical part-of-speech tagging. The goal is to find the correct interpretation of how given word should be tokenized into segments and then to assign the

correct morphosyntactic tag for each segment. To avoid building two separate models, we decided to transform the task into the classical part-of-speech tagging problem and use the off-the-shelf system to solve it.

2.1. Data processing

Data provided by organizers consists of a manually annotated subset of three corpora, a corpus of 17th and 18th century language (Kieraś et al. 2017), a corpus of 19th century language (Kieraś and Woliński 2018), and the National Corpus of Polish (Przepiórkowski et al. 2012). Each word is represented as its morphosyntactic interpretations, as returned by Morfeusz (Woliński 2014). Additionally, for most documents, the year when the text was written is provided.

To transform the morphosyntactic disambiguation task into morphosyntactic tagging we used the following approach. First, for each word we took all of its interpretations. Then, we found the minimal set of subwords from which it is possible to reconstruct all interpretations. For example the word `ABCD` with possible interpretations `[(A, BCD), (ABC, D)]` will be transformed into `[A, BC, D]` subwords. Finally, morphosyntactic tag for each subword was taken from its true interpretation.

2.2. COMBO

As a first baseline, we used COMBO (Rybak and Wróblewska 2018) and train it with default parameters using the dataset described in the previous paragraph. We used the 256-dimensional hidden layer for the final classifier. Next, we evaluated two methods of including information about the year in which the given text was written. First, we encoded the decade in which the text was created as an 8-dimensional embedding and concatenated it with word embedding (later called Word-level Augmentation). Alternatively, we appended the decade in which the text was created to the document as an additional segment (later called Document-level Augmentation).

2.3. XLM-RoBERTa

According to the KLEJ Benchmark[1] (Rybak et al. 2020) the two top-performing models for Polish language understanding are Polish RoBERTa (Dadas et al. 2020) and XLM-RoBERTa (Conneau et al. 2020). We used the latter and trained the tagger released within the `transformer` library (Wolf et al. 2019). We kept default parameters, except for number of epochs which we set to 50.

[1] `https://klejbenchmark.com/leaderboard/`

Table 1: Evaluation results of described models on a validation set. We used accuracy as an evaluation metric. The best scores are in bold.

Model	Accuracy			
	overall	**known**	**unknown**	**manual not ign**
COMBO	92.82	93.59	56.87	57.66
COMBO + Word-level Augmentation	93.08	93.88	55.67	57.52
COMBO + Document-level Augmentation	92.99	93.78	55.79	57.80
XLM-RoBERTa	**94.58**	**94.84**	**82.68**	**77.90**

2.4. Postprocessing

During the inference, we needed to transform the results of the tagger back to the morphosyntactic interpretations. We used simple heuristic and merged subsequent subwords with the same predicted tag into a single segment. We didn't verify if the resulting interpretation was one of the originally returned by Morfeusz.

3. Experiments

We evaluated all proposed models on the validation dataset. The results are summarized in Table 4. Both Word- and Document-level Augmentation increased the overall performance of the models. Interestingly, the improvement in only present for words which are known by the morphological analyser. The Word-level Augmentation obtained better scores than Document-level Augmentation (93.08 vs 92.99).

The XLM-RoBERTa model outperformed the COMBO model by a wide margin (94.58 vs 92.82). It was better for both known and unknown words. The difference is especially large for the latter (82.68 vs 56.87).

4. Conclusions

In this work, we described two simple morphosyntactic disambiguation systems for Polish language. The system based on XLM-RoBERTa proved to be more effective and ranked second in the official evaluation. We also evaluated two methods of including text creation year and showed its positive impact on tagger quality.

References

Conneau A., Khandelwal K., Goyal N., Chaudhary V., Wenzek G., Guzmán F., Grave E., Ott M., Zettlemoyer L. and Stoyanov V. (2020). *Unsupervised Cross-lingual Representation Learning at*

Scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 8440–8451. Association for Computational Linguistics.

Dadas S., Perełkiewicz M. and Poświata R. (2020). *Pre-training Polish Transformer-based Language Models at Scale*. arXiv:2006.04229.

Devlin J., Chang M.-W., Lee K. and Toutanova K. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186.

Kieraś W. and Woliński M. (2018). *Manually Annotated Corpus of Polish Texts Published between 1830 and 1918*. 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. 3854–3859. European Language Resources Association.

Kieraś W., Komosińska D., Modrzejewski E. and Woliński M. (2017). *Morphosyntactic Annotation of Historical Texts. The Making of the Baroque Corpus of Polish*. In Ekštein K. and Matoušek V. (eds.), *Proceedings of the 20th International Conference Text, Speech, and Dialogue (TSD 2017)*, vol. 10415 of *Lecture Notes in Computer Science*, pp. 308–316. Springer International Publishing.

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

Rybak P. and Wróblewska A. (2018). *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., Mroczkowski R., Tracz J. and Gawlik I. (2020). *KLEJ: Comprehensive Benchmark for Polish Language Understanding*. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 1191–1201. Association for Computational Linguistics.

Wolf T., Debut L., Sanh V., Chaumond J., Delangue C., Moi A., Cistac P., Rault T., Louf R., Funtowicz M., Davison J., Shleifer S., von Platen P., Ma C., Jernite Y., Plu J., Xu C., Scao T. L., Gugger S., Drame M., Lhoest Q. and Rush A. M. (2019). *HuggingFace's Transformers: State-of-the-art Natural Language Processing*. arXiv:1910.03771.

Woliński M. (2014). *Morfeusz Reloaded*. In Calzolari N., Choukri K., Declerck T., Loftsson H., Maegaard B., Mariani J., Moreno A., Odijk J. and Piperidis S. (eds.), *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC 2014)*, pp. 1106–1111. European Language Resources Association.

CMC Tagger in the Task of Tagging Historical Texts

Wiktor Walentynowicz, Tomasz Kot (Department of Computational Intelligence, Wrocław University of Science and Technology)

Abstract

In this paper, we present our approach to solving Task 2 from the PolEval 2020[1] competition. We briefly present the architecture used and the process of preparing the submitted solution. In the summary of the paper, we present our ideas for the development of this solution.

Keywords

natural language processing, computational linguistics, morphosyntactic tagging, neural networks

1. Introduction

At this year's edition of PolEval, Task 2 was "Morphosyntactic tagging of Middle, New and Modern Polish". Morphosyntactic tagging is one of the most fundamental tasks in NLP, mainly because many other tasks use the result of morphosyntactic disambiguation. Over the past few years, Polish morphosyntactic tagging has focused on the development of taggers for contemporary language. The main training and test dataset was the one million word subcorpus of the National Corpus of Polish (Przepiórkowski et al. 2012), further NKJP. This task has been extended to historical texts from the corpus KorBa (17th and 18th century; Kieraś et al. 2017) and the 19th-century Polish language corpus (Kieraś and Woliński 2018).

Changing the standard task of morphosyntactic disambiguation requires also the need to chose token segmentation. The role of the method solving the task is therefore both to indicate the correct morphological interpretation and to select the appropriate token segmentation, from those proposed in the data. The dataset is in the form of acyclic graphs, consistent with the standard from the Morfeusz2 (Woliński 2014) analyzer.

[1] `http://2020.poleval.pl`

2. CMC tagger

Our solution for the competition task was based on the CMC tagger (Walentynowicz et al. 2019). It combines the neural model based on multi-task learning, which is an input to a heuristic method of decision making from those proposed by the morphological analyzer. The code of our solution is available in the remote repository.[2]

2.1. Computational part

The calculation part of the CMC tagger can be divided into two modules: an input module and a processing module. In the input module, the sequence of tokens is changed into floating-point vectors, which are then processed by the processing module. In the input module, the token receives four representations, which are concatenated to be one vector. These representations are made up of *fastText* (Bojanowski et al. 2017) vector from KGR10 model[3] (Kocoń 2018), suffix character embedding vector, suffix embedding vector, and Brown cluster (Brown et al. 1992) representation vector.

2.2. Decision-making part

The decision-maker part has two roles – it checks the compliance with the tagset generated by the prediction network and makes the final tag selection. Tag validation is based on verifying the presence of attributes against the predicted grammar class. The superfluous attributes are set to *NONE* value. If the needed attribute does not have a value, no steps are taken as it will be selected from among the possible tags proposed by the analyzer. The corrected tag from the prediction is compared, based on the Levenshtein distance, to the tags given as a set of possible tags for the given token. The closest one is selected as the correct one.

2.3. Segmentation heuristics

We decided to take a heuristic approach to the task of selecting the token segmentation. Segmentation was about choosing the selection of paths in the graph representing the texts. We prepared heuristics, which work on simple principles. The first one was always choosing the shortest path in the graph, the second was choosing the longest path, and the third was based on the statistics from the training set. The statistical heuristics identified cases of ambiguity in tokenization, and then the decision was made based on the frequency of selected paths in a given case. In other words, the heuristics select those paths in tokenization that were more frequently encountered in the training set. If a given case did not occur in the training set, the longest possible path was selected.

[2] `https://gitlab.clarin-pl.eu/syntactic-tools/morphological/cmc-tagger/-/tree/cmc-heuristics`

[3] `http://hdl.handle.net/11321/606`

3. Experiments

During the experiments, we studied the effectiveness of the heuristics of segmentation selection and differences in the ways of teaching models. The models were trained in three variants: on specific period dataset only, on the whole dataset, and the trained model on all data for 7/12 of the learning time and fitted for 5/12 of the learning time on specific period data.

A summary of these experiments can be found in Tables 1–3. We do not include the results of statistical heuristic, because they came close to the results of long path heuristic, but always slightly worse. The measures were obtained using an evaluation script made available in the content of the task on the competition website. All models were trained on the training set and tested on the validation set according to the division given on the task website.

Table 1: Results for models with *short path* heuristic

Measure	**17**	**ALL_17**	**19**	**ALL_19**	**20**	**ALL_20**	**COMB**	**ALL_CMB**
Overall	82.96%	84.03%	92.19%	92.18%	92.81%	93.75%	87.70%	88.42%
Known	84.15%	85.07%	92.81%	92.76%	93.47%	94.38%	88.64%	89.27%
Unknown	40.84%	47.07%	49.41%	51.18%	49.59%	52.07%	43.85%	52.07%
Manual	29.97%	33.20%	41.67%	40.26%	38.96%	40.26%	33.31%	40.26%

Table 2: Results for models with *long path* heuristic

Measure	**17**	**ALL_17**	**19**	**ALL_19**	**20**	**ALL_20**	**COMB**	**ALL_CMB**
Overall	87.59%	88.68%	92.89%	92.94%	92.91%	*93.88%*	90.25%	91.00%
Known	88.90%	89.85%	93.52%	93.54%	93.57%	*94.52%*	91.24%	91.91%
Unknown	40.84%	47.07%	49.41%	51.18%	49.59%	*52.07%*	43.85%	48.63%
Manual	30.37%	33.50%	41.67%	44.17%	38.96%	*40.26%*	33.58%	40.26%

Table 3: Results for models with transfer learning

Measure	**Short 17**	**Long 17**	**Short 19**	**Long 19**	**Short 20**	**Long 20**
Overall	84.10%	*88.83%*	92.82%	*93.62%*	93.47%	93.63%
Known	85.14%	*90.00%*	93.41%	*94.22%*	94.09%	94.28%
Unknown	46.89%	*46.89%*	51.76%	*51.76%*	52.89%	52.89%
Manual	33.10%	*33.60%*	44.17%	*44.17%*	40.69%	40.69%

The *Overall* result refers to the categorical accuracy for all tokens. *Known* only for tokens that had possible morphological interpretations. *Unknown* is a measure for out-of-vocabulary tokens. *Manual* is a measure for tokens that lacked the correct morphological interpretation among those proposed by the analyzer.

In Tables 1 and 2, labels express the type of model and a subset of data on which it was trained and validated. *XX* – the model was trained and validated on a set of XX only. *ALL_XX* – the

model was trained on a whole dataset from all periods and validated on a set of XX. *COMB* is the result of a multitagger combined from models 17, 19, 20. *ALL_CMB* is the result of a tagger trained on all data and validated on the full set. In Table 3, the labels express the type of heuristics and the period of data on which the model was fitted and validated.

Models that have marked results by italic font have been selected as models forming a multitagger, which have generated results for the test set in the task. The results of this model obtained on the test set are presented in Table 4.

Table 4: Results for submission model in the test set

Measure	**Final multitagger**
Overall	91.21%
Known	92.14%
Unknown	50.72%
Manual	16.70%

4. Conclusions

Our proposed method presents the possibility of extending the architecture from a standard morphosyntactic tagging task to a combined tagging and segmentation task. However, the results are not satisfying. The most sensitive element is the heuristics of segmentation selection – they do not use contextual information. We plan to change this in a future version. The second element we want to add is a graph reanalysis module to improve the results among known words for which there was a lack of interpretation and connect a morphological guesser so that out-of-vocabulary words have preliminary morphological analysis proposals.

Acknowledgements

The work was co-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, p. 135–146.

Brown P. F., deSouza P. V., Mercer R. L., Pietra V. J. D. and Lai J. C. (1992). *Class-based N-gram Models of Natural Language*. „Comput. Linguist.”, 18(4), p. 467–479.

Kieraś W. and Woliński M. (2018). *Manually Annotated Corpus of Polish Texts Published between 1830 and 1918*. 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. 3854–3859. European Language Resources Association.

Kieraś W., Komosińska D., Modrzejewski E. and Woliński M. (2017). *Morphosyntactic Annotation of Historical Texts. The Making of the Baroque Corpus of Polish*. In Ekštein K. and Matoušek V. (eds.), *Proceedings of the 20th International Conference Text, Speech, and Dialogue (TSD 2017)*, vol. 10415 of *Lecture Notes in Computer Science*, pp. 308–316. Springer International Publishing.

Kocoń J. (2018). *KGR10 FastText Polish Word Embeddings*. CLARIN-PL digital repository. `http://hdl.handle.net/11321/606`.

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

Walentynowicz W., Piasecki M. and Oleksy M. (2019). *Tagger for Polish Computer Mediated Communication Texts*. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pp. 1295–1303.

Woliński M. (2014). *Morfeusz Reloaded*. In Calzolari N., Choukri K., Declerck T., Loftsson H., Maegaard B., Mariani J., Moreno A., Odijk J. and Piperidis S. (eds.), *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC 2014)*, pp. 1106–1111. European Language Resources Association.

Results of the PolEval 2020 Shared Task 3: Word Sense Disambiguation

Arkadiusz Janz, Joanna Chlebus, Agnieszka Dziob, Maciej Piasecki
(Wrocław University of Science and Technology)

Abstract

This paper reports on the results of the first edition of PolEval 2020 Shared Task focused on Word Sense Disambiguation (WSD). One of the main contributions of this shared task is the introduction of a new independent dataset prepared on the basis of an updated sense inventory compatible with plWordNet 3.2. A new wordnet-based sense inventory opens up a lot of opportunities for the methods that work in the open-domain setting where the texts usually are not limited to predefined domains and domain-specific knowledge. The design of our shared task follows mainly the style of the well-known SENSEVAL and SemEval competitions aiming at all-words WSD task. We present a general statistical view on available development and test data, describe the properties of a dedicated knowledge-base, and are officially announcing the results of the solutions submitted to the competition. We also highlight our future work on data unification for the task of WSD.

Keywords

natural language processing, word sense disambiguation, knowledge bases, wordnets, Polish language

1. Introduction

The Word Sense Disambiguation (WSD) has been proven to be an important part of Natural Language Processing (NLP) affecting many tasks, and especially in the area of computational semantics the lexical ambiguity problem. A word may express multiple lexical meanings (called also *word senses* or shortly *senses*) and these meanings can be either homonymous or polysemous. Lexical polysemy (Pustejovsky and Boguraev 1996) occurs when a word can be associated with multiple but semantically interrelated senses. On the other hand, homonymy is the accidental identity of word-forms with no semantic relatedness between senses. Homonyms have different etymology and should be treated as completely different

words despite having the same word form. Such a kind of ambiguity might be easier for automated disambiguation as homonymous meanings usually occur in significantly different linguistic contexts (e.g. frequently co-occurring words or topics).

Word Sense Disambiguation is still an open NLP problem mainly because of the lack of large-scale sense-annotated training corpora. The available training resources built for other languages (e.g. SemCor; Landes et al. 1998) have been usually accused of having scarce and imbalanced data to train a good supervised model mainly due to the most frequent sense bias. This directly arises from the natural sense distribution in textual corpora. On the other hand, the recent advances in the area of language modeling, especially with the help of deep neural networks and transfer learning suggest that there might be other solution to this problem.

Following the style and design of well-known SENSEVAL (Edmonds and Cotton 2001) and SemEval (Agirre et al. 2008) competitions we decided to propose the first shared task on Polish word sense disambiguation.

2. Previous research

WSD is often treated as a sequence classification problem solved by supervised Machine Learning techniques. However, as it was mentioned in the introduction, such an approach requires a heavy workload on handcrafting language resources, mainly sense-annotated corpora, in order to prepare a robust WSD system. The existing sense-annotated corpora do not usually cover less frequent senses which makes them less useful from the practical perspective.

In order to train a supervised WSD model hundreds of manually disambiguated training examples of word occurrences are needed for every single word sense. A typical corpus has a very imbalanced distribution of senses. Thus, it is very difficult to find usage examples of rare senses, even if we prepare a very large corpus.

In the last three decades, the researchers were looking for other solutions than supervised approaches trying to depart from the limitations of the latter. Such approaches as (Lesk 1986, Banerjee and Pedersen 2002, Agirre and Soroa 2009, Agirre et al. 2014, Moro et al. 2014) rely on linguistic knowledge sources like dictionaries or wordnets (lexical semantic networks) – also called *sense inventories* – that provide some insight about word senses and relations holding between them.

Many different Machine Learning algorithms with their problem-specific extensions have been already applied for WSD task. Decision tree (DT; Brown et al. 1991), decision list (DL; Yarowsky 1994), naïve Bayes classifier (NB; Gale et al. 1992) and k-nearest-neighbour algorithm (kNN; Ng and Lee 1996), or support vector machines (SVM; Lee and Ng 2002) were initially used to train supervised models (e.g. Baś et al. 2008).

With the recent advances of neural language modeling and transfer learning new neural models were successfully applied to WSD task for English language (Kågebäck and Salomonsson 2016, Raganato et al. 2017, Luo et al. 2018, Huang et al. 2019, Kumar et al. 2019).

3. Task description

After analysing the previous research and recent word sense disambiguation trends, we decided to propose two distinct variants of our shared task. The first variant has been called *Fixed Competition*. This type of competition addresses mainly the weakly supervised methods and knowledge-based approaches. The main idea was to encourage the existing NLP community to design new disambiguation methods that do not require labour-intensive manual sense annotation of textual corpora. The supervised models are usually strongly correlated with most frequent senses and express low vocabulary coverage. Alternatively, one may use available sense-focused knowledge bases e.g. wordnets as knowledge sources for designing new word sense disambiguation algorithms. Thus, the *Fixed Competition* variant might be helpful for low-resource languages especially those, for which it is possible to build a word sense knowledge-base by using existing wordnets and also interlingual links. In this variant we restricted the usage of the available knowledge sources mainly to wordnets and raw unstructured textual corpora as they are relatively easy to obtain. We also did not allow to use domain-specific sense annotated data except the data available in the Polish wordnet. Summing up, the participants could use only the following resources:

— *plWordNet* (in Polish: *Słowosieć*) version 3.2 (Maziarz et al. 2016) – the Polish wordnet; its senses with their lexico-semantic structure as well as their glosses (short definition-like descriptions of senses) and usage examples,

— raw unstructured textual corpora with no sense annotations (e.g. for sense induction methods or semi-supervised learning).

The second type of competition called *Open Competition* was focused on designing the best possible solution for WSD using all available knowledge and data sources. In this variant, we also encouraged the participants to use plWordNet, but the main focus was put on available sense annotated corpora including the corpora prepared for other languages as well as any linking of plWordNet with Linked Open Data. We suggested the following development resources:

— sense annotated corpora, mainly *Składnica* (Hajnicz 2014) and *plWordNet* glosses and usage examples, but also other sense annotated corpora prepared for any language,

— *plWordNet* mapping to Linked Open Data, mainly those originating from Wikipedia[1] and ontologies like SUMO ontology (Niles and Pease 2003) or YAGO ontology (Suchanek et al. 2008),

— existing thesauri, valency dictionaries and their mapping to wordnet prepared for any language e.g. Polish *Walenty* (Przepiórkowski et al. 2014), English *FrameNet* (Baker et al. 1998) or *VerbNet* (Schuler 2005).

The initial development data in a simplified format was published for the participants at `https://gitlab.clarin-pl.eu/ajanz/poleval20-wsd`.

[1] `https://pl.wikipedia.org/`

4. Datasets

In this section we precisely describe the current state of the Polish language resources related to the WSD task with a simple statistical view on their properties.

4.1. Development data

The previous sense annotated corpora e.g. *Składnica*, or *Polish Corpus of Wrocław University of Technology* (*KPWr*; Broda et al. 2012) prepared for WSD task were annotated with plWordNet 2.1 (Maziarz et al. 2014) sense inventory which makes them quite outdated. For the purpose of this competition, we decided to upgrade sense inventory to plWordNet 3.2 and suggest it[2] as a dedicated development dataset and annotation resource. This decision was made due to significant differences between 2.1 and 3.2 versions. Table 1 presents a statistical analysis of the differences between old and new sense inventories.

Table 1: Statistical analysis of Polish wordnet-based sense inventories

Feature	**plWordNet 2.1**	**plWordNet 3.2**
number of distinct lexical units	206 567	286 804
number of distinct multi-word lexical units	53 752	70 019
number of distinct synsets	151 252	221 101
number of monosemous lemmas	113 129	141 343
number of polysemous lemmas	33 507	49 049
number of monosemous lemmas (multi-word only)	43 906	56 415
number of polysemous lemmas (multi-word only)	3 898	5 171
number of lexical units with definition or any usage example	37 207	145 901
number of lexical units without definition or any usage example	169 360	140 903
average length of utterance (definition or example)	12.56	11.54
average number of senses per lemma (polysemous only)	2.79	2.96

The other part of our development data was based on a sense annotated corpus called *Składnica* (Hajnicz 2014). This part of the development data was intended for the *Open Competition* variant only. The corpus was annotated with plWordNet 2.1 senses making it slightly incompatible with the current sense inventory and our evaluation data. Still, many of the plWordNet 2.1 senses occurring in *Składnica* should be compatible to some extent with the plWordNet 3.2 senses. The existing compatibility issues will be solved in near future due to the work of CLARIN-PL[3] – see Section 7.

[2] We did not use the most contemporary version of that time, i.e. 4.1, due to existing mapping between plWordNet and Walenty which was not upgraded to the version 4.1.

[3] `http://clarin-pl.eu`

4.2. Evaluation data

We introduced two distinct evaluation datasets for the purpose of the competition:

— "The Adventure of the Speckled Band", henceforth *SPEC* corpus,[4] the eighth Conan Doyle's story about the adventures of the famous character named Sherlock Holmes – based on the modern translation of the original and expanded to an annotated language resource (Błaszczak et al. 2019),

— *KPWr–100* – a new fraction of 100 distinct documents existing in the Polish Corpus of Wrocław University of Science and Technology.

As it was mentioned earlier, the datasets were manually annotated using an updated sense inventory based on plWordNet 3.2. To annotate the data we trained three linguists as annotators. They were already familiar with the overall design and the structure of Polish wordnet because of their involvement in the process of plWordNet development from the very beginning. All the data was annotated independently and inner-annotator agreement was calculated. The Positive Specific Agreement (PSA) scores for SPEC and KPWr-100 corpora were 0.602 and 0.678, respectively.

Annotation guidelines

The guidelines presented here were designed in a way that takes into account the nature of Polish language resources and existing NLP tools. For instance, for the purpose of WSD task, we introduced several rules for handling multi-word expressions as they are an important contextual signal for disambiguation.

We assumed the following initial set of annotation rules:

— a gerund is always lemmatized to its initial bare infinitive form of its source verb and annotated by assigning a specific sense associated with this verb,

— participles are following the same rule as gerunds, i.e. annotated by assigning correct senses representing their source verbs,

— adjectives and adverbs in comparative and superlative forms are annotated by assigning the senses that represent their base positive forms e.g. *najpiękniejszy* 'most beautiful' → *piękny* 'beautiful' → *piękny* 1.adj.

In the case of multi-word expressions (MWE), e.g. *czerwona kartka* 'red card', each of its constituents should be annotated with a particular sense of recognized MWE ignoring the senses of these constituents. Only continuous MWE, i.e. such that all their component words occur in the text in one continuous sequence, were distinguished and annotated with their senses in the corpus. Specific verbs that are joined with agglutinate token *się* were also considered as multi-word expressions and annotated with correct senses.

To handle special expressions we decided to introduce a set of annotation tags to mark exceptional cases in a systematic and consistent way. The annotation tags were available

[4] `https://clarin-pl.eu/dspace/handle/11321/667`

only for the annotators and not included in the version available during the competition. The following set of tags was used:

— `OtherSense` – the tag should be used when a word or expression being disambiguated cannot be annotated due to the lack of its specific senses in our sense inventory; this tag is very useful since it signals the need for extension of the initial sense inventory,

— `TaggingError` – used to handle cases when the disambiguated word has been assigned with a wrong morpho-syntactic tag, e.g. a noun instead of a verb etc. during initial corpus preprocessing,

— `NamedEntity` – handles the cases in which an analysed word or expression represents a named entity, so it cannot be directly linked with some specific sense (plWordNet contains a very limited number of named entities by default), but it might be still useful for disambiguation,

— `IdiomaticPhrase` – this tag was introduced mainly to handle multi-word idiomatic expressions since they might have a strong impact on the disambiguation process,

— `ForeignWord` – represents a non-Polish word or expression found in the corpus,

— `TextError` – marks a word or expression which has a typographical error.

Tokens annotated manually by the annotators as `OtherSense`, `TaggingError`, `NamedEntity`, `ForeignWord`, or `TextError` were excluded from counting during the evaluation procedure (i.e. not taken into account for the measures) since they are incompatible with our sense inventory and the participants would not be able to correctly disambiguate them.

Properties of gold standard data

In this section we discuss the properties of our evaluation datasets – SPEC and KPWR-100. The overall distribution of all annotated tokens is presented in Figure 2. We also provide the statistics of annotated multi-word expressions with respect to their Part-of-Speech as shown in Table 3. The large part of annotated multi-word expressions consists of reflexive verbs with *się*. pronoun. However, we can also notice some noun expressions and adverbs in both corpora.

Table 2: The overall distribution of annotated tokens in our evaluation corpora with respect to their initial Part-of-Speech

Corpus	#Nouns	#Verbs	#Adject.	#Adverbs	#Total
KPWr-100	7 028	3 428	2 442	677	13 891
SPEC	1 617	1 182	487	219	3 689

Table 4 shows the number of annotated single words with respect to their Part-of-Speech excluding the ones that do not belong to any multi-word expression annotated in the corpora. We also analysed the distribution of senses and polysemy ratio in our evaluation datasets.

Table 3: The distribution of annotated multi-word expressions in our evaluation corpora with respect to their Part-of-Speech. The Part-of-Speech for particular MWE expressions might be different than the original Part-of-Speech of its constituents.

Corpus	#Nouns	#Verbs	#Adject.	#Adverbs	#Total
KPWr-100	205	282	0	28	515
SPEC	14	154	0	20	188

Table 4: The distribution of annotated words excluding annotated multi-word expressions

Corpus	#Nouns	#Verbs	#Adject.	#Adverbs	#Total
KPWr-100	6 769	3 140	2 271	673	13 891
SPEC	1 705	1 099	524	229	3 951

Figure 1 shows the distribution of senses for lemmas existing in the corpora. This distribution was determined by analysing our final manual annotations. The mean number of senses μ_{KPWr} and μ_{SPEC} was 1.24 and 1.11, respectively. This means, that for most of the lemmas their contexts were quite uniform in terms of their sense variation usually pointing to the same preferred sense. Figure 2 presents sense distribution of lemmas in our evaluation with respect to the underlying knowledge-base – plWordNet. This distribution shows that the algorithms had to choose between 3 to 4 meanings on average from our sense repository. These two observations, however, should not be treated as a clear indicator of task complexity.

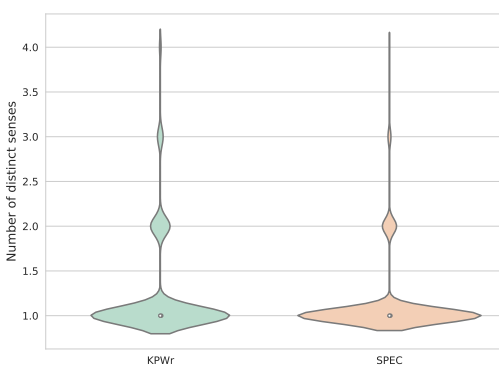


Figure 1: The distribution of the number of distinct senses computed for multi-sense lemmas. Descriptive statistics of sense distributions for KPWR corpus $\mu_{KPWr} = 1.24, \sigma_{KPWr} = 0.64$, and SPEC corpus $\mu_{SPEC} = 1.11$, $\sigma_{SPEC} = 0.39$.

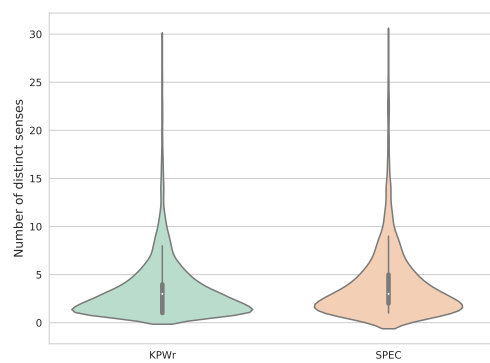


Figure 2: The distribution of the number of senses according to the underlying sense inventory. Descriptive statistics for KPWR corpus $\mu_{KPWr} = 3.45, \sigma_{KPWr} = 3.12$, and SPEC corpus $\mu_{SPEC} = 3.96, \sigma_{SPEC} = 3.68$.

5. Evaluation

The WSD task is usually seen as a classification problem. The standard classification metrics like precision and recall might be used to evaluate solutions. However, it is very difficult to compute a reliable recall score when we deal with highly imbalanced data and missing senses. The evaluation procedure proposed for this shared task was based on original metrics introduced by SENSEVAL and SemEval with precision and recall metrics adapted to the nature of WSD problem. This means that our precision and recall scores are defined as follows:

$$precision = \frac{|N_A^+|}{|N_A|} \tag{1}$$

$$recall = \frac{|N_A^+|}{|N_G|} \tag{2}$$

where the precision is defined here as a ratio of correct decisions of given disambiguation algorithm to all decisions that were made by this algorithm. N_A^+ is a set of words (tokens) with correctly predicted senses computed by algorithm A, and N_A represents a set of all words for which decisions were made by algorithm A. The recall score takes into account Gold Standard annotations, hence it is defined as a ratio of correct decisions of a given disambiguation algorithm to all Gold Standard decisions that were to be made by this algorithm N_G. If a WSD algorithm misses some Gold Standard tokens or disambiguate them incorrectly, then the recall decreases.

Submitted algorithms had to be capable of inferring if a given word is a part of multi-word expression or not. In the case of detecting a MWE, all its words (corresponding to the MWE components) had to be annotated with the same sense. Otherwise, a given algorithm should separately annotate the words with their respective senses. It should be mentioned that during the evaluation process assigning different senses to the tokens of a MWE occurrence is treated as wrong disambiguation.

6. Submitted solutions and results

The first edition of the proposed shared task unfortunately did not attract many participants. In this edition we received only 5 unique submissions from 2 distinct participants which made it the least popular task of PolEval 2020. However, we believe that all the effort on preparing a uniform basis for further experiments and evaluation in the area of WSD will be a motivating factor for the community for future work on developing more sophisticated and effective WSD algorithms. Table 5 presents the performance of all solutions submitted by the participants but limited only to their best-performing submissions evaluated on our test data. The solutions submitted by participants were prepared only for *Fixed Competition*. The model submitted by Dariusz Kłeczek (DK) outperformed the solution proposed by Arleta Juszczak (AJ). *DK-v3* was the best solution submitted by participants attending the contest. However, when we

compared the submitted solutions with selected baselines we found that there is still some place for improvement.

Table 5: The submitted solutions evaluated on KPWr-100 and SPEC data. As a baseline we proposed a heuristic picking always (F)irst (W)ord(N)et (S)ense.

Corpus	Submission	Precision	Recall	F_1
KPWr-100	*DK-V3*	0.599	0.589	0.594
	AJ-V2	0.318	0.231	0.268
	*FWNS**	0.563	0.556	0.559
	*WoSeDon**	**0.625**	**0.618**	**0.621**
SPEC	*DK-V3*	0.592	0.577	0.584
	AJ-V2	0.292	0.201	0.238
	*FWNS**	0.587	0.575	0.581
	*WoSeDon**	**0.607**	**0.594**	**0.600**

The usual baseline solution for WSD is based on the Most Frequent Sense heuristic (MFS) that disambiguates a given text token by taking the most frequent sense of the corresponding word (lemma) as it is observed in sense-annotated corpora. This kind of heuristic appeared to be a tough-to-beat baseline for WSD solutions, but it actually requires large sense annotated corpora to collect data. In the *Fixed Competition* we did not allow to use any of the available sense annotated corpora.[5] Thus, we decided to propose a heuristic that might be a good substitute for the MFS baseline. The *First Wordnet Sense* (FWNS) heuristic picks always a sense with the lowest sense *variant number* from all sense candidates of disambiguated word e.g. always prefers *piękny* 1.adj 'beautiful' than other senses of this word *piękny*, i.e. {*piękny* 2.adj, *piękny* 3.adj, ...} and so on. Senses with lower variant numbers were mostly introduced to plWordNet earlier than the variants with higher numbers.[6] As plWordNet was built by linguists, sense variants might be correlated with the human awareness of existing word senses and their natural ordering in our minds and be somehow positively correlated with sense frequencies.[7] Application of the FWNS heuristic as a correcting factor significantly improves a WSD algorithm for Polish (cf. Janz and Piasecki 2019). FWNS heuristic appeared to perform quite well also in our task (see Table 5).

The second baseline uses WoSeDon (Kędzia et al. 2015, Janz and Piasecki 2019) – a Polish word sense disambiguation tool based on well-known Personalized PageRank algorithm (Page et al. 1999). This method was initially introduced for disambiguating English texts and included later in a tool called UKB (Agirre and Soroa 2009). In this work, WoSeDon was configured to operate only on raw plWordNet 3.2 knowledge-base without any extensions. As all participants declared their solutions as intended for *Fixed Competition*, we also did not use

[5] That are also very small in the case of the Polish language and not representative at all.

[6] Some exception might happen due to sense reorganisations resulting from wordnet editing and correcting over the years.

[7] However, sense numbering is also influenced by the organisation of work on plWordNet construction over years. Thus the above outline hypothesis on possible positive correlation between the sense number and its salience for a given lemma is only an unverified hypothesis only, even if very likely one.

any additional heuristics (including FWNS) that could disqualify the model from the *Fixed Competition*.

The results clearly show that the proposed baselines are still tough-to-beat when the training data is quite limited. WoSeDon outperformed all submitted solutions including *DK-v3* based on BERT architecture.

7. Conclusions

This paper has briefly summarised the first edition of the WSD Shared Task at PolEval 2020. 5 submissions from 2 participants were registered for this task. A few promising results based on neural transformer architecture were presented. For comparison we also introduced two baseline solutions, including one state-of-the-art system for Polish, namely WoSeDon. Moreover, the performance of all solutions, submissions and baselines, was measured on a new test data just created for the task. New evaluation corpora and development data will be publicly available. We also plan to prepare an extended benchmark dataset for Polish WSD compatible with the dedicated sense inventory – plWordNet 3.2. We hope that the collected data and a uniform evaluation environment will facilitate further development of novel word sense disambiguation methods in the future.

Acknowledgements

This research was financed by the National Science Centre in Poland, grant number 2018/29/B /HS2/02919, and supported by the CLARIN-PL[8] research infrastructure.

References

Agirre E. and Soroa A. (2009). *Personalizing PageRank for Word Sense Disambiguation*. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pp. 33–41. Association for Computational Linguistics.

Agirre E., Lopez de Lacalle O., Magnini B., Otegi A., Rigau G. and Vossen P. (2008). *SemEval-2007 Task 01: Evaluating WSD on Cross-Language Information Retrieval*. In Peters C., Jijkoun V., Mandl T., Müller H., Oard D. W., Peñas A., Petras V. and Santos D. (eds.), *Advances in Multilingual and Multimodal Information Retrieval*, pp. 908–917. Springer.

Agirre E., López de Lacalle O. and Soroa A. (2014). *Random Walks for Knowledge-Based Word Sense Disambiguation*. Computational Linguistics 40(1), pp. 57–84.

Baker C. F., Fillmore C. J. and Lowe J. B. (1998). *The Berkeley FrameNet project*. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, pp. 86–90. Association for Computational Linguistics.

[8] `http://clarin-pl.eu`

Banerjee S. and Pedersen T. (2002). *An Adapted Lesk Algorithm for Word Sense Disambiguation Using WordNet*. In Gelbukh A. (ed.), *Proceedings of the International Conference on Intelligent Text Processing and Computational Linguistics: Computational Linguistics and Intelligent Text Processing (CICLing 2002)*, vol. 2276 of *Lecture Notes in Computer Science*, pp. 136–145. Springer.

Baś D., Broda B. and Piasecki M. (2008). *Towards Word Sense Disambiguation of Polish*. In *Proceedings of the International Multiconference on Computer Science and Information Technology: 3rd International Symposium Advances in Artificial Intelligence and Applications (AAIA'08)*, pp. 65–71.

Błaszczak M., Paszke K., Rudnicka E., Oleksy M., Wieczorek J., Kobylińska W., Fikus D. and Kałkus D. (2019). *The Adventure of the Speckled Band 1.0 (manually tagged)*. CLARIN-PL digital repository. `http://hdl.handle.net/11321/667`.

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. European Language Resources Association.

Brown P. F., Della Pietra S. A., Della Pietra V. J. and Mercer R. L. (1991). *Word-Sense Disambiguation Using Statistical Methods*. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics*, pp. 264–270. Association for Computational Linguistics.

Edmonds P. and Cotton S. (2001). *SENSEVAL-2: Overview*. In *Proceedings of the 2nd International Workshop on Evaluating Word Sense Disambiguation Systems (SENSEVAL-2)*, pp. 1–5. Association for Computational Linguistics.

Gale W. A., Church K. W. and Yarowsky D. (1992). *A Method for Disambiguating Word Senses in a Large Corpus*. Computers and the Humanities 26(5–6), pp. 415–439.

Hajnicz E. (2014). *Lexico-Semantic Annotation of Składnica Treebank by means of PLWN Lexical Units*. In Orav H., Fellbaum C. and Vossen P. (eds.), *Proceedings of the 7th Global Wordnet Conference (GWC 2014)*, pp. 23–31. University of Tartu Press.

Huang L., Sun C., Qiu X. and Huang X. (2019). *GlossBERT: BERT for Word Sense Disambiguation with Gloss Knowledge*. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3509–3514. Association for Computational Linguistics.

Janz A. and Piasecki M. (2019). *A Weakly Supervised Word Sense Disambiguation for Polish Using Rich Lexical Resources*. Poznan Studies in Contemporary Linguistics 55(2), pp. 339–365.

Kågebäck M. and Salomonsson H. (2016). *Word Sense Disambiguation using a Bidirectional LSTM*. In *Proceedings of the 5th Workshop on Cognitive Aspects of the Lexicon (CogALex-V)*, pp. 51–56.

Kędzia P., Piasecki M. and Orlińska M. (2015). *Word Sense Disambiguation Based on Large Scale Polish CLARIN Heterogeneous Lexical Resources*. Cognitive Studies 15, pp. 269–292.

Kumar S., Jat S., Saxena K. and Talukdar P. (2019). *Zero-shot Word Sense Disambiguation using Sense Definition Embeddings*. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 5670–5681. Association for Computational Linguistics.

Landes S., Leacock C. and Tengi R. I. (1998). *Building Semantic Concordances*. In *WordNet: an Electronic Lexical Database*, pp. 199–216. MIT Press.

Lee Y. K. and Ng H. T. (2002). *An Empirical Evaluation of Knowledge Sources and Learning Algorithms for Word Sense Disambiguation*. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pp. 41–48. Association for Computational Linguistics.

Lesk M. (1986). *Automatic Sense Disambiguation Using Machine Readable Dictionaries: How to Tell a Pine Cone from an Ice Cream Cone*. In *Proceedings of the 5th Annual International Conference on Systems Documentation*, pp. 24–26. Association for Computing Machinery.

Luo F., Liu T., Xia Q., Chang B. and Sui Z. (2018). *Incorporating Glosses into Neural Word Sense Disambiguation*.

Maziarz M., Piasecki M., Rudnicka E. and Szpakowicz S. (2014). *plWordNet as the Cornerstone of a Toolkit of Lexico-semantic Resources*. In *Proceedings of the 7th Global Wordnet Conference*, pp. 304–312. University of Tartu Press.

Maziarz M., Piasecki M., Rudnicka E., Szpakowicz S. and Kędzia P. (2016). *plWordNet 3.0 – a Comprehensive Lexical-Semantic Resource*. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pp. 2259–2268. The COLING 2016 Organizing Committee.

Moro A., Raganato A. and Navigli R. (2014). *Entity Linking meets Word Sense Disambiguation: a Unified Approach*. Transactions of the Association for Computational Linguistics 2, pp. 231–244.

Ng H. T. and Lee H. B. (1996). *Integrating Multiple Knowledge Sources to Disambiguate Word Sense: An Exemplar-Based Approach*. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pp. 40–47. Association for Computational Linguistics.

Niles I. and Pease A. (2003). *Linking Lexicons and Ontologies: Mapping WordNet to the Suggested Upper Merged Ontology*. In Arabnia H. R. (ed.), *Proceedings of the International Conference on Information and Knowledge Engineering (IKE'03), Volume 2*, pp. 412–416. CSREA Press.

Page L., Brin S., Motwani R. and Winograd T. (1999). *The PageRank Citation Ranking: Bringing Order to the Web*. Technical Report 1999-66, Stanford InfoLab.

Przepiórkowski A., Hajnicz E., Patejuk A., Woliński M., Skwarski F. and Świdziński M. (2014). *Walenty: Towards a Comprehensive Valence Dictionary of Polish*. In Calzolari N., Choukri K., Declerck T., Loftsson H., Maegaard B., Mariani J., Moreno A., Odijk J. and Piperidis S. (eds.), *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC 2014)*, pp. 2785–2792. European Language Resources Association.

Pustejovsky J. and Boguraev B. (1996). *Lexical Semantics: The Problem of Polysemy*. Clarendon Press.

Raganato A., Delli Bovi C. and Navigli R. (2017). *Neural Sequence Learning Models for Word Sense Disambiguation*. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 1156–1167. Association for Computational Linguistics.

Schuler K. K. (2005). *VerbNet: A Broad-Coverage, Comprehensive Verb Lexicon*. Ph.D. thesis, University of Pennsylvania.

Suchanek F. M., Kasneci G. and Weikum G. (2008). *YAGO: A Large Ontology from Wikipedia and WordNet*. Journal of Web Semantics 6(3), pp. 203–217.

Yarowsky D. (1994). *Decision Lists for Lexical Ambiguity Resolution: Application to Accent Restoration in Spanish and French*. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pp. 88–95. Association for Computational Linguistics.

Polbert: Attacking Polish NLP Tasks with Transformers

Dariusz Kłeczek (skok.ai)

Abstract

Different NLP tasks have required different solution architectures. This is especially true for Polish language, given its complexity. The recent trend of pretraining large, transformer-based language models, such as BERT, on large corpora, gives us an opportunity to consider a unifying architecture capable of solving these different tasks. Polbert is a Polish version of BERT language model, pretrained on a large Polish corpus. In this paper, I present Polbert, and discuss how it can be applied to solve a range of tasks from PolEval 2020 challenge, including the winning submission to PolEval 2020 Task 3: Word Sense Disambiguation.[1]

Keywords

BERT, Transformer, Polish, Natural Language Processing

1. Introduction

In the last 2 years, pretrained language models have conquered the NLP benchmarks (Howard and Ruder 2018, Radford et al. 2019). Especially the transformer (Vaswani et al. 2017) and BERT (Devlin et al. 2019) architecture proved very effective in solving various NLP tasks. Pretrained models were being made in monolingual and multilingual variants. When I trained and released Polbert in March 2020, it was the first Polish BERT model broadly available. In this article, I will describe my approach to training Polbert, and how I used it to solve a range of NLP tasks from the PolEval 2020 competition.

[1] `http://2020.poleval.pl/tasks/task3/`

2. Polbert

2.1. Models

Polbert model follows `bert-base-uncased` model architecture (Devlin et al. 2019), with 12 layers, 768 hidden units embeddings, 12 attention heads, and 110M parameters in total. It is available in two variants: cased and uncased, and can be downloaded via HuggingFace library[2] (Wolf et al. 2019).

2.2. Corpus

Polbert is trained on a large Polish-language corpus described in Table 1 (Tiedemann 2012, Ogrodniczuk 2018). The training corpora were divided into sentences with srxsegmenter,[3] concatenated and tokenized with HuggingFace BERT Tokenizer. I initially trained the uncased model, and after working with it for a while I noticed issues that have been corrected in the cased model. First, some Polish characters and accents are not tokenized correctly through the BERT tokenizer when applying lowercase. This doesn't impact sequence classification much, but may influence token classification tasks significantly. Second, I noticed a lot of duplicates in the Open Subtitles dataset, which dominates the training corpus. These duplicates were removed before training the cased model, which resulted in a smaller, but more balanced training corpus.

Table 1: Polbert training corpus

Dataset	Uncased	Cased	Lines	Words
Polish subset of Open Subtitles	x		236 635 408	1 431 199 601
Polish subset of Open Subtitles (deduplicated)		x	41 998 942	213 590 656
Polish subset of ParaCrawl	x	x	8 470 950	176 670 885
Polish Parliamentary Corpus	x	x	9 799 859	121 154 785
Polish Wikipedia (Feb 2020)	x	x	8 014 206	132 067 986
Total uncased	x		262 920 423	1 861 093 257
Total cased		x	68 283 960	646 479 197

2.3. Training

Polbert was trained with code provided in Google BERT's GitHub repository.[4] The training setup is described in Table 2. While training the cased variant, I apply Whole Word Masking. Both models were trained on a single Google Cloud TPU v3-8.

[2] `https://huggingface.co/dkleczek`

[3] SRX rules file for sentence splitting in Polish, written by Marcin Miłkowski: `https://raw.githubusercontent.com/languagetool-org/languagetool/master/languagetool-core/src/main/resources/org/languagetool/resource/segment.srx`

[4] `https://github.com/google-research/bert`

Table 2: Polbert training approach

Model	Training setup			
	Steps	Sequence length	Batch size	Learning rate
Uncased	100 000	128	512	1e-4 (10 000 steps warmup)
	800 000	128	512	5e-5
	100 000	512	256	2e-5
Cased	100 000	128	2048	1e-4 (10 000 steps warmup)
	100 000	128	2048	5e-5
	100 000	512	256	2e-5

2.4. Evaluation

KLEJ benchmark (Rybak et al. 2020) is a set of nine evaluation tasks for Polish language understanding. The results captured in Table 3 are achieved by running a standard set of evaluation scripts, utilizing both cased and uncased variants of Polbert. Can you see how the uncased model performs better than cased on some tasks? I hypothesize this is because of the oversampling of Open Subtitles dataset and its similarity to data in some of these tasks.

Table 3: Polbert results on KLEJ benchmark

Result	Polbert cased	Polbert uncased
NKJP-NER	93.6	90.1
CDSC-E	93.4	93.9
CDSC-R	93.8	93.5
CBD	52.7	55.0
PolEmo2.0-IN	87.4	88.1
PolEmo2.0-OUT	71.1	68.8
DYK	59.1	59.4
PSC	98.6	98.8
AR	85.2	85.4
Average	81.7	81.4

3. Applications

3.1. Post-editing and rescoring of automatic speech recognition results

The goal of this task was to convert a sequence of words from an automatic speech recognition (ASR) system into another sequence of words that reflects the actual spoken utterance. The data provided by organizers consisted of the outputs from an ASR system: 1-best output (each utterance containing a single best transcript), n-best output (each utterance containing up

to 100 best alternative hypotheses of the ASR output), and lattice output (each utterance containing a list of arcs forming a lattice of the ASR output).

How do we frame this as a problem that can be solved by transformer-based language model? It would be interesting to attempt solving this as a sequence-to-sequence problem, similar to Machine Translation. Unfortunately, Polbert consists only of the encoder part, and hence cannot be used in this way.

Salazar et al. (2020) describe a method for using pretrained masked language models to score sentences. We mask tokens in a sentence one by one, then sum the log probability for each missing token to get the pseudo-log-likelihood score of that sentence. This score can be used to select alternative hypotheses among the ASR output (n-best or lattice) that are more likely to represent the actual utterance. As indicated by the authors, this method is computationally very expensive and exceeded my GPU budget to attempt it on the PolEval 2020 ASR task. Salazar et al. (2020) also propose a method to reduce the computational budget by training a network to match BERT's scores without masked tokens, by finetuning the network via regression over the [CLS] token. That assumes, however, that we have already calculated a number of sentence scores which can be used as training examples.

In my solution to the PolEval 2020 ASR task, I developed a simple classification model, using Polbert as the encoder, with a single linear classification layer on top of the [CLS] token hidden state. The model is trained on Polish Parliamentary Corpus, taking all utterance transcripts as examples with *True* label, and the corresponding ASR 1-best output as examples with *False* label. I use this model to classify 100-best hypotheses for each utterance, and select the one with the highest logits. This solution improves the word-error rate on the test dataset from 27.6% to 26.9%.

3.2. Word Sense Disambiguation

The goal of WSD task was to identify the correct sense for each ambiguous word appearing in a text. Given the limited availability of training data, the organizers suggested focusing on knowledge-based approaches to the problem, by using available language resources such as WordNet, and weakly supervised approaches making use of small annotated data and the knowledge extracted from large unstructured textual corpora.

My solutions follow the rules of fixed competition, i.e. I didn't use any data other than the provided Polish WordNet version 3.2 (Maziarz et al. 2016) data to train my model. The solution code is released on Github.[5]

Previous research

My solution is primarily inspired by GlossBERT by Huang et al. (2020) who proposed to construct context-gloss pairs from all possible senses of the target word in WordNet, and then convert WSD to a sentence-pair classification task. Context is defined here as the sentence containing the target word to be disambiguated. Gloss is a brief definition of a sense associated

[5] `https://github.com/kldarek/poleval2020`

with the target word. A binary classification layer indicates whether the context and gloss sentences in a pair correspond to the same word sense.

Huang et al. experiment with three approaches. In *Token-CLS*, the final hidden state of the token corresponding to the target word is the input to the classification layer. In *Sent-CLS*, the final hidden state corresponding to the first token [CLS] goes into the classification layer. *Sent-CLS-WS* also uses the [CLS] token, but additional weak supervision is added to the input: target word is surrounded by quotation marks in the context sentence, and it is also added at the beginning of the gloss sentence.

By comparing the context sentence, containing our target word, with the gloss sentence for each of the WordNet senses, we convert the original N-class classification task into N binary classification tasks, where N equals to the number of senses corresponding to the target word. See examples in Table 4.

Table 4: Construction of sentence pair examples: Glossbert (top), Polbert (bottom)

Label: True	Context:	Wszystkie trzy "pokoje" mają okna od strony parku
	Pair:	pokój: pomieszczenie mieszkalne, w którym się przebywa, także w hotelu
Label: False	Context:	Wszystkie trzy "pokoje" mają okna od strony parku
	Pair:	pokój: stan, gdy nie ma wojny
Label: True	Context:	pokój: Wszystkie trzy **pokoje** mają okna od strony parku
	Pair:	pokój: Dwa **pokoje** z kuchnią w zupełności mi wystarczą...
Label: False	Context:	pokój: Wszystkie trzy **pokoje** mają okna od strony parku
	Pair:	pokój: Za szczególny wkład w promowanie **pokoju** i praw człowieka...

Solution overview

I made two modifications to the *GlossBERT Sent-CLS-WS* design. First, I use both gloss and examples sentences from WordNet to construct my sentence pairs. Second, I modify both sentences in a pair to have the same format (*target word lemma: sentence with emphasis around **target word***). I didn't have time for experiments or ablation study, but I believe that structuring the paired sentences in a similar way, and using all examples available in WordNet, both help the model with the WSD task.

To create my training examples, I iterate over the list of WordNet lemmas with associated synsets and examples. If there are more examples associated with the same lemma and synset, I pick one at random and create a sentence pair with *True* label. If there are examples associated with the same lemma and other synsets, I pick one at random and create a sentence pair with *False* label. In this way, I create 443 316 sentence pairs, using the last 20 000 for validation and the remainder as my training dataset.

I use BERT as an encoder for the sentence-pair example, followed by dropout *(p=0.2)*, and a linear binary classification layer on top of the pooled output, corresponding to the [CLS] token. I use HuggingFace library (Wolf et al. 2019), load the pretrained Polbert uncased model weights, and fine-tune it for 5 epochs with learning rate 2e-5, early stopping, max length of

64, batch size 32, AdamW optimizer and linear schedule with warmup. The accuracy for the best epoch (4) is 85.7% on my validation dataset.

Disambiguation: the baseline

In order to disambiguate words in a target text, I start by splitting the text into sentences and identifying the lemma associated with each word in a sentence (both of these are provided in the PolEval 2020 WSD Task). The lemma with the sentence containing it becomes the first element of the sentence pair provided to Polbert model. I then list all synsets associated with the lemma. For each synset, I list all examples associated with it and construct sentence pairs with those examples. I run the model on each of these sentence pairs and average the model outputs (scores) across all examples associated with a single synset. The synset with the highest average score is selected as the disambiguated sense for the target word.

Disambiguation: Multi-Word Expressions

One problem with the baseline solution is that it doesn't account for multi-word expressions. To make up for this, I pre-process the sentences and check if there is a WordNet lemma associated with any bigram or trigram from that sentence. If I find a match, then I use the lemmas associated with the n-grams for disambiguation, instead of the original lemmas associated with the single words. For example, the trigram *w dużej mierze* has its own lemma in WordNet, and I use that lemma for disambiguation, instead of lemmas associated with the individual words (*w, duży, miara*).

Disambiguation: missing sense examples

Another problem with the baseline solution was that it couldn't disambiguate senses if the corresponding synset lacked examples. For those synsets, I look up synsets connected via both hypernymy and hyponymy relationships and associated examples. I use these examples to score the synset that was missing examples.

To illustrate this, there isn't a definition or example associated with the lemma *szczędzenie*. To disambiguate this word, I would use examples associated with its hipernyms and hyponyms (*oszczędzanie*).

Results

I prepared two solutions to the PolEval 2020 WSD Task. Both are based on the same model described above. The first solution *Baseline* refers to the disambiguation baseline. The second solution *Baseline+MWE+Rels* includes the modifications to address multi-word expressions and missing sense examples. Table 5 presents the results of both solutions on the competition dataset.

Further work

I believe the results on WSD task can be further improved. One way is to improve the sentence-pair classification model by adjusting the finetuning protocol and dataset. Another

Table 5: PolEval 2020 WSD Task Results

Submission	KPWr		Sherlock	
	Precision	Recall	Precision	Recall
Baseline+MWE+Rels	**0.599296**	**0.588727**	**0.592263**	**0.576850**
Baseline	0.564432	0.550860	0.564384	0.542966

way is to improve the disambiguation algorithm, by considering how the related synsets are identified, scored and compared. It would be also interesting to compare Polbert with other transformer-based language models. Finally, this approach didn't use any annotated sense data, which could give a further boost to the model's performance.

3.3. Information extraction and entity typing from long documents with complex layouts

This challenge was about collecting information from real, long documents with complex page layouts by integrating entities found across multiple pages and text sections, tables, plots, forms, etc. This included recognising units with a standard name (NER) (e.g. person, location or organisation), but also the roles of units in whole documents (e.g. chairman of the board, date of signature).

For each document in this task, we get the raw pdf file and two files produced with an OCR tool: raw text and text with positional info. While Garncarek et al. (2020) show how the positional info may be useful for information extraction, I decided to use the raw text only. In my solution, I follow a procedure similar to the one described by Graliński et al. (2020).

My final solution was based on multilingual BERT encoder, NER architecture, and Random Forest postprocessing. It resulted in 0.44 F_1 score on the test set. After the competition finished, I replicated the solution only changing the encoder, from multilingual BERT to Polbert. That resulted in 0.446 F_1 score.

Preprocessing

One difficulty associated with this challenge is the length of documents. In order to deal with it, I start by tokenizing each document and then divide it into chunks. I experimented with chunks of length 128 and 512, and the latter led to better scores and is used in the final submission.

Autotagging

The training and validation datasets provided in this task consist of the documents and the final information extracted from those documents (entities and relationships). We are not given information about the location of these entities in the documents. I decided to use auto-tagging to create a training dataset for the model. For each target entity, I check if it's contained in any of the document chunks, and tag the related tokens in case I find a match.

For some entities (such as period or date), I create several variants of the targets to reflect different ways these entities may be represented in the text, for example: *1 maja 2020, 1.5.2020, 1/5/2020* etc. I believe that this step is crucial to the quality of model outputs, and my final result could be improved by spending more time on tagging the training data.

A special case in this challenge is entity linking - people signing documents need to be linked with their role and date of the signature. I found that in most of the training and validation documents, the date of signature corresponds to the date of the document, and I apply this as a rule after identifying the document date. For linking people and roles, I experimented with two approaches. The first approach consists of identifying people and roles as separate entity types and then connecting them in the same order that they appear in a document chunk. This leads to some problems, for example when a model recognizes a different number of people and role entities in a single chunk. The second approach is based on the observation that majority of the role entities in the training and validation documents can be represented as three classes (*Prezes Zarządu, Wiceprezes Zarządu, Członek Zarządu*). I treat these classes as different entity types and tag the names of the people via their role, rather than a generic *PERSON* entity. Both approaches performed similarly in my experiments, and I used the first approach in my final submission.

Model

In this challenge, I experimented with different choices for the encoder and the head of the information extraction model. I used Polbert, multilingual BERT, and Polish Roberta (Dadas et al. 2020) as encoders. I experimented with two head architectures. In the NER architecture, I apply dropout to the sequence output from BERT encoder, and then a linear classification layer with 9 classes (entity types). In the QA architecture, I use the same sequence outputs followed by dropout, and then apply a linear classification layer to identify the span (start and end tokens) corresponding to each entity type. In my experiments, NER architecture performed better than QA, and I used it in my final submissions. Due to the time and resource constraints, I didn't run a direct comparison of performance across various encoders.

Postprocessing

The first step of postprocessing is to standardize the entities extracted from the documents into the target format, which is essentially a reverse operation to the transform variants performed during auto-tagging. The next step is to aggregate the information extracted across multiple document chunks. Given that the chunks overlap, I assume that only one chunk per document contains the needed information, so for each entity type, I need to select the right chunk. I started by selecting the chunk with the maximum logits for each entity (averaged across the tokens corresponding to a given entity). Then I created a random forest for each entity type, trained on the validation chunks, taking into account the entity logits and other features, such as the logits of other entities identified in the same chunk.

4. Future work

There are many types of NLP problems and many ways to further use transformer-based language models such as Polbert. Beyond pursuing these applications, I believe two important

directions need further work, especially for the Polish language. First, we need smaller variants of the pretrained models, which would make it easier to broadly apply them in production. Second, we need to develop Polish variants of sequence-to-sequence models that can be used for example in translation or summarization.

5. Conclusions

Polbert is a transformer-based language model pretrained on a large Polish language corpus. It offers a universal sentence and token representation, which can be used, after finetuning, on a broad range of NLP tasks.

6. Acknowledgements

I'd like to thank the organizers of PolEval competition (Łukasz Kobyliński, Maciej Ogrodniczuk), ASR Task (Danijel Koržinek), Information Extraction Task (Anna Wróblewska, Filip Graliński, Dawid Lipiński), and specifically the WSD Task (Arkadiusz Janz, Maciej Piasecki) for the opportunity to participate and the guidance.

References

Dadas S., Perełkiewicz M. and Poświata R. (2020). *Pre-training Polish Transformer-based Language Models at Scale*. arXiv:2006.04229.

Devlin J., Chang M.-W., Lee K. and Toutanova K. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. arXiv:1810.04805.

Garncarek Ł., Powalski R., Stanisławek T., Topolski B., Halama P. and Graliński F. (2020). *LAMBERT: Layout-Aware (Language) Modeling using BERT for Information Extraction*. arXiv:2002.08087.

Graliński F., Stanisławek T., Wróblewska A., Lipiński D., Kaliska A., Rosalska P., Topolski B. and Biecek P. (2020). *Kleister: A Novel Task for Information Extraction involving Long Documents with Complex Layout*. arXiv:2003.02356.

Howard J. and Ruder S. (2018). *Universal Language Model Fine-tuning for Text Classification*. arXiv:1801.06146.

Huang L., Sun C., Qiu X. and Huang X. (2020). *GlossBERT: BERT for Word Sense Disambiguation with Gloss Knowledge*. arXiv:1908.07245.

Maziarz M., Piasecki M., Rudnicka E., Szpakowicz S. and Kędzia P. (2016). *plWordNet 3.0 – a Comprehensive Lexical-Semantic Resource*.

Ogrodniczuk M. (2018). *Polish Parliamentary Corpus*. In Fišer D., Eskevich M. and de Jong F. (eds.), *Proceedings of the LREC 2018 Workshop* ParlaCLARIN: Creating and Using Parliamentary Corpora, pp. 15–19. European Language Resources Association.

Radford A., Wu J., Child R., Luan D., Amodei D. and Sutskever I. (2019). *Language Models are Unsupervised Multitask Learners*. `https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf`.

Rybak P., Mroczkowski R., Tracz J. and Gawlik I. (2020). *KLEJ: Comprehensive Benchmark for Polish Language Understanding*. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 1191–1201. Association for Computational Linguistics.

Salazar J., Liang D., Nguyen T. Q. and Kirchhoff K. (2020). *Masked Language Model Scoring*. arXiv:1910.14659.

Tiedemann J. (2012). *Parallel Data, Tools and Interfaces in OPUS*. In Chair) N. C. C., Choukri K., Declerck T., Dogan M. U., Maegaard B., Mariani J., Odijk J. and Piperidis S. (eds.), *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC 2012)*, pp. 2214–2218. European Language Resources Association.

Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser L. and Polosukhin I. (2017). *Attention Is All You Need*. arXiv:1706.03762.

Wolf T., Debut L., Sanh V., Chaumond J., Delangue C., Moi A., Cistac P., Rault T., Louf R., Funtowicz M., Davison J., Shleifer S., von Platen P., Ma C., Jernite Y., Plu J., Xu C., Scao T. L., Gugger S., Drame M., Lhoest Q. and Rush A. M. (2019). *HuggingFace's Transformers: State-of-the-art Natural Language Processing*. arXiv:1910.03771.

Results of the PolEval 2020 Shared Task 4: Information Extraction from Long Documents with Complex Layouts

Filip Graliński (Adam Mickiewicz University,
Faculty of Mathematics and Computer Science)

Anna Wróblewska (Warsaw University of Technology,
Faculty of Mathematics and Information Science)

Abstract

In this paper, we present PolEval Task 4, the challenge for information extraction and entity typing from long documents with complex layouts. We give the rationale for the data set and compare the results obtained during the evaluation campaign.

Keywords

named entity recognition, document layout analysis

1. Introduction

Named Entity Recognition (NER) is a well-known type task in the Natural Language Recognition (NLP) community, as exemplified by, for instance, the CoNLL 2013 challenge (Tjong Kim Sang and De Meulder 2003) or, for Polish, the NKJP NER data set (Przepiórkowski et al. 2012) along with the PolEval 2018 Task 2. NER data sets usually contain entities labeled for general types: persons, organizations, locations, dates, amounts of money. The problem is that it does not reflect the real-world needs, when considering domain-specific legal and business-related documents.

The PolEval Task 4 was built on a novel Polish dataset for Information Extraction and NLP reasoning challenge. The dataset comprises long documents with complicated layouts. Our objective was to prepare a new data set for Polish, resembling business machine learning applications. The task was aimed to represent various challenges faced in business contexts, e.g. non-trivial 2D layouts, business logic, extracting information from long documents,

potentially noisy dataset. We hope that the data set will form a good benchmark for complex Information Extraction systems based on NLP.

2. Data set description

2.1. General description

Information gathering from real-life, long documents must deal with the complex layout of pages by integrating found entities along with multiple pages and text sections, tables, plots, forms etc. To encourage progress on more in-depth and more complex information extraction algorithms, we presented a dataset in which systems must extract the most important information about various types of entities from formal documents. These entities are not only classes from standard named entity recognition (NER) systems (e.g. person, localisation, or organisation), but also the roles of the entities in the whole documents (e.g. CEO, issue date).

The data set is based on reports in the PDF format submitted by companies to the ESPI (*Elektroniczny System Przekazywania Informacji*) system.[1]

2.2. Basic statistics

The dataset consists of two sets (train and validate) and one test set. Each of these folders contains a *.csv* file with ground truth values for each report. For each report there are *.pdf* (raw input), *.txt* (text input) and *.hocr* (text input with positional info) files placed in the *reports/report_id* folder. The dataset (train and validate) contains 2216 unique records. The test set is composed of 555 documents. More data is given in Table 1.

Table 1: The data set in numbers

	Train	**Val**	**Test**
Documents	1628	548	555
Size in chars	255.1M	84.2M	90.9M
Data points	36519	12314	12700

The ground truth contains the following columns (see also Table 2):

— `id` – unique identifier of a specific financial report,

— `company` – name of the company,

— `drawing_date` – date which specifies when the financial report was submitted,

— `period_from` – start of the obligation period,

— `period_to` – end of the obligation period,

[1] `https://www.knf.gov.pl/dla_rynku/espi`

— `postal_code` – postal code of the company,

— `city` – the city where the company is registered,

— `street` – the name of the street where the company is registered,

— `street_no` – the number of the street at which the company is registered,

— `people` – members/chairpersons of the company management; a cell contains a list of tuples, where each tuple has the following form: (<date of signature>, <name and surname>, <position>) e.g. ('2019-12-16', 'Jan Kowalski', 'Prezes Zarządu').

Table 2: Sample ground truth of documents in the dataset

id; company; drawing_date; period_from; period_to; postal_code; city; street; street_no; people
208910;ZAKŁADY MAGNEZYTOWE ROPCZYCE SA;2012-08-30;2012-01-01;2012-06-30;39-100;Ropczyce;ul. Przemysłowa;1;[('2012-08-30', 'Józef Siwiec', 'Prezes Zarządu'), ('2012-08-30', 'Marian Darłak', 'Wiceprezes Zarządu'), ('2012-08-30', 'Robert Duszkiewicz', 'Wiceprezes Zarządu')]
118734;PC GUARD S.A.;2009-08-31;2009-01-01;2009-06-30;60-467;Poznań;Jasielska 16;16;[('2009-08-31', 'Dariusz Grześkowiak', 'Prezes Zarządu'), ('2009-08-31', 'Mariusz Bławat', 'Członek Zarządu')]

For each column, no more than 15% of documents can have an incorrect ground-truth value (see Table 3). The table also presents the results of our baselines comprising regular expressions crafted to each entity individually.[2]

Table 3: Coverage of entities in the whole dataset

Entity	**% of the documents in the dataset**	**Regex benchmark (accuracy on test set)**
company	88.16	57
street	88.99	24
drawing_date	93.40	32
postal_code	94.70	39
city	98.85	34
street_no	98.92	38
period_from	99.57	64
period_to	99.75	66
people	100.00	–

[2] `https://github.com/wojciech1871/hocr_parser`

3. Evaluation

F_1 was used as the evaluation metric, as implemented (as MultiLabel-F1) in the GEval evaluation tool (Graliński et al. 2019). This evaluation scheme was natural for 'atomic' data points, but the `people` values expressed as tuples posed a special difficulty. It was decided to use the following scheme for them:

1. The 3 elements of tuples (<date of signature>, <name and surname>, <position>) were added as 'atomic' values.
2. In order to check the actual relations, two data-point types were introduced: one for the person-date-of-signature relation, one for the person-position relation.
3. All these 5 types were treated within the F_1 just as the remaining types.

Also, some simple pre-processing was done as part of the evaluation procedure:

— all values were case-folded (hence, the metric was case-insensitive),

— the abbreviation *ul.* (*street*) was stripped (but not similar abbreviations), as it was used inconsistently (in gold standard vs actual documents).

The results are reported with bootstrap sampling with 200 samples (the `-B 200` option in GEval) to calculate the 95% confidence intervals for F_1 (Koehn 2004).

For better presentation, the results are given as percentages (multiplied by 100).

4. Result summary

There were a total of 8 submissions by 3 unique submitters. The final overall results (on the test set) are presented in Table 4. As can be seen, the differences between the best solutions submitted by distinct submitters are substantial, values for precision and recall are quite similar, not much precision-recall trade-off was done.

Table 4: The overall results

Submission	**F_1**	**P**	**R**
double_big	60.6 ± 1.7	60.8 ± 1.7	60.4 ± 1.8
double_small	58.8 ± 1.8	60.0 ± 1.8	57.7 ± 1.9
middle_big	58.5 ± 1.6	59.9 ± 1.7	57.1 ± 1.7
MBART+RF	44.0 ± 1.4	49.1 ± 1.4	39.8 ± 1.7
CLEX	**65.1 ± 1.9**	**63.8 ± 2.5**	**66.6 ± 1.7**
100_RF	58.4 ± 1.6	56.8 ± 1.6	60.0 ± 1.9
300_xgb	59.2 ± 1.5	57.0 ± 1.6	61.4 ± 1.9
300_RF	58.7 ± 1.5	56.9 ± 1.7	60.5 ± 1.8

The results for specific types of data points (limited to 'pure' extractions, i.e. extraction of atomic data points) are given in Table 5. The data-point types range from surprisingly hard

(`drawing_date` and `person__signature_date`) to practically solved (`period_from/to`). Interestingly, for some types the overall best submission was surpassed by other solutions (in particular for `drawing_date`).

Table 5: The results for pure information extraction

Submission	pure IE	address	company	date	period	name	position
double_big	64.1±1.4	67.1±2.2	59.5±3.9	46.5±4.4	93.8±1.4	69.4±2.4	67.7±2.5
double_small	62.8±1.4	65.9±2.3	58.9±3.6	43.2±4.3	93.4±1.5	67.4±2.5	67.0±2.6
middle_big	62.6±1.4	66.1±2.1	59.6±3.6	42.0±3.9	94.0±1.5	68.2±2.1	65.6±2.6
MBART+RF	49.2±1.3	64.3±2.4	20.5±3.2	19.5±2.6	66.3±3.6	56.9±3.1	59.8±2.9
CLEX	69.7±1.6	81.1±2.1	79.7±3.3	41.0±3.7	97.2±1.2	77.5±2.3	67.8±3.0
100_RF	61.7±1.3	66.6±2.3	59.9±3.9	46.0±4.1	82.7±2.1	67.7±1.9	64.7±2.4
300_xgb	62.8±1.3	67.6±2.1	60.9±4.1	46.3±4.0	90.7±1.7	67.2±2.1	65.4±2.3
300_RF	61.8±1.3	66.7±1.9	57.7±4.1	46.3±4.2	82.1±2.3	68.6±2.1	66.2±2.2

Finally, the results obtained for the 'relational' data points (i.e. originally expressed as tuples) are given in Table 6. They are not much different from the results for single data points, it might indicate that the relations did not introduce specific challenges.

Table 6: The results for relation extraction

Submission	all relations	person-position	person-signature
double_big	46.4 ± 2.5	55.4 ± 2.9	**37.4 ± 4.0**
double_small	43.9 ± 2.8	52.8 ± 2.5	34.7 ± 4.3
middle_big	43.0 ± 2.3	52.9 ± 2.7	33.4 ± 3.7
MBART+RF	27.1 ± 2.1	45.5 ± 3.1	8.9 ± 2.4
CLEX	**51.3 ± 2.9**	63.8 ± 2.9	**38.9 ± 4.1**
100_RF	45.2 ± 2.3	52.3 ± 2.5	**38.0 ± 3.6**
300_xgb	44.7 ± 2.2	52.8 ± 2.4	**36.8 ± 3.7**
300_RF	45.9 ± 2.3	53.9 ± 2.7	**37.6 ± 3.6**

5. Conclusions

PolEval Task 4 represents a new type of information extraction tasks, closer to business contexts. For some data points, it is still far from being resolved. It remains to be seen whether using the layout information can bring more improvement or whether a more challenging data set of this type should be produced.

6. Acknowledgements

We would like to give thanks to Dawid Lipiński from Applica.ai for his valuable analysis of available open documents and their description in the Web. Also, we are grateful to the Students of Warsaw University of Technology, named Paweł Rzepiński, Ryszard Szymański, Tymon Czarnota, who worked on gathering the dataset and preparing ground truth. We are also thankful to other students of Data Science in WUT who worked on benchmarks.

References

Graliński F., Wróblewska A., Stanisławek T., Grabowski K. and Górecki T. (2019). *GEval: Tool for Debugging NLP Datasets and Models*. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 254–262. Association for Computational Linguistics.

Koehn P. (2004). *Statistical Significance Tests for Machine Translation Evaluation*. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pp. 388–395. Association for Computational Linguistics.

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

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*.

CLEX — Information Extraction from Documents with Complex Layouts

Michał Marcińczuk, Michał Olek, Marcin Oleksy, Jan Wieczorek
(Wrocław University of Science and Technology)

Abstract

The paper presents a rule-based and knowledge-based system for extracting information from documents with a complex layout called CLEX. The system was designed to solve the PolEval 2020 Task 4. This task's goal was to extract values for a given set of fields from financial reports in PDF format. The system utilizes document layout available in the form of HOCR files obtained from the PDF documents. The document processing workflow in CLEX consists of four stages: (1) document structure preprocessing, (2) annotation of relevant text fragments based on the local context, (3) extraction of the final value based on the global document view, and (4) lemmatization and/or normalization of the values if necessary. The presented approach obtained F_1 score of 0.651 on the test set and got first place in the task. The system is available under the GPL license at `http://github.com/CLARIN-PL/clex`.

Keywords

natural language processing, information extraction, knowledge-based systems, rule-based systems, complex layout, PDF, HOCR

1. Introduction

In recent years layout-aware information extraction from structured and partially structured text documents has gained importance in many fields. In robotic process automation (RPA) information extraction tools are used to process electronic invoices (Patel and Bhatt 2020) or to import data from forms and tables into computer systems (Yin et al. 2020). In the commercial world, the rule-based systems dominate the machine learning approaches (Chiticariu et al. 2013). At the same time, some of the layout-based tasks which are very similar between different domains (like table and form parsing and understanding) can be already processed

by fully automated systems — Amazon Textract[1] or Microsoft Azure Form Recognizer.[2] More complex tasks still need some level of supervision to obtain production-level results.

For tasks based on information extraction from plain text (recognition of named entities, temporal expressions, or semantic relations), the state of the art results are being obtained by systems utilizing different types of generic language models — BERT and its variants.[3] This success was possible due to the presence of (1) generic language models trained on large volumes of text and (2) manually annotated datasets required for fine-tuning the downstream task. In 2020 several language models which are trained to jointly model interactions between text and layout information emerged — LayoutLM (Xu et al. 2020), LAMBERT (Garncarek et al. 2020) and TABERT (Wei et al. 2020). The models were tested on form and table understanding tasks and obtained, on average better results than the systems based on purely text-based language models. The improvement varied from 1–2 pp for TABERT and LAMBERT to 9 pp for LayoutLM.

2. Task description

2.1. Problem statement

The goal of the task is to extractor values for a defined set of fields from documents with a complex layout. The documents' layout is called complex because the document contains multiple pages and text sections, tables, plots, forms, etc. The set of fields includes the following categories:

— *company* — name of the company

— *drawing_date* — date which specifies when the financial report was submitted

— *period_from* — start of the obligation period

— *period_to* — end of the obligation period

— *postal_code* — postal code of the company

— *city* — the city where the company is registered

— *street* — the name of the street where the company is registered

— *street_no* — the number of the street at which the company is registered

— *people* — members/chairmen of the company management. For each entry three values should be extracted: *name* — person name, *role* — role in the company, and *sign* — signature date.

[1] `https://aws.amazon.com/textract/features`

[2] `https://azure.microsoft.com/en-us/services/cognitive-services/form-recognizer/`

[3] `https://gluebenchmark.com/leaderboard`

2.2. Datasets

Task organizers provided three datasets: training (1662 documents), developer/validation (554 documents), and test (555 documents). Each document consists of three files: PDF — the original document, TXT — text extracted from the PDF file, and HOCR — text with positional information extracted from the PDF.

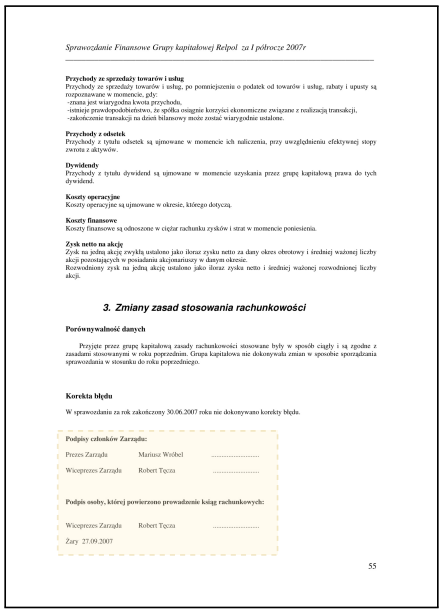
During the preliminary data analysis, we have identified the following features of the datasets:

1. There is no one standardized structure of the document. Each financial report may provide the same information in different locations (e.g., at the beginning, or the end of the document) or in different forms. For example, the *people* field may have a variety of structures (see Figure 1).

2. Field values may appear in many different formats and variants. For example, *date* can be presented in one of the following formats (the list does not contain all of the possible forms):
   — 19 września 2020
   — 19 września 2020 roku
   — 19 września 2020 r.
   — 19.09.2020
   — 19-09-2020
   — 19/09/2020
   — 2020.09.19
   — 2020-09-19

   The same applies to periods. Moreover, there are various methods of expressing the beginning and end of the reporting period. It is not necessary to point the date literally, e.g., the expression *I półrocze 2013 r.* (En. 'first half of 2013') refers to the specific period, but it doesn't contain exact dates. Consequently, there was a need not only for the normalization of time expressions but also for interpreting some brief descriptions. The problem of different ways of expressing the same (or almost the same) information concerned also the names of the companies. There are full and short names, e.g. *Zakład Budowy Maszyn „ZREMB – CHOJNICE" S.A.*, *ZBM „ZREMB – CHOJNICE" S.A.* and *„ZREMB – CHOJNICE" S.A.*.

3. The lack of a unified standard structure for reports resulted in many problems requiring normalization. The most common issues are listed below.
   (a) Different variants of *city* — some listed companies have their headquarters in small towns that do not have their own post office. This minor detail results in unclear information about the city — the seat, e.g., *Fugasówka*, *ul. Reja 4* (*street*, *street_no*), *42-440* (*postal_ code*) *Ogrodzieniec* (name of the town where the post office is located).



Sprawozdanie Finansowe Grupy kapitałowej Relpol za I półrocze 2007r

Przychody ze sprzedaży towarów i usług
Przychody ze sprzedaży towarów i usług, po pomniejszeniu o podatek od towarów i usług, rabaty i upusty są rozpoznawane w momencie, gdy:
- znana jest wiarygodna kwota przychodu,
- istnieje prawdopodobieństwo, że spółka osiągnie korzyści ekonomiczne związane z realizacją transakcji,
- zakończenie transakcji na dzień bilansowy może zostać wiarygodnie ustalone.

Przychody z odsetek
Przychody z tytułu odsetek są ujmowane w momencie ich naliczenia, przy uwzględnieniu efektywnej stopy zwrotu z aktywów.

Dywidendy
Przychody z tytułu dywidend są ujmowane w momencie uzyskania przez grupę kapitałową prawa do tych dywidend.

Koszty operacyjne
Koszty operacyjne są ujmowane w okresie, którego dotyczą.

Koszty finansowe
Koszty finansowe są odnoszone w ciężar rachunku zysków i strat w momencie poniesienia.

Zysk netto na akcję
Zysk na jedną akcję zwykłą ustalono jako iloraz zysku netto za dany okres obrotowy i średniej ważonej liczby akcji pozostających w posiadaniu akcjonariuszy w danym okresie.
Rozwodniony zysk na jedną akcję ustalono jako iloraz zysku netto i średniej ważonej rozwodnionej liczby akcji.

3. Zmiany zasad stosowania rachunkowości

Porównywalność danych

Przyjęte przez grupę kapitałową zasady rachunkowości stosowane były w sposób ciągły i są zgodne z zasadami stosowanymi w roku poprzednim. Grupa kapitałowa nie dokonywała zmian w sposobie sprezentowania sprawozdania w stosunku do roku poprzedniego.

Korekta błędu

W sprawozdaniu za rok zakończony 30.06.2007 roku nie dokonywano korekty błędu.

Podpisy członków Zarządu:

Prezes Zarządu Mariusz Wróbel .....
Wiceprezes Zarządu Robert Tęcza .....

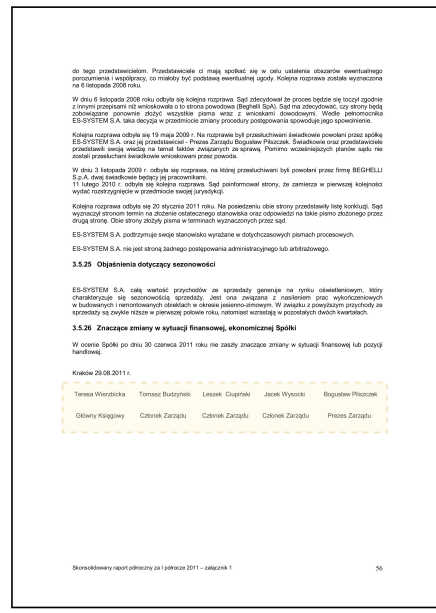
Podpis osoby, której powierzono prowadzenie ksiąg rachunkowych:

Wiceprezes Zarządu Robert Tęcza .....

Żary 27.09.2007

55

(a) Document 62096



do tego przedstawicielom. Przedstawiciele ci mają spotkać się w celu ustalenia obszarów ewentualnego porozumienia i współpracy, co miałoby być podstawą ewentualnej ugody. Kolejna rozprawa została wyznaczona na 6 listopada 2008 roku.

W dniu 6 listopada 2008 roku odbyła się kolejna rozprawa. Sąd zdecydował że proces będzie się toczył zgodnie z innymi przepisami niż wnioskowała o to strona powodowa (Beghelli SpA). Sąd ma zdecydować, czy strony będą zobowiązane ponownie złożyć wszystkie pisma wraz z wnioskami dowodowymi. Wedle pełnomocnika ES-SYSTEM S.A. taka decyzja w przedmiocie zmiany procedury postępowania spowoduje jego spowolnienie.

Kolejna rozprawa odbyła się 19 maja 2009 r. Na rozprawie był przesłuchiwani świadkowie powołani przez spółkę ES-SYSTEM S.A. oraz jej przedstawiciel - Prezes Zarządu Bogusław Pilaszek. Świadkowie oraz przedstawiciele przedstawili swoją wiedzę na tamat faktów związanych ze sprawą. Pomimo wcześniejszych planów sądu nie zostali przesłuchani świadkowie wnioskowani przez powoda.

W dniu 3 listopada 2009 r. odbyła się rozprawa, na której przesłuchiwani byli powołani przez firmę BEGHELLI S.p.A. dwaj świadkowie będący jej pracownikami.
11 lutego 2010 r. odbyła się kolejna rozprawa. Sąd poinformował strony, że zamierza w pierwszej kolejności wydać rozstrzygnięcie w przedmiocie swojej jurysdykcji.

Kolejna rozprawa odbyła się 20 stycznia 2011 roku. Na posiedzeniu obie strony przedstawiły listę korekcji. Sąd wyznaczył stronom termin na złożenie ostatecznego stanowiska oraz odpowiedzi na takie pismo złożonego przez drugą stronę. Obie strony złożyły pisma w terminach wyznaczonych przez sąd.

ES-SYSTEM S.A. podtrzymuje swoje stanowisko wyrażane w dotychczasowych pismach procesowych.

ES-SYSTEM S.A. nie jest stroną żadnego postępowania administracyjnego lub arbitrażowego.

3.5.25 Objaśnienia dotyczący sezonowości

ES-SYSTEM S.A. całą wartość przychodów ze sprzedaży generuje na rynku oświetleniowym, który charakteryzuje się sezonowością sprzedaży. Jest ona związana z nasileniem prac wykończeniowych w budowanych i remontowanych obiektach w okresie jesienno-zimowym. W związku z powyższym przychody ze sprzedaży są zwykle niższe w pierwszej połowie roku, natomiast wzrastają w pozostałych dwóch kwartałach.

3.5.26 Znaczące zmiany w sytuacji finansowej, ekonomicznej Spółki

W ocenie Spółki po dniu 30 czerwca 2011 roku nie zaszły znaczące zmiany w sytuacji finansowej lub pozycji handlowej.

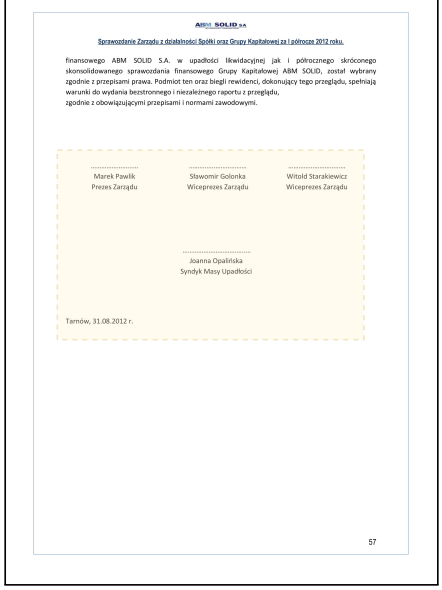
Kraków 29.08.2011 r.

Teresa Wierzbicka	Tomasz Budzyński	Leszek Ciupiński	Jacek Wysocki	Bogusław Pilaszek
Główny Księgowy	Członek Zarządu	Członek Zarządu	Członek Zarządu	Prezes Zarządu

Skonsolidowany raport półroczny za I półrocze 2011 – załącznik 1

56

(b) Document 174621



ABM SOLID S.A.

Sprawozdanie Zarządu z działalności Spółki oraz Grupy Kapitałowej za I półrocze 2012 roku.

finansowego ABM SOLID S.A. w upadłości likwidacyjnej jak i półrocznego skróconego skonsolidowanego sprawozdania finansowego Grupy Kapitałowej ABM SOLID, został wybrany zgodnie z przepisami prawa. Podmiot ten oraz biegli rewidenci, dokonujący tego przeglądu, spełniają warunki do wydania bezstronnego i niezależnego raportu z przeglądu,
zgodnie z obowiązującymi przepisami i normami zawodowymi.

Marek Pawlik
Prezes Zarządu

Sławomir Golonka
Wiceprezes Zarządu

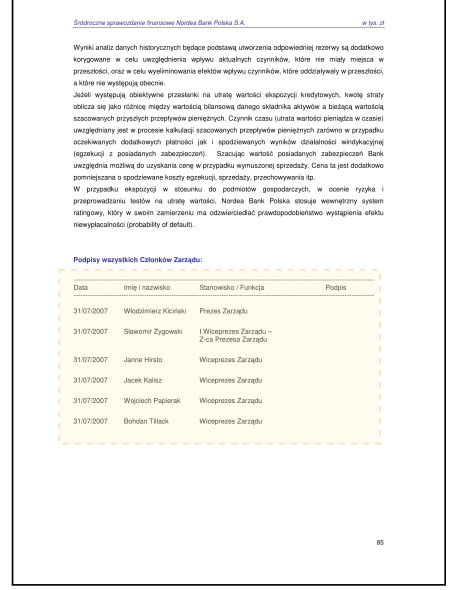
Witold Starakiewicz
Wiceprezes Zarządu

Joanna Opalińska
Syndyk Masy Upadłości

Tarnów, 31.08.2012 r.

57

(c) Document 209136



Śródroczne sprawozdanie finansowe Nordea Bank Polska S.A. *w tys. zł*

Wyniki analiz danych historycznych będące podstawą utworzenia odpowiedniej rezerwy są dodatkowo korygowane w celu uwzględnienia wpływu aktualnych czynników, które nie miały miejsca w przeszłości, oraz w celu wyeliminowania efektów wpływu czynników, które oddziaływały w przeszłości, a które nie występują obecnie.
Jeżeli występują obiektywne przesłanki na utratę wartości ekspozycji kredytowych, kwotę straty oblicza się jako różnicę między wartością bilansową danego składnika aktywów a bieżącą wartością szacowanych przyszłych przepływów pieniężnych. Czynnik czasu (utrata wartości pieniądza w czasie) uwzględniany jest w procesie kalkulacji szacowanych przepływów pieniężnych zarówno w przypadku oczekiwanych dodatkowych płatności jak i spodziewanych wyników działalności windykacyjnej (egzekucji z posiadanych zabezpieczeń). Szacując wartość posiadanych zabezpieczeń Bank uwzględnia możliwą do uzyskania cenę w przypadku wymuszonej sprzedaży. Cena ta jest dodatkowo pomniejszana o spodziewane koszty egzekucji, sprzedaży, przechowywania itp.
W przypadku ekspozycji w stosunku do podmiotów gospodarczych, w ocenie ryzyka i przeprowadzaniu testów na utratę wartości, Nordea Bank Polska stosuje wewnętrzny system ratingowy, który w swoim zamierzeniu ma odzwierciedlać prawdopodobieństwo wystąpienia efektu niewypłacalności (probability of default).

Podpisy wszystkich Członków Zarządu:

Data	Imię i nazwisko	Stanowisko / Funkcja	Podpis
31/07/2007	Włodzimierz Kiciński	Prezes Zarządu	
31/07/2007	Sławomir Żygowski	I Wiceprezes Zarządu – Z-ca Prezesa Zarządu	
31/07/2007	Janne Hirsto	Wiceprezes Zarządu	
31/07/2007	Jacek Kalisz	Wiceprezes Zarządu	
31/07/2007	Wojciech Papierak	Wiceprezes Zarządu	
31/07/2007	Bohdan Tillack	Wiceprezes Zarządu	

85

(d) Document 58528

Figure 1: Different layouts for the *people* field (marked with a dashed rectangle)

(b) Incomplete or incorrect *role* — standard role names have been abbreviated several times to a single word, e.g. *Prezes zarządu* (Chairman of the Management Board) abbreviated to *Prezes* (Chairman) also *Członek zarządu* (Member of the Management Board) to *Member*. The developer/validation data set also included additional information on other functions of the board members, e.g., *Piotr Kaźmierczak, Członek Zarządu, Dyrektor Finansowy* (Piotr Kaźmierczak, Member of the Management Board, Financial Director).

(c) Inconsistent use of the middle name — in some cases, the middle name of a board member was used in the main text of the document, but in the final part (with signatures), there was only the first name, e.g., *Dariusz Krawiec* vs. *Dariusz Jacek Krawiec*.

3. CLEX

3.1. Motivation

We decided to use a rule-based and knowledge-base approach to solve the information extraction task. We found that machine learning methods for this task will not be effective enough, as we identified the following problems:

1. **Lack of annotated dataset for the downstream task.** The task's goal is to extract values of a specified set of fields from the document content. The ground truth contained only the fields type and value without the information about the value position in the document. Datasets in this form could be used for the classification task but not the extraction task.

2. **Field values in ground truth were not present in the documents or had different form.** In order to prepare the dataset for the extraction task, we considered automatic annotation of documents based on the ground truth. We have analyzed 100 randomly selected documents and found the following problems:
   — field value was not present in the document — 75% of documents did not contain a signature date,
   — field value has a different value in the document and in the ground truth — 50% of documents had different drawing date,
   — field value had a different form in the document than in the ground truth — for instance, the ground truth contained people middle names while in the documents there were only first and last names or *vice versa*.

3. **Field values were ambiguous.** Values from ground truth could appear in the document multiple times, and not every occurrence was valid. For instance, a person's name could occur not only on the signature page but also on a list of board members or another context. The invalid matches could introduce noise to the training dataset.

Considering the quality of the ground truth and potential problems with generating a high-quality dataset for the information extraction task, we decide to use a rule-based and knowledge-based approach. This approach could be later used to create the training dataset for any machine learning approach.

3.2. System architecture

Although files in three formats are provided, we only use one format for recognition — HOCR. It contains the full text and spatial information about the relative position of text fragments to each other.

In Figure 2, we presented the system's workflow. While loading the file, we decompose document to pages, lines, and tokens. Tokens are words or punctuation marks or words and punctuation marks together. There is a preprocessing step that separates trailing punctuation marks from words if they happen to be in the same token. There are more preprocessing steps while loading the document, and they are described in Section 3.4.

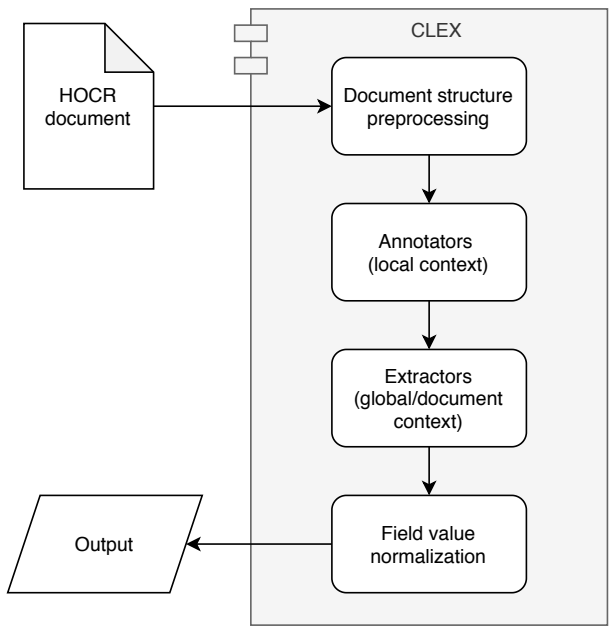


Figure 2: CLEX workflow

The actual processing is divided into three steps:

1. Local rules search for information of a specific type.
2. Global heuristics select the final information for each type of information based on all the premises.
3. If necessary, the field value is lemmatized and normalized.

All the processing stages are described in Section 3.4.

3.3. Resources

Throughout the processing, CLEX uses the following resources:

— company names in base form — the base forms were extracted from training and development sets.

— company addresses — extracted from training and development sets. The motivation was the assumption that company names and addresses change relatively infrequently. It is highly probable that the name and address used in one document are also correct for another document of a given company.

— person names — the names were obtained from NELexicon2 (Marcińczuk 2014),

— city names — the names were obtained from the Polish TERYT database. The data about towns and streets were combined, and only towns large enough to have named streets were selected. The underlying assumption was, for cities that are so small that they do not have named streets, that there is very little chance that they will be listed in the provided financial documents. In further processing, the list is even more shortened as it turned out that many cities have names identical to common words ("Dobra", "Ruda", "Szklana") leading to misclassifications. We sorted them by the number of streets they have (descending) and took only the first two thousand. Even further, cases where the name of the city coincided with the common personal name of a person (e.g., "Jarosław") have to be handled in a special way during processing documents.

3.4. Processing stages

Document preprocessing

The system performs the following preprocessing operations:

1. **Restoring order** — Generally, the order of tokens is not preserved in provided HOCR files. It means some tokens have spatial information indicating that they should be placed in the document after some other tokens that are actually placed after the first ones. Also, it happens that whole lines are not in order or incorrectly repeated multiple times. There are additional steps of preprocessing that reorganize all tokens and lines to have them stored in proper order.
2. **Separation punctuation marks** — Separate trailing punctuation marks from words if they happen to be in the same token.
3. **Handling empty leading pages** — Next, we detect the number of empty leading pages. By default, **extractors** prefer data from the first page of the document, and usually, it is enough even when there are leading empty pages in the document. But for fields *period_from* and *period_to* we needed to process precisely the real — not artificial and empty — first ('title') page of the document.

4. **Calculating line heights and its distribution** — Since we have spatial information in the HOCR file, it is possible to extract each line's height in the document and calculate its distribution per page and document. Initially, it was meant to help find lines significantly higher than the average one and identify separate sections of documents and their titles. In the end, only row height data was used. It influenced the evaluation of the *date* tokens found: the higher the row, the higher the score. They were also used when cutting footers and headers from the actual content of the document.
5. **Separating headers and footers from the document** — Headers and footers span throughout many pages, and this repetition can lead to misclassification of some tokens they contain. However, they may also include relevant information, so simply dismissing them would not be the best solution. That is why an algorithm for cutting off headers and footers and creating a separate structure from them was developed. This has proved to be a non-trivial task as headers and footers can be multi-line, and some lines may vary between pages. The same procedure is deployed separately for headers and then for footers. In the further description, we focus on processing headers. There are two stages of this procedure:
   (a) The first is to build a tree of lines that compose headers. The basic idea is that this procedure sets its range of activity as a whole document and then 'walks' from the start page to the last page of the range, checking the first line of each page. If the line of the current page is the same as the line from the previous page, it is very likely 'in' header. There is a parameter saying how many times the exact line must repeat for the procedure to decide it is 'in' header. If the line is not the same, it means it detected the end of the header processed so far. It then holds processing at the current level and 'dives in' to check if the newly found header line has some header lines below. It does it by cutting the first line from processed pages, narrowing the range of activity to contain only them, and invoking itself recursively with these parameters. It continues to 'dive in' as long as it finds new header lines on subsequent levels. If it does not find new header lines on the current level, it starts to continue its work from the last place it holds until the end of the document is reached. The actual implementation is slightly different, but the idea is kept.
   (b) This stage exists only due to technical issues. All the complex pattern-matching machinery works with document pages and not with trees of lines, so we need to flatten and 'linearize' the tree from the previous step to create a list of artificial pages with just headers and footers, and only then we can apply **annotators**.

Annotators — local context

At the lowest level, we have a set of matchers that match the given text or the given regular expression. We can specify whether attention should be paid to the case of letters and — in the case of multi-member matches — what is the maximum distance in the text between these members. As we describe in the following paragraph, the results are processed by annotators, get labeled with type, and stored per page. There are also matchers that search inside these

stored values for a specific type of results or all results but specific types. This is used to create more sophisticated rules. There are also two more that work with set or sequences of words: the first compares each encountered token with the words in the given dictionary, the second compares a sequence of two or one words against sequences in the given dictionary. The latter is used to identify the names of cities in the text.

The matchers described in the previous paragraph are combined to define much more elaborated patterns. And then sets of these patterns are used by annotators to create even more sophisticated rules that allow extracting fragments of the text that are likely to contain words or phrases that the system is looking for. This layer can be considered as the heart of the whole system. Generally, each annotator corresponds to a type of a field that needs to be extracted and is mentioned in the task description (*city, postal_code, drawing_date*). However, there are some types of fields requiring a more subtle approach e.g., there are separate annotators searching for information about persons: one searches horizontal layouts, another searches vertical layouts. Also, there is a whole family of annotators dealing with the street and house number together. The results are stored per page.

Annotators can form chains. One annotator can utilize the output of the others. This mechanism allows to decompose the extraction problem, and share information required for different fields. In Figure 3 we present annotators and dependencies between them.

Extractors — global view

In this layer we have extractors that try to extract the best result from the ones found and stored so far. There are eight different extractors corresponding to fields that need to be extracted and listed in the task description. Fields *period_from* and *period_to* are combined together inside one extractor. Similarly fields *street* and *street_no*. There is one additional extractor that is responsible for finding the signature page.

Extractors have a defined behavior depending on the type of field they are trying to find. The simplest extractor tries to find if there are results found on the first page, and if they are, then only they are later considered. If there are no such results, all of them (from all document's pages) are taken into account. These results are then sorted by score, and the first one in the document is chosen as the final result.

One extractor searches for a value that does not correspond to any field described in the task description. The extractor tries to find the page with signatures. It uses two ways to find the page. First is just by using matcher searching for the text "Podpisy" and then checking on the page that match. If the page is not selected, then the second way is used by using matcher that searches for the city's name and a string representing date positioned side by side. If the page is found, then this information is used by some other extractors that prefer results found on the page with signatures (e.g., people data).

Lemmatization and normalization

There is an additional step for fields *company* and *street*. Just before returning the result, the text is lemmatized. In the case of *company* field, first, each separate word is lemmatized,

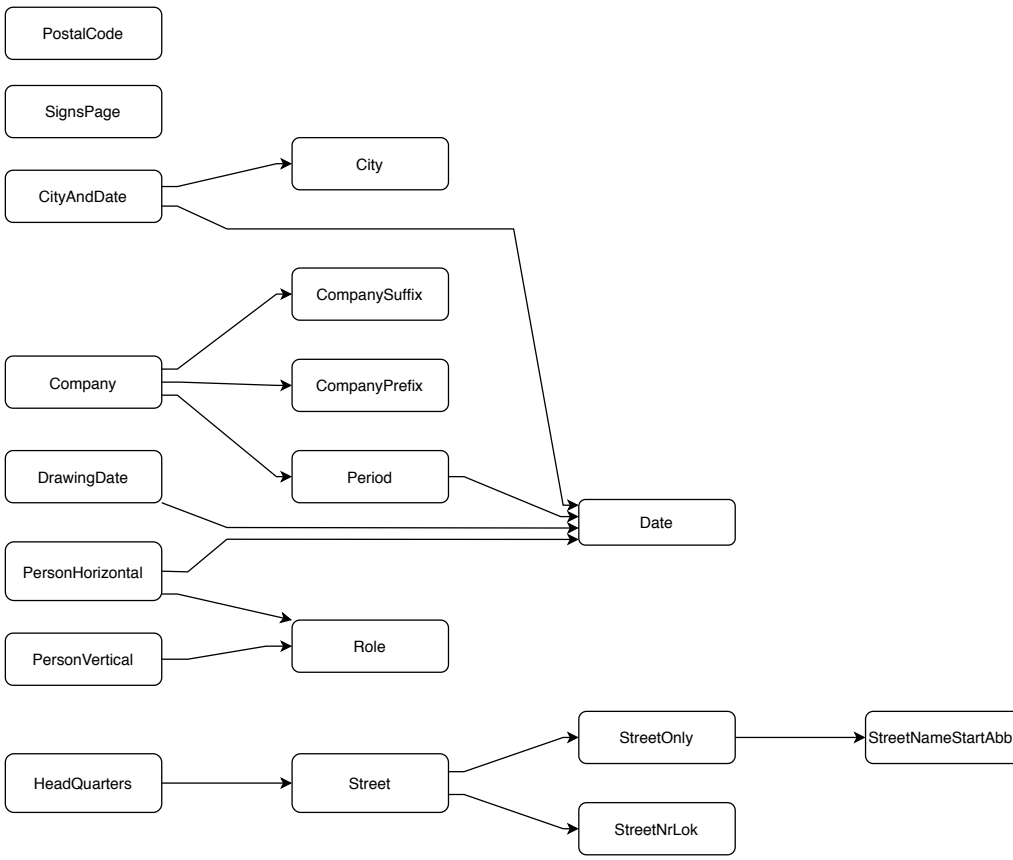


Figure 3: Chains of annotators used by CLEX

then some of these separate words may be ignored (like 'DOMINUJĄCA', 'FIRMY'), and then the full company name is lemmatized. In the case of *street* field, first, we check if the actual street name is just one word. If not, the text is not changed. If yes, the word's ending is changed according to the records in a special table containing inflected and the basic versions of endings.

Additional global heuristics

The most different from the standard result extraction procedure described in the previous chapter is the one that selects information about people. First, it checks if the page with signatures was found. If yes, then it tries to process results found on that page. If there are no such results, then it tries to find them differently than it is done in the first step: first, it finds all tokens with words denoting a person role like "prezes", "wiceprezes", "prokurent", "księgowy" etc. And for each such token, it finds the first token above and below — that's why we needed to reorganize lines and tokens properly. Then it scans patterns that match

first and last names in the nearest horizontal neighborhood of the newly found tokens. If something is found, we have a set of tokens with roles and a set of corresponding tokens with first and last names. Tokens with first and last names can be in the nearest horizontal neighborhood of two different tokens with roles. Then, disambiguation is done by comparing the Euclidean distance between geometric centers of tokens — exactly square of the distance to avoid floating-point processing overhead. In case still nothing is found, CLEX falls back to the default procedure described earlier — omitting checking results from the first page.

Not only can we introduce changes in choosing the final result by changing the algorithm but also by changing the score assigned to intermediate results. This possibility is not heavily used, but there are places taking advantage of it. When calculating the score for some types (*city, company*), a small alteration is introduced: if an intermediate result is found in the header or footer, its score is much higher than the 'standard' ones. When searching for *period_from* and *period_to* values, the highest score is assigned to intermediate results from the title page, and half of this score is assigned to result from the consecutive page. Further results are assigned significantly fewer scores — that corresponds to the height of the row in which the token was found.

3.5. Summary

The CLEX architecture itself is quite simple, but the approach's strength lies in the two-tier system of patterns and rules, which are combined in sophisticated ways. They do most of the work. Additional heuristics and preprocessing add up to the result.

Table 1: Number of patterns and matchers used to extract field (total)

Field	**annotators**	**patterns**	**matchers**
address			
– city	1	1	1
– postal_code	1	1	1
– street & street_no	4	7	37
company	3	18	54
dates			
– date	1	4	14
– drawing_date	1	3	10
– period_from & period_to	1	7	29
person	2	5	26
person_position	1	2	4

At the lowest level, we have eleven types of matchers used to compose more complicated patterns. The patterns are created and used by annotators. Then, extractors select the final results. Table 1 shows how many annotators, patterns, and matchers were used to identify each type of field. The numbers are total numbers, it means that, for example, to identify *company*, we have three annotators that contain 18 patterns, and there are 54 instances of matchers

used to build up these 18 patterns. Annotator for *date* is used by annotator for *drawing_date* and *period_from* and *period_to*. Some annotators use results of other annotators e.g. annotator for *person* uses results produced by annotators dealing with *date* and *person_position*. The number of matchers might not be perfectly accurate as matchers automatically created from multi-word strings are counted as one.

4. Evaluation

Table 2 contains detailed evaluation results on the development set. Fields *address* and *person* are compound fields and consist of the following subfields: *address*={*city*, *postal_code*, *street*, *street_no*}, *person*={*name* and *date*, *position*}. Fields *person_name_position* and *person_name_sign* are also compound fields and consist of two out of three subfields for *person*, respectively {*name*, *position*} and {*name*, *sign*}.

Table 2: Evaluation results of CLEX on the development set

Field	**F_1**	**Precision**	**Recall**
address	0.901 ± 0.016	0.901 ± 0.016	0.901 ± 0.016
– city	0.952 ± 0.017	0.955 ± 0.017	0.952 ± 0.017
– postal_code	0.907 ± 0.024	0.942 ± 0.019	0.876 ± 0.029
– street	0.900 ± 0.023	0.911 ± 0.023	0.888 ± 0.025
– street_no	0.907 ± 0.023	0.928 ± 0.022	0.889 ± 0.026
company	0.893 ± 0.027	0.904 ± 0.028	0.881 ± 0.029
drawing_date	0.437 ± 0.043	0.445 ± 0.042	0.428 ± 0.043
period_from	0.971 ± 0.013	0.974 ± 0.013	0.968 ± 0.014
period_to	0.967 ± 0.014	0.970 ± 0.014	0.964 ± 0.015
person	0.607 ± 0.024	0.597 ± 0.033	0.619 ± 0.023
– person_name	0.805 ± 0.023	0.792 ± 0.035	0.821 ± 0.024
– person_position	0.724 ± 0.030	0.712 ± 0.039	0.739 ± 0.031
– person_sign_date	0.427 ± 0.042	0.425 ± 0.046	0.432 ± 0.043
– person_name_position	0.695 ± 0.031	0.682 ± 0.039	0.707 ± 0.032
– person_name_sign	0.399 ± 0.042	0.394 ± 0.046	0.410 ± 0.043
F_1 (UC)	**0.694** ± 0.017		
F_1	0.682 ± 0.018		

CLEX obtained the highest scores for *period_from* and *period_to* — above 0.96. This field was the easiest to recognize, and the values were the most consistent between the ground truth and documents. The majority of errors are due to mistakes in the ground truth. Field *address* obtained the third-highest score of 0.90. The errors were equally distributed between system mistakes, ground truth mistakes, and differences in field forms (for instance, “Plac Trzech Krzyży” vs. “Pl. Trzech Krzyży”, “E. Plater” vs. “Emilli Plater”). Near the same score was obtained for *company* with a similar distribution of errors, including differences in field value forms (for instance, “BUDOPOL-WROCŁAW SA” vs. “BUDOPOL WROCŁAW SA”, “POLSKI KONCERN NAFTOWY ORLEN SA” vs. “ORLEN SA”).

The lowest scores were obtained for *person* and *drawing_date*. In case of *person* the most problematic was signature date which obtained 0.427 — near half the score for *name* and *position*. The main reason for the low score is that many documents were missing signature date, while in the ground truth, each record contains some value. For *position* most of the errors were related to value normalization, for example "Wice Prezes Zarządu" vs "Wiceprezes Zarządu", "Prezes Zarządu" vs "Prezes". For *drawing_date* CLEX obtained a significantly lower score than for the remaining fields, which is 0.437. The main problem with this filed was the large discrepancy between the documents' values and the ground truth.

Table 3: Evaluation results on the test set

Submission	**F_1 (UC) on test**
CLEX	**0.651** ± 0.019
double_big	0.606 ± 0.017
300_xgb	0.592 ± 0.015
double_small	0.588 ± 0.018
300_RF	0.587 ± 0.015
middle_big	0.585 ± 0.016
100_RF	0.584 ± 0.016
Multilingual BERT + Random Forest	0.440 ± 0.014

On the test set, CLEX obtained a score of 0.651, which is higher by 0.045 than the second-best score. The full leaderboard for Task 4 is presented in Table 3.

5. Summary

In the paper, we presented a system for information extraction from a complex layout called CLEX. Based on the training dataset analysis, we decided to use the rule-based and knowledge-based approach instead of machine learning due to some dataset's problems. Our system achieved a score of 0.651 on the test set, which was the highest score among the submitted solutions.

The system is available under the GPL license at `http://github.com/CLARIN-PL/clex`.

Acknowledgements

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

References

Chiticariu L., Li Y. and Reiss F. R. (2013). *Rule-Based Information Extraction is Dead! Long Live Rule-Based Information Extraction Systems!* In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 827–832, Seattle, Washington, USA. Association for Computational Linguistics.

Garncarek Ł., Powalski R., Stanisławek T., Topolski B., Halama P. and Graliński F. (2020). *LAMBERT: Layout-Aware (Language) Modeling using BERT for Information Extraction*. arXiv:2002.08087.

Marcińczuk M. (2014). *NELexicon2*. CLARIN-PL digital repository, `http://hdl.handle.net/11321/247`.

Patel S. and Bhatt D. (2020). *Abstractive Information Extraction from Scanned Invoices (AIESI) using End-to-end Sequential Approach*. arXiv:2009.05728.

Wei M., He Y. and Zhang Q. (2020). *Robust Layout-aware IE for Visually Rich Documents with Pre-trained Language Models*. arXiv:2005.11017.

Xu Y., Li M., Cui L., Huang S., Wei F. and Zhou M. (2020). *LayoutLM: Pre-training of Text and Layout for Document Image Understanding*. „Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining".

Yin P., Neubig G., tau Yih W. and Riedel S. (2020). *TaBERT: Pretraining for Joint Understanding of Textual and Tabular Data*. arXiv:2005.08314.

BiLSTM Recurrent Neural Networks in Heterogeneous Ensemble Models for Named Entity Recognition Problem in Long Polish Unstructured Documents

Aleksandra Świetlicka (Institute of Automatic Control and Robotics, Poznań University of Technology; Wrocław Institute of Spatial Information and Artificial Intelligence)

Adam Iwaniak (Wrocław Institute of Spatial Information and Artificial Intelligence; Institute of Geodesy and Geoinformatics, Wrocław University of Environmental and Life Sciences)

Mateusz Piwowarczyk (Faculty of Computer Science and Management, Wrocław University of Science and Technology; Wrocław Institute of Spatial Information and Artificial Intelligence)

Abstract

In this paper we present a solution to the Task 4 of the PolEval 2020 evaluation campaign for natural language processing tools for Polish. This task was to extract information and type entities from long Polish documents with complex layouts. As a solution to this problem we propose two machine learning algorithms. The first one is an Artificial Neural Network (ANN) composed from LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Unit) layers. The second one was an ensemble model based on ANN, random forests and XGBoost.

Keywords

natural language processing, named entity recognition, ensemble model, artificial neural network, BiLSTM, GRU, XGBoost, Random Forest

1. Introduction

Natural Language Processing (NLP) is one of the most interesting, developing and yet challenging area of research, which is undertaken by many scientists from around the world. Among the issues of NLP we can find text classification, named entity recognition (NER), sentiment analysis (usually based on tweets, IMDb film reviews, amazon products reviews, etc.), text generation including caption generation, language translations, etc.

Polish belongs to the group of inflectional languages, thus it is much more complicated and more difficult to machine processing than, for example, English. While for English there are many very well developed machine learning models, e.g. Transformers (Wolf et al. 2019), i.e. BERT (Vaswani et al. 2017), RoBERTa (Liu et al. 2019), XLNet (Yang et al. 2020) or SpaCy (Honnibal and Montani 2017), for Polish it is usually possible to use only multilingual options. SpaCy model for Polish has been actually developed in the past few years (Tuora and Kobyliński 2019), and there were unofficial versions of these models, but it was officially accepted in the `SpaCy` package only in July of 2020.

In this research we are facing a problem of information extraction and entity typing from long Polish documents with complex layouts, which was one of the PolEval 2020 tasks. Documents are financial reports of Polish corporations, which contain unstructured arrangement of text such as very long tables, space for signatures, headers and footers, two-column format, etc. For each of the reports there were three files available: PDF – the original report, TXT – optical character recognition (OCR) of the report and HOCR – data representation for formatted text obtained from OCR. In our solutions we have focused only on TXT files, which we have noticed were very "tidy" and quite easy to process and prepare for further analysis with machine learning algorithms. Additionally to each of the provided by organizers sets (except from the evaluation one) there were provided so called ground truth tables, which contained data that should be extracted from reports. Among extracted data it is possible to find: name of the organization, address of the organization (separately city, street and street number), names of people managing the organization with their positions, and dates (date and time scope of the report).

Reviewing related to NER papers and articles it is possible to find many approaches to this problem. It is rather obvious that finding e.g. a postal code might be as simple as building a proper regular expression rule to extract the preferable data. If there is only one postal code in the document, then the situation is obvious, but what with the situation when there are several different postal codes in the document – which one should we choose? On the other side, there are already tools for names extraction, like NLTK Python package (Bird et al. 2009). But what if there are several or over a dozen or even several dozens of names in one document and we need just a few of them? How to choose which of them are relevant? All of the above ambiguities led us to design our own models as a solution to a given problem.

Another option for solving the NER problem may also be to look for already described in the literature machine learning algorithms, like conditional random fields (CRF) (Song et al. 2019) or artificial neural networks (Lample et al. 2016, Chiu and Nichols 2016) and many others (Yadav and Bethard 2019, Anandika and Mishra 2019). Unfortunately, the problem

is that the proposed solutions are often dedicated to specific types of documents and it is difficult to generalize such solutions to any texts.

In this research we present our approach to the information extraction problem which is based on artificial neural networks (ANNs) and ensemble models. The latter ones, built on the basis of neural networks (ANNs as weak learners), have been the subject of researchers for many years (Hansen and Salamon 1990, Perrone and Cooper 1995). In more recent papers it is possible to find ensemble models specifically for NER problem (Speck and Ngomo 2014, Won et al. 2018). Due to the specificity of the issue and the unique nature of the reports that we consider, we decided to design our own solutions. The first presented solution is naturally based on artificial neural networks constructed from Long Short-Term Memory (LSTM) layers, which are known for their wide area of applications in NLP problems. As an extension of this solution we have decided to combine ANN with random forests as weak learners and use another random forest or XGBoost as the ensemble one.

The paper is organized as follows. In Section 2 we present preprocessing of reports' texts and details of proposed machine learning algorithms for information extraction. In Section 3 we are briefly presenting results of our solutions, while in Section 4 we give final remarks.

2. Data and methods

Polish NLP problems have been in the area of our not only scientific but also commercial interests, for a long time. Our research is related to various branches of the NLP issues. Our recent work for example has concerned the classification of Polish land areas according to their future use, based on spatial development plans (Kaczmarek et al. 2020). Lately we have also worked on NER problem in legal documents. The PolEval 2020 contest gave us then an opportunity to test our existing solutions on new documents with different layouts.

In this specific task of information extraction from long Polish documents with complex layouts we are in fact facing a problem of so-called Multi-Word Expressions (MWE; Sag et al. 2002). We do not know how many words are there in a single entity, and the list of people managing the organization with their positions seems to be the most tricky issue, as we do not exactly know how many people are we actually looking for. Hence, the biggest challenge was to pre-process the data and prepare it for further processing by selected machine learning methods, especially since the considered documents differ from those we have dealt with so far.

In this section the arduous process of data preparation and machine learning methods that we have developed in our hitherto projects are described.

2.1. Data preparation – labels

In the first step of this part of our solution, we focused on transformation of text into "readable" for artificial neural networks form. As can be expected we have decided to use a Tokenizer, which is a dictionary that assigns to each word in a text an unequivocal integer number. After

many experiments we have decided to expand this dictionary with new words. The first idea was to add a base of Polish female and male names and surnames, which is available online,[1] and a base of Polish postal codes with their associated cities and street names – also available online.[2] Of course we also thought about adding all of the Polish words (e.g. extracted from `word2vec` models), but this idea did not improve our solution.

Tokenization was carried out with the assumption that all the standard special characters are removed (e.g. $, @, #). During browsing both `txt` files and extracted entities we have noticed that there are many special characters which are not included in the standard filters but they occur directly in names. The reason is probably as obvious as that the special characters occur in text due to the OCR process. We then expanded our Tokenizer with new special characters, e.g. •, ♦, §, etc.

The second idea of improvement of our solution was to expand not only the dictionary but also a structure of the designed artificial neural network (which, in detail is described further in the paper), which was at the end built from two threads, where one processed the original text while the second one the same text but each of its words was transformed to its basic form. To extract these basic forms of words we used an inflectional analyzer and generator for Polish morphology *Morfeusz2* (Woliński 2014).

The basic forms of words turned out to be very useful not only during labeling of the given text but also during extraction of entities from text. It is very important to notice that we are dealing with Polish which is full of declensions of nouns, adjectives, adverbs, and counting words, e.g. in Table 1 there is a conjugation of the word "eat" (pol. *jeść*) in English and Polish.

Table 1: Conjugation of the word "eat" (Pl. *jeść*) in English and Polish

ENGLISH: EAT
eat, ate, eaten, eats, eating
POLISH: JEŚĆ
jeść, jem, jesz, je, jemy, jecie, jedzą, jadłem, jadłam, jadłeś, jadłaś, jadł, jadła, jadło, jedliśmy, jedliście, jedli, jadłyśmy, jadłyście, jadły, jedz, jedzmy, jedzcie, jadłbym, jadłbyś, jadłby, jadłabym, jadłabyś, jadłaby, jadłoby, jedlibyśmy, jedlibyście, jedliby, jadłybyśmy, jadłybyście, jadłyby, jedzący, niejedzący, jedząca, niejedząca, jedzące, niejedzące, jedzony, jedzona, jedzone, jedzeni, ...

By looking in text for a specific name (e.q. Janusz Malinowski) we can fail just because there is no exact match, but there is its inflected form (e.g. Januszowi Malinowskiemu). Basic forms then helped us find the specific desirable word, but also during extraction saving the proper basic forms of words. Polish can be however very tricky: with names, companies or cities, we wish to save their basic forms, while with streets it is not so obvious. In Table 2 we are showing some examples of text sequences with their labels, to show how differently labels can be assigned. Types of labels are explained further in text.

[1] `https://dane.gov.pl`

[2] `https://www.poczta-polska.pl`

Table 2: An example of labeling of text

The original text in English	**Tokens (original text in Polish)**	**Label**	**Lemmata (from Morfeusz2)**	**Label**	**What should be extracted**
Marie	ul	o	ul	o	street:
Skłodowska-	Marii	street_start	Maria	o	ul. Marii
Curie	Skłodowskiej	street_continue	Skłodowska	o	Skłodowskiej
Street	Curie	street_continue	Curie	o	Curie
Bank	Bank	company_start	Banek	o	company: Bank
Ochrony	Ochrony	company_continue	Ochrona	o	Ochrony
Środowiska	Środowiska	company_continue	Środowisko	o	Środowiska
S.A.	S.A.	o	S.A.	o	S.A.
... together	wraz	o	wraz	o	people: Jerzy
with the	z	o	z	o	Wiśniewski
president	Prezesem	o	Prezes	position_start	Prezes
of the board	Zarządu	o	Zarządu	position_continue	Zarządu
Jerzy	Jerzym	o	Jerzy	human_start	
Wiśniewski ...	Wiśniewskim	o	Wiśniewski	human_continue	

The preparation of the output vectors for an artificial neural network required a thorough analysis of the texts in comparison with the available ground truth tables.

On this basis, the following labels have been prepared:

— street_start, street_continue, street_no
— company_start, company_continue
— drawing_date_day, drawing_date_month, drawing_date_year
— period_from_day, period_from_month, period_from_year
— period_to_day, period_to_month, period_to_year
— postal_code_pre, postal_code_post
— city_start, city_continue, city
— human_start, human_continue, position_start, position_continue

In addition to the above, a label was added conventionally called "o", which meant that a given word/token is from our point of view irrelevant.

Each of the above categories was separately analyzed:

— Each of the named entities, like company, street, people, were considered to be longer than one word, that is why we decided to use *name*_start to indicate the beginning of this name in text and *name*_continue as its continuation. We have then noticed that in the training set, there was no city names longer than one word, that is why we added the label city.

— Because each of the considered companies was a joint-stock company, they all had an adequate “postscript”: *SA*, *S.A.* or *Spółka Akcyjna*. That is why we have decided not to search for this sequence in text, but only for the basis of the full name and after extraction of this basis we added this “postscript” manually.

— Each date (drawing_date, period_from, period_to) was separated into three tokens: day, month and year. In Poland we use two basic formats of writing dates: `dd.mm.yyyy` and `dd month yyyy`. Additionally, in the second format, month should be written in genitive, while it is a common mistake to write it in nominative (e.g. correct: *13 marca 2003 r.* and incorrect *13 marzec 2003 r.*). Anyway, both these versions were considered during labeling. A problem with dates was also that often the period covered by a given report was not written explicitly with dates, but descriptively, e.g. the first half of 2003 or second quarter of 2003, etc. Due to the large variety, we did not undertake the extraction of dates from such period descriptions, thus it can be one of the considered extensions of this work.

— Postal codes were also considered as two separate tokens, due to the aforementioned tokenization, which used "-" (hyphen) and "–" (dash) as filters.

2.2. Data preparation – choosing training data

After assigning labels to each word in the given text we were ready to divide our data into train and validation sets. We have decided that as the input of a machine learning algorithm we will use a sequence of 15 words (represented as a vector of integer numbers), while there was one output, which returned a class of the middle word of the input vector. Summarizing the number of training vectors: a sequence of 15 words and an according class of its middle word, it was obvious that classes are highly unbalanced. The biggest class was the one with irrelevant data noted as "o". Without any extra oversampling we have decided to choose all data with important labels and choose a random subset of data with label "o" which cardinality was equal to the sum of the cardinalities of each of the remaining classes.

We have performed many experiments on artificial neural networks and then created our own ensemble model based on one chosen structure of an artificial neural network, random forest (RF) and XGBoost classifier. We had three, provided by organizers of the contest, data sets: train, validate and test, which contained 1662, 554 and 555 reports, respectively. For the purposes of this article, we have combined train and validate sets and extracted desirable training vectors (according to the procedure described in section 2.1). Ultimately, the number of training vectors was over 3.5 million. The first test that involved only neural networks assumed classic division of training data into train and validation sets (e.g. 75% vs. 25%), while the tests were performed on the data provided by the PolEval 2020 organizers (noted as `validation` one). In the expanded model – the ensemble one – we have used the same data set but then split it into two subsets (for both token and basic versions of the input):

1. the first subset was used to train the ANN and random forests sub-models and it contained 65% vectors (approx 2.3 million) of the whole set – let us denote pairs of this set as (X_1, y_1);
2. from the second subset, which contained 35% vectors (approx. 1.25 million), let us denote it as (X_2, y_2), X_2 was used to make a prediction on ANN and RFs obtaining three vectors $y_{2\text{pred}}$ for each of the sub-models: $y_{2\_\text{pred}\_\text{ANN}}, y_{2\_\text{pred}\_\text{RF}\_\text{token}}, y_{2\_\text{pred}\_\text{RF}\_\text{basic}}$;
3. finally pairs $\left(y_{2\_\text{pred}}, y_2\right)$ were used to train concatenating sub-models XGBoost.

2.3. Artificial neural networks

As we already mentioned we have performed many experiments on different structures of artificial neural networks (ANNs) and finally focused on two of them. The first one assumed that as the input we will consider only the original text (see Fig. 1A) while the second one – both, original text and the one where each word is transformed to its basic form (see Fig. 1B). Let us denote the first one as **middle** and the second one as **double**.

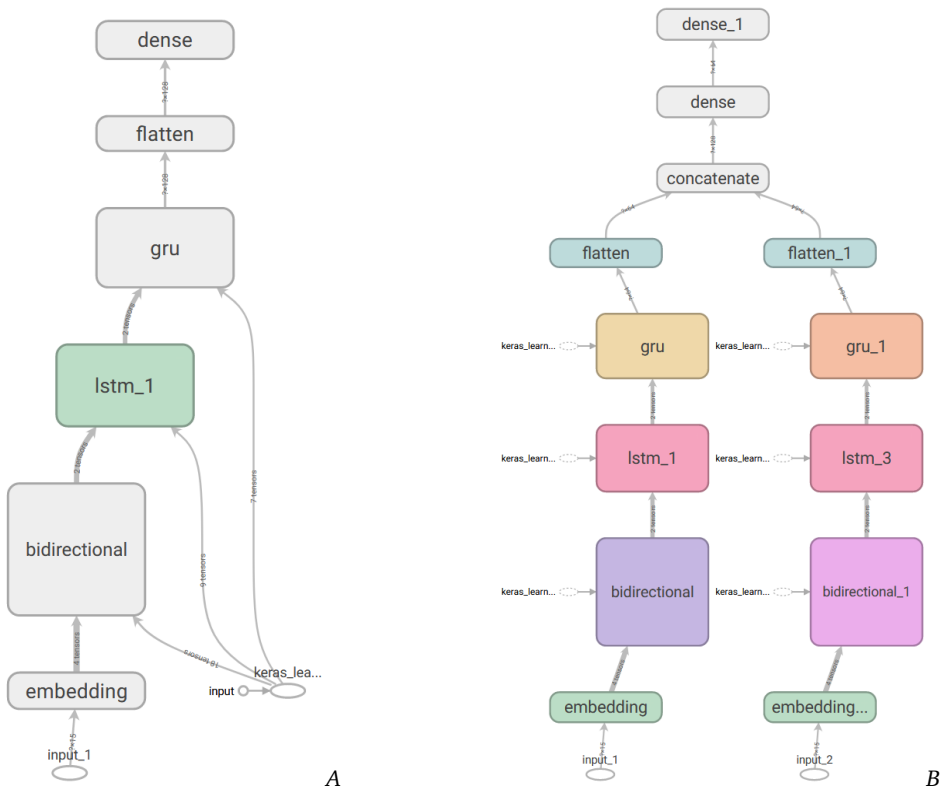


Figure 1: Structures of considered ANNs: A. model with one thread, that process only the original text (**middle**), B. model with two threads, that process text with words in their original and basic forms (**double**)

Both structures were built from the embedding as the first layer. Embedding is a matrix, in which each row correspond to a different word and it assigns to it an n-dimensional vector of real numbers, thereby placing each word in n-dimensional space, thus representing an abstract distance between the words. It is possible to find various versions of embedding matrices, among the most popular there are those provided by the Institute of Computer Science, Polish Academy of Sciences (IPI PAN),[3] and Facebook fastText (Joulin et al. 2016a,b).

[3]Over 100 different models available at `http://dsmodels.nlp.ipipan.waw.pl`

Both threads of the ANN that takes two vectors as inputs (token and basic) are actually of the same structure as the ANN with one thread. The difference occurs after GRU (Gated Recurrent Unit) layer, where one ANN goes to the output Dense layer, while the other one concatenate two threads and two additional Dense layers are added. Embedding layers in both ANN structures are followed sequentially with bidirectional Long Short-Term Memory (BiLSTM), LSTM and GRU layers. LSTM, BiLSTM and GRU layers were built from 128 neurons, with default set `tanh` activation and `sigmoid` recurrent activation functions. The penultimate Dense layer in ANN with two threads was implemented with 64 neurons and `relu` activation function. The last Dense layer in both structures had k outputs, where k denotes the number of classes – `Keras` built-in `to_categorical` function converts a class vector of integers to a binary class matrix. This last layer was implemented with `softmax` activation function. Both models were compiled with mean squared error (MSE) loss and RMSprop optimizer.

2.4. Ensemble model with XGBoost and Random Forest

Ensemble machine learning model is a model consisting of multiple individual models, the results of which are aggregated in order to achieve greater prediction accuracy than for a single model (Dietterich 2000). Theory behind ensemble learning is based mostly on the bias-variance-covariance decomposition. It is an extension of the bias-variance decomposition, for linear combinations of models. Bias component describes how accurate the model is, on average across different possible training data sets. The variance component describes how sensitive the learning algorithm is to small changes in the training data set.

Previous research (Dietterich 2000, Tang et al. 2006, Mendes-Moreira et al. 2012) showed that one of the most important characteristics of accurate ensemble model is diversity of base learners and training data. This diversity can be achieved in many different ways, and can be measured differently according to the approach of achieving it. There are three main approaches for achieving diversity in ensemble methods: **data diversity** (bagging, random forests, random subspace - random subsets of the full feature space), **parameter diversity** (multiple kernel learning), **structural diversity** (heterogeneous, homogeneous, hybrid).

The main challenges facing the methods of creating ensemble machine learning models are methods for aggregating individual classifiers within a model ensemble, selection of appropriately diverse and complementary classifiers included in the models' ensemble, selection of final aggregation technique of single classifiers, preventing overfitting.

The conventional ensemble methods include:

— **Bagging** – Builds many individual classifiers based on randomly selected samples from the original training data. Results of these multiple classifiers are combined using average or majority voting.

— **Boosting** – Iterative technique in which algorithm adjust the weight of a single training example based on the last classification. If training example was classified incorrectly, it tries to increase the weight of this single example or decrease it if it was classified correctly.

— **Stacking** – A technique in which another learner (meta-model) is used to combine output from individual base models of an ensemble. This technique performs well in building ensemble learners for problems where different types of base models are capable to learn some part of the problem, but not the whole space of the problem.

As we already mentioned in the first attempts we have analyzed many structures of artificial neural networks (ANNs). After many tests we have decided to expand our model and design an ensemble one, which apart of the ANN took also random forests and XGBoost classifiers into consideration (see Fig. 2).

Random Forest is an ensemble supervised machine learning method that fits a number of decision tree classifiers. Its great advantage is that these classifiers are trained on various sub-samples of the training data and uses averaging to improve the accuracy and control over-fitting at the same time.

XGBoost (Chen and Guestrin 2016) is a Python open-source package that provides a high-performance implementation of gradient boosted decision trees. XGBoost models are also ensemble ones, but they differ from the "classic" approach, while their ensemble decision trees are trained one by one until the error of the model does not improve anymore.

As it is shown in Figure 2 ANN took as an input vectors of words in both configurations: created from original texts and words transformed into their basic forms. We also implemented two random forests, one of them was used to process vectors of words in their original forms, while the other one – in their basic forms. At the position where XGBoost is inserted, we also considered a random forest classifier.

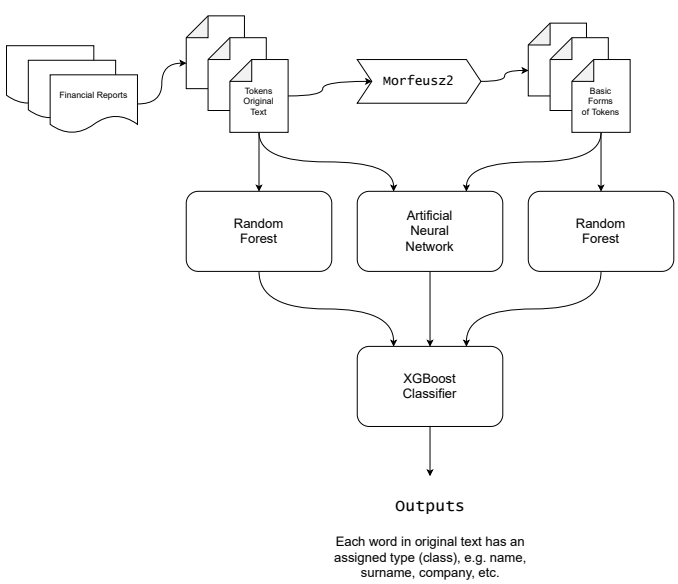


Figure 2: General idea of an ensemble model used for information extraction

3. Results

The evaluation of the NER results involves the assessment of the correct recognition of the type of entity and the correct recognition of the word boundaries in which the entity is located. To evaluate the accuracy of entity recognition algorithms, hand-tagged sets are usually used as a reference to the results obtained from the algorithms. There are two ways to compare the results: exact match and relaxed match. In the first approach, a correctly recognized result is one in which there is an entity type match and an exact match as to the word boundary (e.g. the address *Generała Władysława Sikorskiego 34/5* recognized as *Sikorskiego 34/5* will not be considered correctly detected). To measure effectiveness, measures known from classification problems are used: Precision, Recall, F-Score. In the second approach, some blurring is allowed within the recognized expression. Recognition can be passed if the entity is included in the recognized string of words, regardless of their length.

In this section we will focus on results obtained with particular models of ANN, random forests and then proceed to the results of the ensemble one, where the exact match method of comparison with F_1 score was used.

We have tested a few different configurations of random forests classifiers, and some of the results are exposed in Table 3. The `oob_score` is a score of the training data set obtained using an out-of-bag estimate. `PS` abbreviation stands for prediction score and it is an accuracy of the prediction on the validation set (denoted earlier in the paper as (X_2, y_2)). Analyzing values presented in Table 3 we could conclude that increasing the number of estimators did not improve the quality of the classifier, but it significantly extended the time of training and prediction. That is why we have decided to choose 200 estimators for further tests.

Table 3: Prediction score PS and oob_score for random forests sub-models of the ensemble model

Input version	n_estimators	oob_score	PS
token	100	0.9551	0.9959
	200	0.9555	0.9559
	500	0.9956	0.9558
basic	100	0.9586	0.9598
	200	0.9594	0.9600
	500	0.9597	0.9600

Figure 3 shows accuracy and loss plots for considered ANN models. Figures 3A and 3C concern the ANN structure with one thread (**middle**), while 3B and 3D – with two threads (**double**). Please notice that waveforms are drawn starting from the third epoch, while drawing all of the epochs made the plots unreadable. We have considered both ANN models with different embedding matrices: *nkjp* abbreviation stands for The National Corpus of Polish (Pol. Narodowy Korpus Języka Polskiego; Przepiórkowski et al. 2012) and refers to models shared by IPI PAN, while *fb* – for Facebook fastText model. From over 100 models provided by IPI PAN we have chosen two ones: with 100 and 300 size of the embedding matrix, while Facebook fastText provide embedding the matrix of dimension 300.

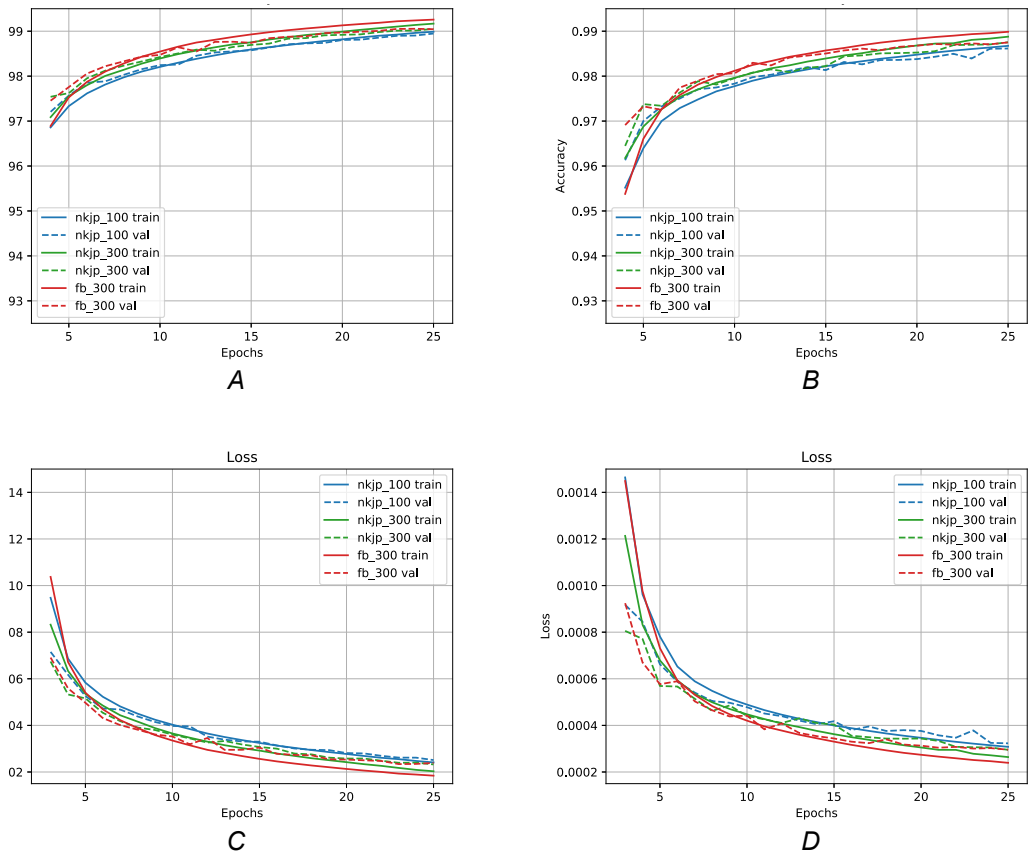


Figure 3: Line plots of accuracy and MSE loss over training epochs for both structures of ANNs considered in this paper: A. Accuracy of **middle** model, B. Accuracy of **double** model, C. Loss of **middle** model, D. Loss of **double** model

Analyzing Fig. 3 we can see that in fact the model with one thread has better performance of training than the model with two threads, although both models appeared as well qualified. Although based on these plots it seems that the first one should provide better results than the second one, we tried to empirically choose the right model. Our tests consisted primarily of manual viewing of final tables prepared on the basis of the results obtained from the neural network and of capturing irregularities. These irregularities appeared in the text as unknown entities (field of table was filled with "Unknown" word) or as tokens not recognized by the tokenizer (unknown for tokenizer words were denoted as "__Unknown__").

In Table 4 we are presenting F_1 score calculated separately for each entity, with use of the test set provided by organizers of the contest. The last entity – "people" – was separated into names and positions and F_1 score was also calculated separately for each of them. Analyzing Table 4 it is possible to notice that F_1 score varies between 0.41 and 0.98, depending on kind of the entity. Apart from that we can observe that within a single entity type each of the models gave similar value of the F_1 score.

Table 4: Number of unrecognized fields in the final result. Models **middle** and **double** refer to ANNs, RF and XGBoost refer to ensemble models with RF and XGBoost as the concatenating one, respectively. Numbers 100 and 300 refer to the size of the embedding layer, while NKJP and FB to its source.

Entity	**Model**						
	middle 100 NKJP	**middle 300 NKJP**	**double 100 NKJP**	**double 300 NKJP**	**RF 100 NKJP**	**RF 300 NKJP**	**XGBoost 300 FB**
company	0.5982	0.6054	0.4126	0.4432	0.6000	0.5784	0.6090
drawing date	0.4667	0.4631	0.4739	0.4667	0.4687	0.4739	0.4767
period from	0.8991	0.9027	0.8883	0.8865	0.8162	0.8054	0.8667
period to	0.9838	0.9784	0.9802	0.9838	0.8378	0.8378	0.9495
postal code	0.6216	0.6216	0.6450	0.6486	0.6378	0.6288	0.6432
city	0.7712	0.7585	0.7748	0.7982	0.7495	0.7441	0.7531
street	0.4955	0.4793	0.4964	0.4983	0.5063	0.5009	0.5063
street no	0.6252	0.6162	0.6450	0.6324	0.6594	0.6775	0.6793
people names	0.7132	0.6975	0.6539	0.6500	0.7513	0.7585	0.7430
people positions	0.8059	0.8020	0.8139	0.8182	0.8292	0.8396	0.8349

Summarizing our research, we have submitted 6 models to the PolEval 2020 contest, where the first attempt assumed only ANNs (3 models) and the second one – ensemble ones (3 models). In Table 5 we are showing the evaluation scores presented by the organizers of the contest.

A full description of each of the submitted models is as follows:

— double_big → ANN with two threads and embedding of size 300 (NKJP)
— 300_xgb → ensemble with XGBoost and embedding of size 300 (FB)
— double_small → ANN with two threads and embedding of size 100 (NKJP)
— 300_RF → ensemble with RF and embedding of size 300 (NKJP)
— middle_big → ANN with one thread and embedding of size 300 (NKJP)
— 100_RF → ensemble with RF and embedding of size 100 (NKJP)

Table 5: Evaluation scores

Model configuration	**F_1 score**
double_big	0.606 ± 0.017
300_xgb	0.592 ± 0.015
double_small	0.588 ± 0.018
300_RF	0.587 ± 0.015
middle_big	0.585 ± 0.016
100_RF	0.584 ± 0.016

By reviewing the results presented in Table 4 and 5, it can be concluded that the calculation of the F_1 score may not be consistent with organizers' methods. We do not know for example how was considered the joint-stock company annotation (*Spółka Akcyjna*), while it can be

written in the following forms: *SA, S.A., Spółka Akcyjna*. On the other hand, we have noticed that not all of the desired entities appeared in documents.

4. Discussion and conclusion remarks

Analysing our results, trying to improve them, while browsing available documents we have noticed that not every entity appears in the considered financial report. This fact, together with the notion that we are dealing with entities of the MWE type and due to the specificity of Polish, makes the presented issue of NER very complicated. It is necessary to remember, that Polish is an inflectional language and the task of named entity recognition is much more complicated than e.g. for English. In the literature it is possible to find F_1 values of 70–95%, however it should be emphasized that these studies are limited to carefully selected categories of NER (usually the ones that give the best results), and the algorithms use frequently hand-created rules and lexicons.

Additionally, it is possible to observe that there is an imbalance in data sets, while specific entities appear in text just a few times. While in other tasks, e.g. image processing, it is possible to enlarge the training set, in NLP problems it is rather impossible. We believe then, that more documents (financial reports) and ultimately more training vectors would improve the evaluation results of our solutions. It can be noticed, however, that the ANN and ensemble models achieved very promising results. We are constantly working on extending the proposed solutions and we can confidently say that the results have already improved.

Acknowledgements

This work was supported by The National Centre for Research and Development in Poland under the POIR 01.01.01-00-1274/17-00 project *Building a knowledge base on real estate*.

References

Anandika A. and Mishra S. P. (2019). *A Study on Machine Learning Approaches for Named Entity Recognition*. In *2019 International Conference on Applied Machine Learning (ICAML)*, pp. 153–159.

Bird S., Loper E. and Klein E. (2009). *Natural Language Processing with Python*. O'Reilly Media Inc.

Chen T. and Guestrin C. (2016). *XGBoost: A Scalable Tree Boosting System*. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2016)*, pp. 785–794. ACM.

Chiu J. P. and Nichols E. (2016). *Named Entity Recognition with Bidirectional LSTM-CNNs*. „Transactions of the Association for Computational Linguistics", 4, p. 357–370.

Dietterich T. G. (2000). *Ensemble Methods in Machine Learning*. In *Proceedings of the First International Workshop on Multiple Classifier Systems (MCS 2000)*, pp. 1–15. Springer.

Hansen L. K. and Salamon P. (1990). *Neural network ensembles*. „IEEE Transactions on Pattern Analysis and Machine Intelligence", 12(10), p. 993–1001.

Honnibal M. and Montani I. (2017). *spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing*.

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

Joulin A., Grave E., Bojanowski P., Douze M., Jégou H. and Mikolov T. (2016b). *FastText.zip: Compressing Text Classification Models*. arXiv:1612.03651.

Kaczmarek I., Iwaniak A., Świetlicka A., Piwowarczyk M. and Harvey F. (2020). *Spatial Planning Text Information Processing with Use of Machine Learning Methods*. „ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences", VI-4/W2-2020, p. 95–102.

Lample G., Ballesteros M., Subramanian S., Kawakami K. and Dyer C. (2016). *Neural Architectures for Named Entity Recognition*. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 260–270. Association for Computational Linguistics.

Liu Y., Ott M., Goyal N., Du J., Joshi M., Chen D., Levy O., Lewis M., Zettlemoyer L. and Stoyanov V. (2019). *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. arXiv:1907.11692.

Mendes-Moreira J., Soares C., Jorge A. M. and Sousa J. F. D. (2012). *Ensemble Approaches for Regression: A Survey*. „ACM Computing Surveys", 45(1), p. 1–40.

Perrone M. P. and Cooper L. N. (1995). *When Networks Disagree: Ensemble Methods for Hybrid Neural Networks*, pp. 342–358. World Scientific.

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

Sag I. A., Baldwin T., Bond F., Copestake A. and Flickinger D. (2002). *Multiword Expressions: A Pain in the Neck for NLP*. In Gelbukh A. (ed.), *Computational Linguistics and Intelligent Text Processing*, pp. 1–15. Springer Berlin Heidelberg.

Song S., Zhang N. and Huang H. (2019). *Named Entity Recognition Based on Conditional Random Fields*. „Cluster Computing", 22, p. 5195–5206.

Speck R. and Ngomo A.-C. N. (2014). *Ensemble Learning for Named Entity Recognition*. In Mika P., Tudorache T., Bernstein A., Welty C., Knoblock C., Vrandečić D., Groth P., Noy N., Janowicz K. and Goble C. (eds.), *Proceedings of the 13th International Semantic Web Conference (ISWC 2014)*, vol. 8796 of *Lecture Notes in Computer Science*, pp. 519–534. Springer International Publishing.

Tang E. K., Suganthan P. N. and Yao X. (2006). *An Analysis of Diversity Measures*. „Machine learning", 65(1), p. 247–271.

Tuora R. and Kobyliński Ł. (2019). *Integrating Polish Language Tools and Resources in Spacy*. In *Proceedings of PP-RAI 2019 Conference*, pp. 210–214. Department of Systems and Computer Networks, Faculty of Electronics, Wrocław University of Science and Technology.

Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser L. and Polosukhin I. (2017). *Attention Is All You Need*. arXiv:1706.03762.

Wolf T., Debut L., Sanh V., Chaumond J., Delangue C., Moi A., Cistac P., Rault T., Louf R., Funtowicz M., Davison J., Shleifer S., von Platen P., Ma C., Jernite Y., Plu J., Xu C., Scao T. L., Gugger S., Drame M., Lhoest Q. and Rush A. M. (2019). *HuggingFace's Transformers: State-of-the-art Natural Language Processing*. arXiv:1910.03771.

Woliński M. (2014). *Morfeusz Reloaded*. In Calzolari N., Choukri K., Declerck T., Loftsson H., Maegaard B., Mariani J., Moreno A., Odijk J. and Piperidis S. (eds.), *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC 2014)*, pp. 1106–1111. European Language Resources Association.

Won M., Murrieta-Flores P. and Martins B. (2018). *Ensemble Named Entity Recognition (NER): Evaluating NER Tools in the Identification of Place Names in Historical Corpora*. „Frontiers in Digital Humanities", 5, p. 1–12.

Yadav V. and Bethard S. (2019). *A Survey on Recent Advances in Named Entity Recognition from Deep Learning Models*. arXiv:1910.11470.

Yang Z., Dai Z., Yang Y., Carbonell J., Salakhutdinov R. and Le Q. V. (2020). *XLNet: Generalized Autoregressive Pretraining for Language Understanding*. arXiv:1906.08237.