Deep Learning in Event Detection in Polish

Łukasz Kobyliński, Michał Wasiluk
Institute of Computer Science,
Polish Academy of Sciences
Jana Kazimierza 5, 01-248 Warszawa, Poland
lkobylinski@ipipan.waw.pl, m.wasiluk89@gmail.com

Abstract

Event detection is an important NLP task that has been only recently tackled in the context of Polish, mostly due to lack of language resources. The available annotated corpora are still relatively small and supervised learning approaches are limited by the size of training datasets. Event detection tools are very much needed, as they can be used to annotate more language resources automatically and to improve the accuracy of other NLP tasks, which rely on the detection of events, such as question answering or machine translation. In this paper we present a deep learning based approach to this task, which proved to capture the knowledge contained in the training data most effectively and outperform previously proposed methods. We show a direct comparison to previously published results, using the same data and experimental setup.

1 Introduction

The task of identifying events in natural language has a direct impact on the effectiveness of many other tasks in the area of natural language processing. An obvious example is the task of question answering, where the knowledge base has the form of a collection of texts in natural language (Saurí et al., 2005). The answer to the question *When was the current president elected?* requires recognition of the current system time, determining who the current president (of Poland, by implicit assumption) is and identifying the event of election. Other NLP tasks directly influenced by the results of event detection include summarization (Filatova and Hatzivassiloglou, 2004), (Vanderwende et al., 2004), (Li et al., 2006) and machine translation (Horie et al., 2012). In the first case, the events identified in the text allow organizing the content of the summarized document by topics and ordering them chronologically. In the case of machine translation, event detection may be used to create the intermediate knowledge representation layer that is independent of any natural language, which is then used to form the final translation.

In the case of the Polish language, there are only a few published papers on the identification of temporal expressions in natural language text. This is largely due to the current lack of resources, enabling this type of study. For example the authors of (Jarzębowski and Przepiórkowski, 2012) use parallel corpora and annotation projection to Polish to gather the necessary evaluation material. They use the National Corpus of Polish (Przepiórkowski et al., 2012), which contains the basic annotation of simple temporal expressions. Specifically, the manually annotated subcorpus of the NCP includes such tags as: *date* (calendar dates, such as *24 October, 1945*) and *time* (hours, minutes and seconds, e.g. *five after twelve*).

The recently published subcorpus of the KPWr corpus (Kocoń and Marcińczuk, 2015) has been specifically annotated with temporal expressions and events, using an adaptation of the TimeML specification (Saurí et al., 2006). This collection of annotated texts along with additional dictionaries has been used in (Kocoń and Marcińczuk, 2016) to train a CRF-based classifier for the task of identifying events.

2 Event Detection Task

We define the task of detecting events in text as a problem of identifying tokens or token sequences, which should be annotated as an event mention according to the TimeML specification, adapted to Polish by (Marcińczuk et al., 2015). As in the original TimeML specification, we understand events as situations that happen or occur, an

"event is anything that takes place in time (date, time and/or duration) and space (has a location), may involve agents (executor or participants), may contain or be part of other events and may produce some outcome (object)." (Marcińczuk et al., 2015). We aim to classify identified events into one of the following categories, defined by the specification:

- **action** (a dynamic situation which occurs in time and space),
  e.g. *run, fly, hit*,
- **state** (a static situation, which does not change over a period of time),
  e.g. *stand, sit, remain*,
- **reporting** (a dynamic situation where an agent informs about an event or narrates an event),
  e.g. *explain, tell, inform*,
- **perception** (a physical perception of an event by an agent),
  e.g. *see, hear, observe*,
- **aspectual** (indicates a change of a phase of another event),
  e.g. *begin, start, interrupt*,
- **i_action** (intensional action, a situation, where an agent declares his or her will to perform an action or give a command),
  e.g. *try, promise, delay*,
- **i_state** (intensional state, a possible action or state; an agent refers to some possible event, which may or may not occur in the future),
  e.g. *believe, fear, wish*.

The goal of the task is thus to create an annotation layer, which associates event category labels with corresponding tokens. Below is an example annotation, taken from the training corpus (other annotation layers not shown here for readability):

(1.) Po tym **zwycięstwie**$_{\text{action}}$ MKS został liderem grupy 2.

(1.) *After this* ***victory***$_{action}$ *MKS became the leader of group 2.*

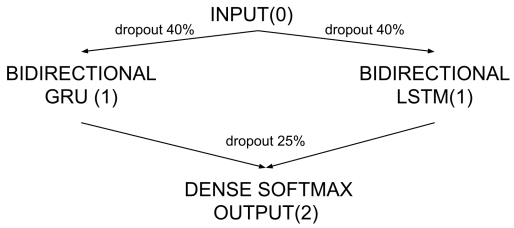

Figure 1: Branched bi gru-lstm architecture.

3 Deep Learning Approach to Event Detection

Preprocessing In the first stage of the proposed method we preprocess the available data and generate feature vectors for the neural network. We scan through the text using a fixed-length processing window: for each token in a sentence a sequence composed of this token (in the center of the window) and its W nearest neighbors within the sentence is generated. Thus, the sequence has a length of $2W + 1$, where W is called the *window size*. The neural network takes as an input a sequence of feature vectors of individual tokens and classifies the central token into one of previously described categories, with an additional *not event* class for not relevant tokens.

Features We use two kinds of embeddings for the real-valued feature vector generation:

1. Simple indexed embeddings, which turn positive integers (indexes) into dense vectors of fixed size by means of simple matrix multiplication:
   — **struct** — structure of a token (vector size: 5) - a token string with all digits replaced by 'd', lowercased characters replaced by 'x', uppercased characters replaced by 'X' and any other character replaced by '-' ("Warszawa-2017" $\rightarrow$ "Xxxxxxx-dddd"). A packed structure is a structure with all neighbouring duplicate code characters removed ("Xxxxxxx-dddd" $\rightarrow$ "Xx-d"),
   — **position** — position of a token in a sequence (3).
2. Pretrained Word2vec (Mikolov et al., 2013) embedding models:
   — **orth** — trained on orthographical word forms from National Corpus of Polish and the Polish Wikipedia (vector size: 300),

Annotation	action	aspectual	i_action	i_state	perception	reporting	state
Number	12861	316	717	1205	149	341	1318

Table 1: Annotations in KPWr-540 by category.

architecture	accuracy	F1						
		action	aspectual	i_action	i_state	perception	reporting	state
br bi gru-lstm	96.291	86.06	74.46	55.73	80.39	90.82	77.60	74.92
br bi lstm-lstm	96.282	86.18	74.22	57.89	77.62	88.28	77.21	74.58
br bi gru-gru	96.229	85.83	72.03	56.77	78.92	89.97	78.55	73.57
br gru-lstm	96.181	85.75	72.68	54.76	77.11	88.20	77.30	73.65
br gru-gru	96.174	85.75	73.80	55.55	76.89	87.33	76.82	71.91
br lstm-lstm	96.162	85.66	73.24	54.87	76.94	85.93	75.33	73.16
bi gru	96.117	85.44	72.97	53.32	79.39	88.42	75.28	72.15
bi lstm	96.098	85.47	71.34	52.55	76.44	87.78	76.19	73.72
lstm	95.937	84.87	71.88	47.57	75.83	74.07	71.91	72.14
gru	95.834	84.47	71.20	47.82	75.20	73.17	71.28	69.38

Table 2: A comparison of network architectures, ordered by overall accuracy (80—20 data split, average from 5 tests, KPWr-540, W = 1, dropout = 0.4, {'hypernym-1', 'lemma', 'orth', 'class'} embeddings).

— **base** — trained on lemmatized word forms (300),
— **class** — trained on POS classes of words from National Corpus of Polish and the Corpus of Polish language of the 1960s (PL1960)[1] (30),
— **ctag** — trained on POS tags of words (300),
— **hypernym-1** — trained on hypernyms of words taken from plWordNet[2] (100),
— **synonym** — trained on synonyms of words taken from plWordNet (100).

In the case of word sense ambiguity during generation of plWordNet-based features (several matching synonyms or hypernyms), the first base form common to all synonyms from all matching synsets is chosen (in alphabetically sorted order). If there is no common base form, or there is no match, the base form of the original token is selected.

Word2vec embedding models were trained in two main steps:

1. Replacement of all tokens in the corpus with corresponding values of the given feature.

2. Training of the w2v model on this newly created corpus using the gensim library[3].

Word2vec feature vectors are assigned to individual tokens by computing given feature value (lemma, hypernym etc.), which then is directly mapped to corresponding feature vector. Eg. *ludzie -> człowiek -> [feature vector]*.

The input vector of an individual token is a concatenation of all component feature vectors. The size of the input vector of the individual token in a sequence with all described features included was 1138 elements.

Network architecture Based on preliminary experiments (described in the Experimental Results section), we have chosen a network consisting of two distinct subnetworks as the most promising for further experiments. The network is split into two branches, Bi-LSTM and Bi-GRU subnetworks. Each of these subnetworks takes the same input, but with a different random dropout applied to it. Bi-LSTM and Bi-GRU can simultaneously model word representation with its preceding and following information. They are composed of two LSTM/GRU neural networks with a hyperbolic (tanh) and hard sigmoid activation functions. The forward LSTM/GRU allows to model the preceding contexts, and a backward LSTM/GRU to model the following contexts respectively.

[1] http://clip.ipipan.waw.pl/PL196x

[2] http://plwordnet.pwr.wroc.pl/wordnet/, (Maziarz et al., 2016)

[3] https://radimrehurek.com/gensim/

dropout	accuracy	F1						
		action	aspectual	i_action	i_state	perception	reporting	state
0.4	96.29	86.06	74.46	55.73	80.39	90.82	77.60	74.92
0.5	96.28	86.14	74.00	56.84	79.42	89.45	76.89	73.84
0.3	96.27	86.00	73.40	56.64	80.11	89.43	78.07	73.91
0.6	96.24	86.06	73.19	56.62	78.72	89.51	77.94	73.96
0.2	96.17	85.68	71.91	54.53	78.57	87.14	77.21	72.97
0.1	96.12	85.36	70.97	53.19	78.98	86.32	76.19	73.61
0.7	96.08	85.62	72.37	54.15	77.45	88.63	76.75	73.12
0.0	96.03	85.11	71.35	50.45	78.59	81.79	74.25	71.57
0.8	95.72	84.26	71.30	49.85	77.22	84.80	73.27	70.56
0.9	95.14	82.38	71.85	35.05	75.69	28.42	67.38	61.40

Table 3: The influence of the input dropout parameter on network accuracy (80-20 data split, average from 9 tests, branched bi gru-lstm architecture, KPWr-540, W = 1, {'hypernym-1', 'lemma', 'orth', 'class'} embeddings).

We flatten and concatenate the bidirectional sequence features learned from the subnetworks and apply random dropout to the result. Then, we use a dense softmax approach to perform final classification. The architecture of the network has been presented on Figure 1.

We train our model with categorical cross-entropy loss function and Adam optimizer (Kingma and Ba, 2014) with small learning rate decay. For GRU and LSTM we use glorot (Glorot and Bengio, 2010) and orthogonal (Saxe et al., 2013) initializers.

4 Experimental Results

Data The KPWr-540 corpus (Kocoń and Marcińczuk, 2015) has previously been used to train machine learning methods for the task of event detection. Here we use the same dataset to allow direct comparisons with previously published approaches. The dataset contains 540 documents, 6 915 sentences (948 sentences without any event utterance) and 121 747 tokens. In total, there are 17 078 human-made annotations in the corpus. The breakdown of the annotation types has been presented in Table 1. The annotations consist predominantly of a single token, only 4 annotations have a token span length of 2.

Preliminary experiments To determine the appropriate network architecture for the stated problem we have conducted a series of preliminary experiments on the available dataset, using a 80—20 split between train and test data. The most representative differences between network architectures, as measured during these experiments, have been presented in Table 2. The accuracy column represents overall classification accuracy of the network (*no event* class included).

In further experiments we have measured the influence of the dropout parameter on classification accuracy (the results are presented in Table 3) and we have found the optimal set of features (the results are presented on Figure 2).

Evaluation The final evaluation of the proposed method accuracy has been performed using a 10-fold cross-validation on the available data. In these experiments each fold's training set was additionally split into 2 parts: train (80%) — used for neural network training and validation (20%) — used for early stopping and best model selection. We have also evaluated two approaches to the train set splitting: performing a simple split and a balanced split with preserved ratio of event category samples. The weights were balanced for each class.

The results of the final comparison of the proposed method to the CRF-based approach presented in (Kocoń and Marcińczuk, 2016) has been shown in Table 4. The presented deep-learning approach proved to perform better for each event category, as measured by the F1 score.

5 Conclusions and Future Work

In this paper we have applied a deep-learning approach to the problem of detecting events in text. As in many other NLP tasks, modern neural networks proved to perform very well in this domain and outperformed the previously proposed method, which was based on Conditional Random Fields.

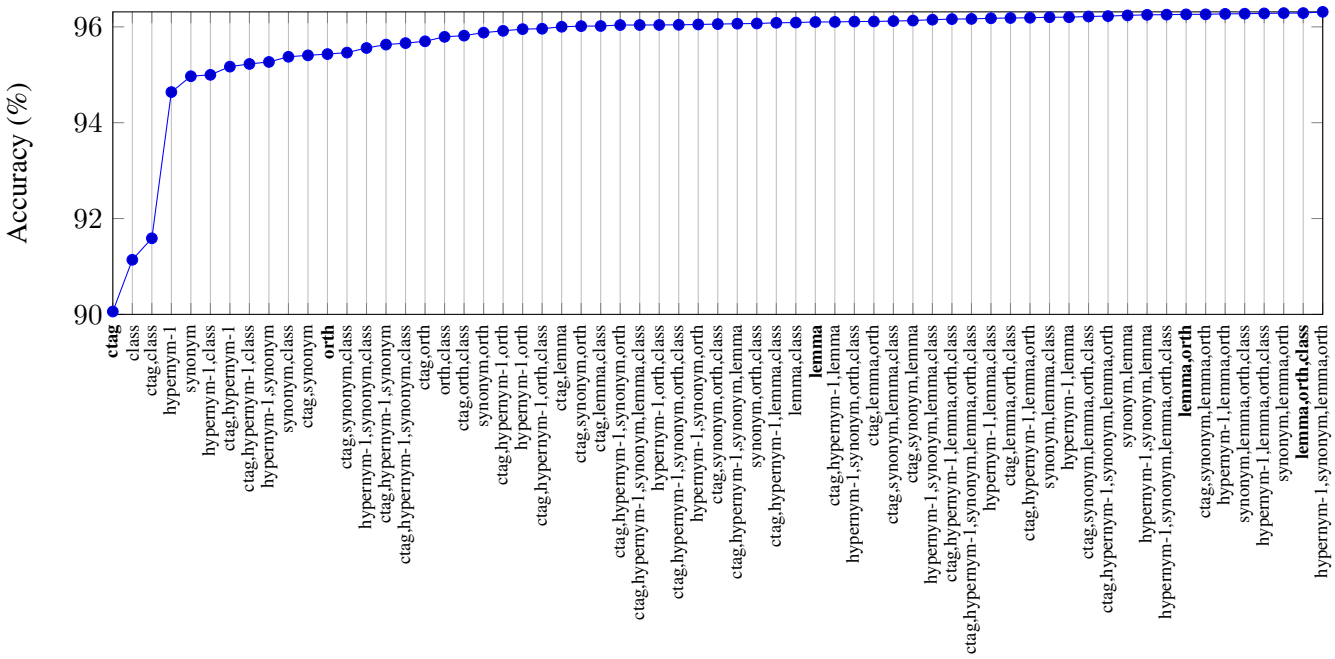

Figure 2: The impact of word embeddings configuration on overall classification accuracy (80-20 data split, average from 5 tests, KPWr-540, W = 1, dropout = 0.4).

Category	branched bi gru-lstm W=1			Liner2 w/o dictionaries			Liner2 with dictionaries (*)		
	P	R	F1	P	R	F1	P	R	F1
action	**84.90**	**88.33**	**86.57**	82.51	84.90	83.69	82.49	83.87	83.18
aspectual	85.87	**72.96**	**78.67**	87.56	60.13	71.29	**87.58**	59.24	70.68
i_action	66.89	**58.58**	**62.12**	**67.48**	42.54	52.18	63.56	40.92	49.79
i_state	84.35	**82.60**	**83.38**	84.35	78.26	81.19	**85.19**	77.56	81.20
perception	85.17	**75.61**	**79.33**	**97.53**	53.02	68.70	85.90	55.37	67.34
reporting	69.29	**66.65**	**67.11**	**75.00**	57.18	64.89	71.13	51.30	59.61
state	**73.03**	**69.09**	**70.86**	71.84	61.15	66.07	68.10	62.17	65.00

Table 4: Comparison of the best performing network architecture against the previously proposed CRF-based approach. Ten-fold cross-validation on the KPWr-540 corpus. (*) Results taken from (Kocoń and Marcińczuk, 2016).

It is interesting to note that features based on words (**orth**, **lemma** in Figure 2) influenced the resulting accuracy the most, proving that such embeddings carry essential information for this task. On the other hand, features based on part-of-speech tags (**ctag**, **class**) were among the least informative. A characteristic feature of processing inflected languages is the importance of lemmatization and including lemmas in the feature set. The large number of inflected word forms in languages such as Polish (and other Slavic languages), makes it more difficult for word-form embeddings to capture information that is properly generalized. Generating embeddings from lemmas helps to solve the problem, as long as the lemmatization is accurate and does not introduce additional disambiguation difficulties.

In future work we would like to tackle a more general task of event recognition in text, including the identification of textual arguments of events. This may include such entities as place names (where the event takes place), time and date specifications (when it takes place), or person names (agents or beneficiaries of an event). We would also like to analyze the relationships occurring between several events recognized in a text fragment (e.g. event identity).

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

Elena Filatova and Vasileios Hatzivassiloglou. 2004. Event-based extractive summarization. In *Proceedings of the ACL Workshop on Summarization*, pages 104–111.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May. PMLR.

André Horie, Kumiko Tanaka-Ishii, and Mitsuru Ishizuka. 2012. Verb temporality analysis using Reichenbach's tense system. In *Proceedings of COLING 2012: Posters*, pages 471–482, Mumbai, India, December. The COLING 2012 Organizing Committee.

Przemysław Jarzębowski and Adam Przepiórkowski. 2012. Temporal information extraction with cross-language projected data. In Hitoshi Isahara and Kyoko Kanzaki, editors, *Advances in Natural Language Processing: Proceedings of the 8th International Conference on NLP, JapTAL 2012, Kanazawa, Japan, October 22-24, 2012*, volume 7614 of *Lecture Notes in Artificial Intelligence*, pages 198–209. Springer-Verlag, Heidelberg.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Jan Kocoń and Michał Marcińczuk. 2015. KPWr events. CLARIN-PL digital repository.

Jan Kocoń and Michał Marcińczuk, 2016. *Generating of Events Dictionaries from Polish WordNet for the Recognition of Events in Polish Documents*, pages 12–19. Springer International Publishing, Cham.

Wenjie Li, Mingli Wu, Qin Lu, Wei Xu, and Chunfa Yuan. 2006. Extractive summarization using inter- and intra- event relevance. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, ACL-44, pages 369–376, Stroudsburg, PA, USA. Association for Computational Linguistics.

Michał Marcińczuk, Marcin Oleksy, Tomasz Bernaś, Jan Kocoń, and Michał Wolski. 2015. Towards an event annotated corpus of Polish. *Cognitive Studies*, 15:253–267, 12.

Marek Maziarz, Maciej Piasecki, Ewa Rudnicka, Stan Szpakowicz, and Paweł Kędzia. 2016. PlWordNet 3.0 – a Comprehensive Lexical-Semantic Resource. In Nicoletta Calzolari, Yuji Matsumoto, and Rashmi Prasad, editors, *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, pages 2259–2268. ACL, ACL.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Adam Przepiórkowski, Mirosław Bańko, Rafał L. Górski, and Barbara Lewandowska-Tomaszczyk, editors. 2012. *Narodowy Korpus Języka Polskiego*. Wydawnictwo Naukowe PWN, Warsaw.

Roser Saurí, Robert Knippen, Marc Verhagen, and James Pustejovsky. 2005. Evita: A robust event recognizer for qa systems. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 700–707, Stroudsburg, PA, USA. Association for Computational Linguistics.

Roser Saurí, Jessica Littman, Bob Knippen, Andrea Gaizauskas, Robert abd Setzer, and James Pustejovsky. 2006. Timeml annotation guidelines.

Andrew M. Saxe, James L. McClelland, and Surya Ganguli. 2013. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *CoRR*, abs/1312.6120.

Lucy Vanderwende, Michele Banko, and Arul Menezes. 2004. Event-centric summary generation. In *Working Notes of the 2004 Document Understanding Conference (DUC'04)*.