Proceedings of the PolEval 2019 Workshop

Maciej Ogrodniczuk, Łukasz Kobyliński (eds.)

Institute of Computer Science, Polish Academy of Sciences
Warszawa, 2019

ISBN 978-83-63159-28-3

Publikacja finansowana przez Instytut Podstaw Informatyki PAN.

Instytut Podstaw Informatyki PAN
01-248 Warszawa, ul. Jana Kazimierza 5

Wydanie 1, Warszawa 2019

Contents

PolEval 2019: Introduction

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

This volume consists of proceedings of PolEval session, organized during the AI & NLP Day 2019 (`https://nlpday.pl`), which took place on May 31st, 2019 at the Institute of Computer Science, Polish Academy of Sciences in Warsaw. PolEval (`http://poleval.pl`) is an evaluation campaign organized since 2017, which focuses on Natural Language Processing tasks for Polish, promoting research on language and speech technologies.

As a consequence of the global growth of interest in Natural Language Processing and rising number of published papers, frameworks for an objective evaluation and comparison of NLP-related methods become crucial to support effective knowledge dissemination in this field. Following the initiatives of ACE Evaluation[1], SemEval[2], or Evalita[3], we have observed the need to provide a similar platform for comparing NLP methods, which would focus on Polish language tools and resources.

Since 2017 we observe a steady growth of interest in PolEval participation: the first edition has attracted 20 submissions, while in 2018 we have received 24 systems for evaluation. During the 2019 edition of PolEval six different tasks have been announced and teams from both academia and business submitted 34 systems in total (see Figure 1).

In 2019 the systems competed in the following tasks:

— Task 1: Recognition and normalization of temporal expressions

— Task 2: Lemmatization of proper names and multi-word phrases

— Task 3: Entity linking

— Task 4: Machine translation (EN-PL, PL-RU, RU-PL)

— Task 5: Automatic speech recognition

— Task 6: Automatic cyberbullying detection
(harmful vs non-harmful and detecting type of harmfulness).

[1] `https://en.wikipedia.org/wiki/Automatic_content_extraction`
[2] `https://en.wikipedia.org/wiki/SemEval`
[3] `http://www.evalita.it/`

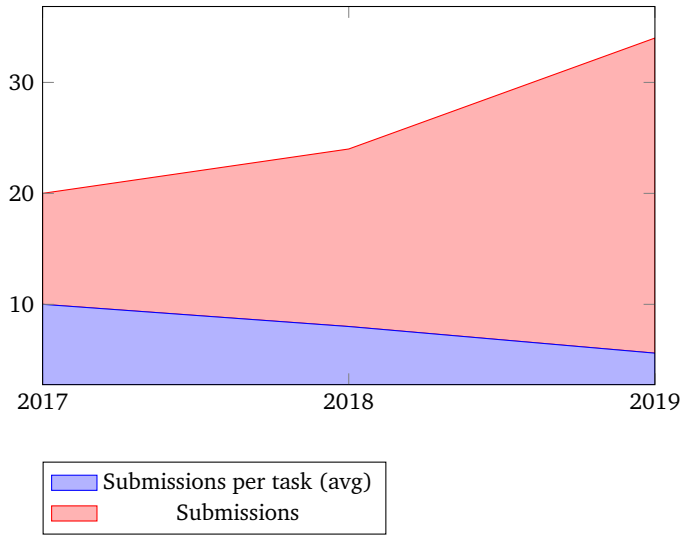

Figure 1: Number of PolEval submissions and average submissions per task in 2017–2019

The number of submissions per each task has varied greatly (see Figure 2): the subject of hate speech and cyberbullying has attracted the most submissions during this edition of the campaign.

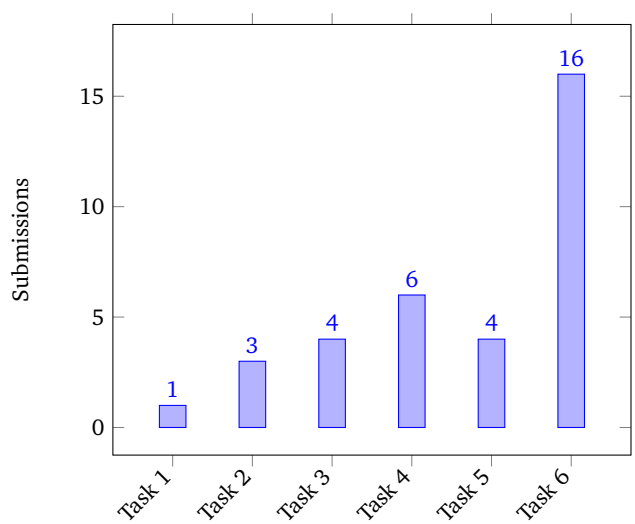

Figure 2: Number of PolEval submissions per task in 2019

We hope you will join us next year for PolEval 2020! Please feel free to share your ideas for improving this competition or willingness to help in organizing your own NLP tasks.

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: Recognition and normalization of temporal expressions

Jan Kocoń (Wrocław University of Science and Technology)

Task 2: Lemmatization of proper names and multi-word phrases

Michał Marcińczuk (Wrocław University of Science and Technology)

Task 3: Entity linking

Aleksander Smywiński-Pohl (AGH University of Science and Technology)

Task 4: Machine translation

Krzysztof Wołk (Polish-Japanese Academy of Information Technology)

Task 5: Automatic speech recognition

Danijel Koržinek (Polish-Japanese Academy of Information Technology)

Task 6: Automatic cyberbullying detection

Michał Ptaszyński (Kitami Institute of Technology, Japan)
Agata Pieciukiewicz (Polish-Japanese Academy of Information Technology)
Paweł Dybała (Jagiellonian University in Kraków)

Results of the PolEval 2019 Shared Task 1: Recognition and Normalization of Temporal Expressions

Jan Kocoń, Marcin Oleksy, Tomasz Bernaś, Michał Marcińczuk
(Department of Computational Intelligence, Wrocław University of Science and Technology)

Abstract

This article presents the research in the recognition and normalization of Polish temporal expressions as the result of the first PolEval 2019 shared task. Temporal information extracted from the text plays a significant role in many information extraction systems, like question answering, event recognition or text summarization. A specification for annotating Polish temporal expressions (PLIMEX) was used to prepare a completely new test dataset for the competition. PLIMEX is based on state-of-the-art solutions for English, mostly TimeML. The training data provided for the task is Polish Corpus of Wrocław University of Science and Technology (KPWr) fully annotated using PLIMEX guidelines.

Keywords

natural language processing, information extraction, temporal expressions, recognition, normalization, Polish

1. Introduction

Temporal expressions (henceforth *timexes*) tell us *when* something happens, *how long* something lasts, or *how often* something occurs. The correct interpretation of a timex often involves knowing the context. Usually, people are aware of their location in time, i.e., they know what day, month and year it is, and whether it is the beginning or the end of week or month. Therefore, they refer to specific dates, using incomplete expressions such as *12 November*, *Thursday*, *the following week*, *after three days*. The temporal context is often necessary to determine to which specific date and time *timexes* refer. These examples do not exhaust the complexity of the problem of recognizing timexes.

The possibility of sharing information about recognized timexes between different information systems is very important. For example, a Polish programmer may create a method that will recognize the expression *the ninth of December* and normalize it (knowing the context of the whole document) to the form `09.12.2015`. A programmer from the United States could create a similar method that normalizes the same expression to a form `12/9/2015`. A serious problem is the need to use the information from two such methods, e.g. for the analysis of multilingual text sources, where the expected effect is a certain metadata unification and the application of international standards. Normalization allows to determine the machine-readable form of timexes and requires the analysis of each expression in a broad context (even the whole document), due to the need to make some calculations on relative timexes, such as *five minutes earlier, three hours later*.

TimeML (Saurí et al. 2006) is a markup language for describing timexes that has been adapted to many languages. The specification was created as part of the TERQAS[1] workshop, as part of the AQUAINT project[2], aimed at improving the quality of methods for question answering(Pustejovsky et al. 2005). The aim of this study was to improve access to information in the text with the focus on in-depth analysis of the content, not only through keywords. A key problem was the recognition of events and their location in time.

2. Task Description

The aim of this task is to advance research on the processing of timexes, which are used in other NLP applications like question answering, textual entailment, document classification, summarization, etc. This task follows on from previous TempEval events organized for evaluating time expressions for English and Spanish like SemEval-2013 (UzZaman et al. 2013). This time a corpus of Polish documents fully annotated with temporal expressions was provided. The annotation consists of boundaries, classes and normalized values of temporal expressions. The annotation for Polish texts is based on a modified version of original TIMEX3 annotation guidelines[3] at the level of annotating boundaries/types[4] and local/global normalization[5] (Kocoń et al. 2015).

3. Data

The training dataset contains 1500 documents from KPWr corpus. Each document is an XML file with the given annotations, e.g.:

[1] Time and Event Recognition for Question Answering Systems. An Advanced Research and Development Activity Workshop on Advanced Question Answering Technology

[2] Advanced Question and Answering for Intelligence, `http://www.informedia.cs.cmu.edu/aquaint/index.html`

[3] `https://catalog.ldc.upenn.edu/docs/LDC2006T08/timeml_annguide_1.2.1.pdf`

[4] `http://poleval.pl/task1/plimex_annotation.pdf`

[5] `http://poleval.pl/task1/plimex_normalisation.pdf`

```
<DOCID>344245.xml</DOCID>
<DCT>
  <TIMEX3 tid="t0" functionInDocument="CREATION_TIME"
          type="DATE" value="2006-12-16">
  </TIMEX3>
</DCT>
<TEXT>
  <TIMEX3 tid="t1" type="DATE" value="2006-12-16">Dziś</TIMEX3>
  Creative Commons obchodzi czwarte urodziny - przedsięwzięcie
  ruszyło dokładnie <TIMEX3 tid="t2" type="DATE" value="2002-12-16">
  16 grudnia 2002</TIMEX3> w San Francisco.
  (...)
  Z kolei w <TIMEX3 tid="t4" type="DATE" value="2006-12-18">
  poniedziałek</TIMEX3> ogłoszone zostaną wyniki głosowania
  na najlepsze blogi. W ciągu <TIMEX3 tid="t5" type="DURATION"
  value="P8D">8 dni</TIMEX3> internauci oddali ponad pół miliona
  głosów. Z najnowszego raportu Gartnera wynika, że w <TIMEX3
  tid="t6" type="DATE" value="2007">przyszłym roku</TIMEX3>
  blogosfera rozrośnie się do rekordowego rozmiaru 100 milionów
  blogów.
  (...)
</TEXT>
```

4. Evaluation

The same evaluation procedure was utilized as described in article (UzZaman et al. 2013). It answers the three given questions:

1. how many entities are correctly identified
2. if the extents for the entities are correctly identified
3. how many entity attributes are correctly identified.

Evaluation metrics used are classical precision (P), recall (R) and F1-score (F1 – a harmonic mean of P and R) for the recognition. Each of the above steps is solved as follows:

1. Annotated chunks (entities) recognized as timexes are evaluated with the given equations:

$$P = \frac{\left|\mathrm{Sys}_{\mathrm{entity}} \cap \mathrm{Ref}_{\mathrm{entity}}\right|}{\left|\mathrm{Sys}_{\mathrm{entity}}\right|} \qquad R = \frac{\left|\mathrm{Sys}_{\mathrm{entity}} \cap \mathrm{Ref}_{\mathrm{entity}}\right|}{\left|\mathrm{Ref}_{\mathrm{entity}}\right|}$$

   where, $\mathrm{Sys}_{\mathrm{entity}}$ contains the entities extracted by the system that we want to evaluate, and $\mathrm{Ref}_{\mathrm{entity}}$ contains the entities from the reference annotation that are being compared.

2. Entities are compared with both strict match and relaxed match. When there is an exact match between the system entity and gold entity then it is called a *strict match*, e.g. *16 grudnia 2002* vs *16 grudnia 2002*. When there is an overlap between the system entity and gold entity then it is called a *relaxed match*, e.g. *16 grudnia 2002* vs *2002*. When there is a relaxed match, the attribute values are compared.
3. Entity attributes are evaluated using the *attribute F1-score*, which captures how well the system identified both the entity and attribute together:

$$\text{attrP} = \frac{\left|\forall x | x \in (\text{Sys}_{\text{entity}} \cap \text{Ref}_{\text{entity}}) \wedge \text{Sys}_{\text{attr}}(x) = \text{Ref}_{\text{attr}}(x)\right|}{\left|\text{Sys}_{\text{entity}}\right|}$$

$$\text{attrR} = \frac{\left|\forall x | x \in (\text{Sys}_{\text{entity}} \cap \text{Ref}_{\text{entity}}) \wedge \text{Sys}_{\text{attr}}(x) = \text{Ref}_{\text{attr}}(x)\right|}{\left|\text{Ref}_{\text{entity}}\right|}$$

P, R, F1 are calculated for both strict and relaxed match and relaxed F1 for value and type attributes. The most important metric is the *relaxed F1 value*.

5. Participating Systems and Results

The best result in the main competition (excluding a baseline system provided by organizers) was achieved by Alium team with its Alium solution. Alium solution is an engine to process texts in natural language and produce results according to rules that define its behaviour. Alium can work either on single words or on triples – *word, lemma, morphosyntactic tag*. Words are additionally masked (digits, special signs, etc.) so that Alium can work on parts of words as well.

Rules that are accepted by Alium can be built with a bottom-up approach. Lower level rules can produce results, that are further consumed by higher-level rules. A rule can filter a word (or set of words) that fulfill complex conditions (based on its orthographic form, lemma, morphosyntactic tags or attributes defined in the lower-level rule). Rules can detect words that share the same case, number, type, aspect, etc. A rule can produce versatility of results with the support of different functions (e.g. lemmas form words, number in digits from numerical texts, part of results from lower level rule results, etc.). Alium engine is ready to handle even hundreds of thousands of rules with efficient pruning algorithms.

With all that functionality in mind, temporal expressions detection and normalization task required no more than 420 rules to detect date, time, duration or vague expressions. All those rules created a hierarchy of rules with 6 levels. Rules are based on words only, not on triples. With rules based on triples, the number of rules would be lower (other experiments show that it could require 30% less rules). Final texts were generated by scripts in the post-processing phase. The main goal of those scripts was to produce texts that conform to TIMEX3 format. We compared Alium system with Liner2 tool (Marcińczuk et al. 2017) used as a baseline. Results are presented in Table 1.

Table 1: Results of the recognition and normalization of temporal expressions obtained by Alium system, compared to results obtained by Liner2 system (Kocoń and Marcińczuk 2017, Marcińczuk et al. 2017).

Strict Match	**F1**	**P**	**R**
Alium	58.81	58.91	58.72
Liner2 (baseline)	87.63	86.17	89.14
Relaxed Match	**F1**	**P**	**R**
Alium	86.49	86.63	86.35
Liner2 (baseline)	91.19	89.67	92.76
Attribute F1	**Value**	**Type**	
Alium	68.70	80.23	
Liner2 (baseline)	76.96	87.79	

6. Conclusions

The proposed Alium system uses a rule-based approach to perform the recognition and normalization phase. We did not take part in the competition but we compared the results of Alium with our Liner2 system (see Table 1), which performs the recognition of named entities (Marcińczuk et al. 2013, Marcińczuk et al. 2017), events (Kocoń and Marcińczuk 2016) and temporal expressions (Kocoń and Marcińczuk 2015, 2017, Kocoń and Marcińczuk 2017). Liner2 is an open-source system available with the configuration used for this task in CLARIN-PL DSpace repository: `http://hdl.handle.net/11321/531` (available soon in main Liner2 GitHub repository: `https://github.com/CLARIN-PL/Liner2`). In all test cases, Liner2 outperformed Alium. The most significant differences can be observed with *Strict Match* results. The most likely reason is that our system uses Conditional Random Fields to recognize timexes, similarly as the best system presented in (UzZaman et al. 2013). The most promising improvement is the use of deep recurrent neural network, as it was presented in work (Kocoń and Gawor 2019).

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

Kocoń J. and Gawor M. (2019). *Evaluating KGR10 Polish Word Embeddings in the Recognition of Temporal Expressions Using BiLSTM-CRF*. „CoRR", abs/1904.04055.

Kocoń J. and Marcińczuk M. (2015). *Recognition of Polish Temporal Expressions*. „Proceedings of the Recent Advances in Natural Language Processing", pp. 282–290. Recent Advances in Natural Language Processing (RANLP 2015).

Kocoń J. and Marcińczuk M. (2016). *Generating of Events Dictionaries from Polish WordNet for the Recognition of Events in Polish Documents*. In *Text, Speech and Dialogue, Proceedings of the 19th International Conference TSD 2016*, vol. 9924 of *Lecture Notes in Artificial Intelligence*, Brno, Czech Republic. Springer.

Kocoń J. and Marcińczuk M. (2017). *Supervised approach to recognise Polish temporal expressions and rule-based interpretation of timexes*. „Natural Language Engineering", 23(3), p. 385–418.

Kocoń J., Marcińczuk M., Oleksy M., Bernaś T. and Wolski M. (2015). *Temporal Expressions in Polish Corpus KPWr*. „Cognitive Studies — Études Cognitives", 15.

Kocoń J. and Marcińczuk M. (2017). *Improved Recognition and Normalisation of Polish Temporal Expressions*. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pp. 387–393.

Marcińczuk M., Kocoń J. and Janicki M. (2013). *Liner2 – A Customizable Framework for Proper Names Recognition for Polish*. In Bembenik R., Skonieczny Ł., Rybiński H., Kryszkiewicz M. and Niezgódka M. (eds.), *Intelligent Tools for Building a Scientific Information Platform*, vol. 467 of *Studies in Computational Intelligence*, pp. 231–253. Springer Berlin Heidelberg.

Marcińczuk M., Kocoń J. and Oleksy M. (2017). *Liner2 — a Generic Framework for Named Entity Recognition*. In *Proceedings of the 6th Workshop on Balto-Slavic Natural Language Processing*, pp. 86–91, Valencia, Spain. Association for Computational Linguistics.

Pustejovsky J., Ingria B., Sauri R., Castano J., Littman J., Gaizauskas R., Setzer A., Katz G. and Mani I. (2005). *The Specification Language TimeML*. In *The language of time: A reader*, pp. 545–557. Oxford University Press.

Saurí R., Littman J., Gaizauskas R., Setzer A. and Pustejovsky J. (2006). *TimeML Annotation Guidelines, Version 1.2.1*.

UzZaman N., Llorens H., Derczynski L., Allen J., Verhagen M. and Pustejovsky J. (2013). *SemEval-2013 Task 1: TempEval-3: Evaluating Time Expressions, Events, and Temporal Relations*. In *2nd Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the 7th International Workshop on Semantic Evaluation (SemEval 2013)*, vol. 2, pp. 1–9.

Results of the PolEval 2019 Task 2: Lemmatization of Proper Names and Multi-word Phrases

Michał Marcińczuk, Tomasz Bernaś (Wrocław University of Science and Technology)

Abstract

This paper summarises the PolEval 2019 shared task on lemmatization of proper names and multi-word phrases for Polish. The participating system has to generate a lemma for each phrase marked in the input set of documents following the KPWr lemmatization guidelines. Each document contains a plain text with a set of phrases marked with XML tags. The input data does not contain segmentation, tokenization, morphological analysis, nor semantic category of the phrases. Two systems took part in the task: *zbronk.nlp.studio* and *PolEval2019-lemmatization*. The winner of the task was *zbronk.nlp.studio* which obtained the score of 87.46. The second place went to *PolEval2019-lemmatization* with the score of 74.86.

Keywords

natural language processing, lemmatization, multi-word phrases, proper names

1. Introduction

Task 2 of PolEval 2019 focuses on lemmatization of proper names and multi-word phrases for Polish. Lemmatization consists in generating a dictionary form of a phrase. For example, the following noun phrases *radę nadzorczą*, *radzie nadzorczej*, *radą nadzorczą* which are inflected forms of *board of directors* should be lemmatized to *rada nadzorcza*. Polish is a highly inflectional language which causes that lemmatization of text phrases is needed in many natural language processing task, including keyword extraction, sentiment analysis and information aggregation.

The problem of phrase lemmatization has already been addressed by several researchers using different approaches. Piskorski et al. (2007) presented two methods (based on rules and string distance metrics) for person names lemmatization. Degórski (2012) proposed a rule-based

method for lemmatization nominal syntactic groups utilized a shallow grammar. Radziszewski (2013) presented an approach to noun phrase lemmatization based on transformations learned from a training data using a machine learning technique. Małyszko et al. (2015) automatically retrieved a list of lemmatization rules for multi-word units based on a corpus analysis. Marcińczuk (2017) used a set of language resources, manually crafted rules and heuristics to lemmatize multi-word expressions and proper names. Each of the presented solutions was evaluated on a different dataset, therefore it impossible to compare the solutions. However, none of them gave the ultimate solution for the problem.

The difficulty of multi-word phrase lemmatization is due to the fact that the expected lemma is not a simple concatenation of base forms for each word in the phrase (Marcińczuk 2017). In most cases only the head of the phrase is changed to a nominative form and the remaining words, which are the modifiers of the head, should remain in a specific case. For example in the phrase *piwnicy domu* (Eng. *house basement*) only the first word should be changed to their nominative form while the second word should remain in the genitive form, i.e. *piwnica domu*. A simple concatenation of tokens' base forms would produce a phrase *piwnica dom* which is not correct.

In the case of proper names the following aspects make the lemmatization task difficult:

1. Proper names may contain words which are not present in the morphological dictionaries. Thus, dictionary-based methods are insufficient.
2. Some foreign proper names are subject to inflection and some are not.
3. The same text form of a proper name might have different lemmas depending on their semantic category. For example *Słowackiego* (a person last name in genitive or accusative) should be lemmatized to *Słowacki* in case of person name and to *Słowackiego* in case of street name.
4. Capitalization does matter. For example a country name *Polska* (Eng. *Poland*) should be lemmatized to *Polska* but not to *polska*.

2. Task Description

The participating system has to generate a lemma for each phrase marked in the input set of documents. The generated lemmas should follow the KPWr guidelines for phrase lemmatization (Oleksy et al. 2018). The lemma should take into consideration the context in which the phrase occurred. For instance, the phrase *Sienkiewicza* as a person name should be lemmatized to *Sienkiewicz* but as a street name should remain as *Sienkiewicza* (see Table 1).

3. Data

Each dataset consists of the set of XML files representing the documents. Each document contains a plain text with a set of phrases marked with XML tags. The input data does not contain segmentation, tokenization, morphological analysis, nor semantic category of the phrases (see Figure 1).

Table 1: Sample phrases with their lemmas based on the phrase semantic category

NE category	Surname	Street
form	Sienkiewicza	Sienkiewicza
word lemma	Sienkiewicz	Sienkiewicz
lemma	Sienkiewicz	Sienkiewicz**a**

```
<?xml version="1.0" encoding="UTF-8"?>
<document id="00107258">
  2006-01-28:
  <phrase id="318041">
    <phrase id="318042">Jarosław</phrase>
    <phrase id="318043">Kaczyński</phrase>
  </phrase>: Koalicja rządowa z <phrase id="318044">Samoobroną</phrase>
  i <phrase id="318045">LPR</phrase> na razie niemożliwa

  <phrase id="318046">
    <phrase id="318047">Jarosław</phrase>
    <phrase id="318048">Kaczyński</phrase>
  </phrase> powiedział w <phrase id="318049">Sygnałach Dnia</phrase>,
  że koalicja rządowa z <phrase id="318050">Samoobroną</phrase> i
  <phrase id="318051">Ligą
    <phrase id="319896">Polskich</phrase> Rodzin
  </phrase> jest w tej chwili niemożliwa. Kaczyński zaznaczył
  jednocześnie, że nie należy rozpisywać nowych wyborów. Prezes <phrase
  id="486119">Prawa i Sprawiedliwości</phrase> dodał, że liczy się
  z głosem społeczeństwa, które - jego zdaniem - sprzeciwia się wyborom.
  Z kolei Kazimierz Marcinkiewicz uważa, że jeżeli nie dojdzie
  do podpisania <phrase id="318058">paktu stabilizacyjnego</phrase>
  w ciągu dwóch tygodni, to powinno dojść do rozwiązania parlamentu.
  Rozmowy w parlamencie pomiędzy ugrupowaniami politycznymi trwają. Albo
  w ciągu najbliższego tygodnia, maksymalnie dwóch, nastąpi podpisanie
  <phrase id="318059">paktu stabilizacyjnego</phrase>, czyli znajdzie
  się większość, która będzie wspierała rząd w ciągu przynajmniej
  najbliższych sześciu miesięcy, albo - jeśli nie - trzeba będzie
  odwołać się do demokracji - zaznacza premier. Jarosław Kaczyński
  stwierdził także, że jeszcze wczoraj była szansa na porozumienie
  z <phrase id="486121">Platformą Obywatelską</phrase>. Jego zdaniem
  partia
  <phrase id="318064">Donalda
    <phrase id="318066">Tuska</phrase>
  </phrase> stawiała warunki dominacji w rządzie. Prezes PiS obarczył
  winą PO za zamieszanie w obecnym <phrase id="318069">Sejmie</phrase>
  i brak rządu o stałym poparciu.
</document>
```

Figure 1: A sample document with marked phrases from the training dataset

The XML files were accompanied by a TSV file with a list of phrases (see Figure 2).

```
486119   00107258   Prawa i Sprawiedliwości   Prawo i Sprawiedliwość
318048   00107258   Kaczyński                 Kaczyński
318041   00107258   Jarosław Kaczyński        Jarosław Kaczyński
318064   00107258   Donalda Tuska             Donald Tusk
318049   00107258   Sygnałach Dnia            Sygnały Dnia
318066   00107258   Tuska                     Tusk
318044   00107258   Samoobroną                Samoobrona
318069   00107258   Sejmie                    Sejm
318059   00107258   paktu stabilizacyjnego    pakt stabilizacyjny
318050   00107258   Samoobroną                Samoobrona
318058   00107258   paktu stabilizacyjnego    pakt stabilizacyjny
318051   00107258   Ligą Polskich Rodzin      Liga Polskich Rodzin
486121   00107258   Platformą Obywatelską     Platforma Obywatelska
```

Figure 2: A list of phrases with their lemmas taken from the TSV file from the training dataset

3.1. Training Datasets

The training dataset consists of 1629 documents from the KPWr corpus (Broda et al. 2012) with more than 24k phrases[1]. Participants are allowed to use any other resources as training data.

3.2. Tuning Datasets

The tuning dataset consists of 200 documents from the Corpus of Economic News (CEN) corpus with 1145 phrases[2]. The complete CEN corpus (Marcińczuk 2007) contains 797 documents from Polish Wikinews[3]. The tuning dataset was released as an additional dataset with sample phrases subjected to lemmatization (without phrase gold lemmatization).

3.3. Testing Dataset

The testing dataset consists of 99 documents from the Polish Spatial Texts (PST)[4] corpus with 1997 phrases[5]. PST is a collection of articles from Polish travel blogs. This set was used to evaluate and compare system responses.

[1] http://poleval.pl/task2/poleval2019_task2_training_190221.tar.gz
[2] http://poleval.pl/task2/task2_test.tar.gz
[3] https://pl.wikipedia.org/wiki/Wikinews
[4] https://clarin-pl.eu/dspace/handle/11321/543
[5] http://poleval.pl/task2/poleval2019_task2_test_second_190517.zip

4. Evaluation

The score of the system responses was calculated using the following formula:

$$Score = 0.2 * Acc_{CS} + 0.8 * Acc_{CI}$$

Acc refers to the accuracy, i.e. a ratio of the correctly lemmatized phrases to all phrases subjected to lemmatization.

The accuracy was calculated in two variants: *case sensitive* (Acc_{CS}) and *case insensitive* (Acc_{CI}). In the case insensitive evaluation lemmas were converted to lower cases.

System with the highest *Score* was the winner.

5. Participating Systems and Results

Two systems partook in the lemmatization task:

1. *zbronk.nlp.studio* — the solution is based on a set of lemmatization heuristics utilizing different language tools and resources, including: Morfeusz morphological analyzer (Kieraś and Woliński 2017), a proprietary quasi-dependency parser, NKJP corpus (Przepiórkowski et al. 2012), a large corpus of unnannotated texts (4 billion words) and Multisłownik[6] (Ogrodniczuk et al. 2018).
2. *PolEval2019-lemmatization* — the system works by solving a tagging problem, where the tags represent transformations that need to be performed on each word. It was trained using the data provided by the contest organizers. Tokenization was performed using the UDPipe tokenizer. Additionally, each sentence containing a phrase was tokenized using the COMBO tokenizer. The system architecture is as follows: first, the data and the dependency parser features are fed to two separate embedding layers. The embeddings are concatenated and fed to a bidirectional LSTM layer. Calculated features are then truncated to match the lemmatized phrase and fed to a CRF layer. Operations represented by the predicted tags are performed using the Morfeusz morphological analyzer (Kieraś and Woliński 2017).

For *PolEval2019-lemmatization* we got two submissions: *model3* trained solely on the training datasets and *new1* which utilized both the training and tuning dataset. The winner of the task was *zbronk.nlp.studio* which obtained the score of 87.46. The second place went to *PolEval2019-lemmatization* with the score of 74.86.

6. Conclusions

It is worth noting that the two submitted system follows two opposite approaches to the task: machine learning vs. knowledge-based heuristics. The last edition of PolEval competition

[6] `http://multislownik.nlp.ipipan.waw.pl`

Table 2: Results of lemmatization task

System name	Variant	Acc_{CS}	Acc_{CI}	Score
zbronk.nlp.studio	–	84.78	88.13	**87.46**
PolEval2019-lemmatization	new1	72.46	75.46	74.86
PolEval2019-lemmatization	model3	68.85	71.71	71.14

(Ogrodniczuk and Kobyliński 2018) showed, that the systems based on machine learning methods have achieved much higher scores. In this case the heuristic-based system outperformed the one based on machine learning. It might be due to (1) relatively small size of the training dataset comparing to the PolEval 2018 tasks, and (2) some of the rare cases described in the guidelines were not sufficiently covered in the training dataset.

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

Broda B., Marcińczuk M., Maziarz M., Radziszewski A. and Wardyński A. (2012). *KPWr: Towards a Free Corpus of Polish*. In Calzolari N., Choukri K., Declerck T., Doğan 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)*, Istanbul, Turkey. European Language Resource Association.

Degórski Ł. (2012). *Towards the lemmatisation of Polish nominal syntactic groups using a shallow grammar*. In Bouvry P., Kłopotek M. A., Leprevost F., Marciniak M., Mykowiecka A. and Rybiński H. (eds.), *Security and Intelligent Information Systems: International Joint Conference (SIIS 2011). Revised Selected Papers*, number 7053 in Lecture Notes in Computer Science, pp. 370–378. Springer-Verlag.

Kieraś W. and Woliński M. (2017). *Morfeusz 2 – analizator i generator fleksyjny dla języka polskiego*. „Język Polski", XCVII(1), p. 75–83.

Małyszko J., Abramowicz W., Filipowska A. and Wagner T. (2015). *Lemmatization of Multi-Word Entity Names for Polish Language Using Rules Automatically Generated Based on the Corpus Analysis*. „Human Language Technologies as a Challenge for Computer Science and Linguistics", pp. 540–544.

Marcińczuk M. (2007). *CEN*. CLARIN-PL digital repository.

Marcińczuk M. (2017). *Lemmatization of Multi-word Common Noun Phrases and Named Entities in Polish*. In Mitkov R. and Angelova G. (eds.), *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2017)*, pp. 483–491. INCOMA Ltd.

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

Ogrodniczuk M., Bilińska J., Bronk Z. and Kieraś W. (2018). *Multisłownik: Linking plWordNet-based Lexical Data for Lexicography and Educational Purposes*. In Bond F., Kuribayashi T., Fellbaum C. and Vossen P. (eds.), *Proceedings of the 9th Global WordNet Conference (GWC 2018)*, pp. 368–375, Singapore. University of Tartu.

Oleksy M., Radziszewski A. and Wieczorek J. (2018). *KPWr Annotation Guidelines – Phrase Lemmatization*. CLARIN-PL digital repository.

Piskorski J., Kupść A. and Sydow M. (2007). *Lemmatization of Polish Person Names*. In *Proceedings of the Balto-Slavonic Natural Language Processing 2007*, pp. 27–34. The Association for Computational Linguistics.

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

Radziszewski A. (2013). *Learning to Lemmatise Polish Noun Phrases*. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL 2013). Volume 1: Long Papers*, pp. 701–709. The Association for Computer Linguistics.

Results of the PolEval 2019 Shared Task 3: Entity Linking

Aleksander Smywiński-Pohl (AGH University of Science and Technology)

Abstract

This paper concentrates on the description of the task of entity linking (EL) in 2019 edition of PolEval as well as the results of the task. The aim of the task of EL is the automatic disambiguation of expressions in a text that refer to entries in a knowledge base (KB). For the first time this task is defined for Polish using the following resources: Wikidata as the reference KB, Polish Wikipedia as the corpus used to train the disambiguation model and National Corpus of Polish (NKJP) as the corpus used to test the model. The paper provides the relevant details of these resources and the guidelines related to the manual annotation of NKJP used to determine the reference entities. The paper also presents the participating systems and their performance on the test data.

Keywords

entity linking, entity disambiguation, Wikipedia, Wikidata, National Corpus of Polish, dataset, Polish, natural language processing, evaluation

1. Introduction

Entity linking (Moro and Navigli 2015, Rosales-Méndez et al. 2018) covers the identification of mentions of entities from a knowledge base (KB). In this task as the reference KB we use Wikidata (WD)[1], an offspring of Wikipedia – a knowledge base that unifies structured data available in various editions of Wikipedia and links them to external data sources and knowledge bases. Thus making a link from a text to WD allows for reaching a large body of structured facts, including: the semantic type of the object, its multilingual labels, dates of birth and death for people, the number of citizens for cities and countries, the number of students for universities and many, many more. The identification of the entities is focused on the disambiguation of a phrase against WD. The scope of the phrase is provided in the test data, so the task boils down to the selection of exactly one entry for each linked phrase.

[1] `https://www.wikidata.org/`

Regarding Polish the research on EL is not much elaborated. In the past a system based on the Wikipedia Miner (Milne and Witten 2013) was adapted for Polish by Pohl (2012). The same system was also tested on a related task of Named Entity Recognition (Pohl 2013). A Polish model in the DBpedia Spotlight (Mendes et al. 2011) could be also obtained, yet it wasn't tested on any Polish EL dataset.

2. Task Description

The task covers the identification of mentions of entities from Wikidata in Polish texts. For instance the following text:

> Zaginieni 11-latkowie w środę rano wyszli z domów do szkoły w **Nowym Targu**, gdzie przebywali do godziny 12:00. Jak informuje „**Tygodnik Podhalański**", 11-letni Ivan już się odnalazł, ale los Mariusza Gajdy wciąż jest nieznany. Chłopcy od chwili zaginięcia przebywali razem między innymi w **Zakopanem**. Mieli się rozstać w czwartek rano.
>
> Source: `gazeta.pl`

has 3 entity mentions:

— Nowym Targu – `https://www.wikidata.org/wiki/Q231593`

— Tygodnik Podhalański – `https://www.wikidata.org/wiki/Q9363509`

— Zakopanem – `https://www.wikidata.org/wiki/Q144786`.

Even though there are more mentions that have their corresponding entries in WD (such as *środa*, *dom*, *12:00*, etc.) we restrict the set of entities to a closed group of WD types: names of countries, cities, people, occupations, organisms, tools, constructions, etc. (with important exclusion of times and dates). The full list of entity types is given in Appendix A. It should be also noted that names such as *Ivan* and *Mariusz Gajda* should not be recognized, since they lack corresponding entries in WD.

The task is similar to Named Entity Recognition (NER), with the important difference that in EL the set of entities is closed. To some extent EL is also similar to Word Sense Disambiguation (WSD), since mentions are ambiguous between competing entities.

In this task we have decided to ignore nested mentions of entities, so names such as *Zespół Szkół Łączności im. Obrońców Poczty Polskiej w Gdańsku, w Krakowie* which has an entry in Wikidata, should be treated as an atomic linguistic unit, even though there are many entities that have their corresponding Wikidata entries (such as *Poczta Polska w Gdańsku*, *Gdańsk*, *Kraków*). Also the algorithm is required to identify all mentions of the entity in the given document, even if they are exactly the same as the previous ones.

3. Data

3.1. Training Data

The most common training data used in EL is Wikipedia itself. Even though it wasn't designed as a reference corpus for that task, the structure of internal links serves as a good source for training and testing data, since the number of links inside Wikipedia is counted in millions. The important difference between the Wikipedia links and EL to Wikidata is the fact that the titles of the Wikipedia pages evolve, while the WD identifiers remain constant.

The second important difference is the fact that according to the Wikipedia editing rules, a link should be provided only for the first mention of any salient concept present in a page. It is different from the requirements of this task in which all mentions have to be identified.

The following training data is available:

— tokenised and sentence-split Wikipedia text – `http://poleval.pl/task3/tokens-with-entities.tsv.bz2`

— tokenised, sentence-split, tagged and lemmatized Wikipedia text – `http://poleval.pl/task3/tokens-with-entities-and-tags.tsv.bz2`

— list of selected Wikidata types – `http://poleval.pl/task3/entity-types.tsv`

— Wikidata items – `http://poleval.pl/task3/entities.jsonl.bz2`

— various data extracted from Wikipedia – the meaning of each file is provided in the readme.txt file – `http://poleval.pl/task3/wikipedia-data.tar.bz2`.

The data in the first and the second dataset have sentences separated by an empty line. Each line in the first dataset contains the following data (separated by tab character):

— `doc_id` – an internal Wikipedia identifier of the page; it may be used to disambiguate entities collectively in a single document (by using internal coherence of entity mentions)

— `token` – the value of the token

— `preceding_space` – 1 indicates that the token was preceded by a blank character (space in the most of the cases), 0 otherwise

— `link_title` – the title of the Wikipedia page that is a target of an internal link containing given token; some of the links point to pages that do not exist in Wikipedia; _ (underscore) is used when the token is not part of a link

— `entity_id` – the ID of the entity in Wikidata; this value has to be determined by the algorithm; _ (underscore) is used when the ID could not be established.

Sample data annotated according to the format is given in Appendix B. `Alfred V. Aho` and `Brian Kernighan` have their corresponding Wikidata IDs, since it was possible to determine them using the Wikipedia and Wikidata datasets. `Peter Weinberger` does not have the ID, even though there is an entry in Wikidata about him. Yet, there is no such page in the Polish Wikipedia and the link could not be established automatically. In the test set only the items

that have the corresponding Polish Wikipedia pages will have to be determined. Moreover, the algorithm will only have to determine the target of the link, not the span.

The second dataset is an extension of the first dataset with two additional columns:

— `lemma` – the lemma of the token determined by KRNNT tagger (Wróbel 2017)

— `morph_tags` – the morphosyntactic tags of the token determined by KRNNT.

A sample of the data is given in Appendix C.

3.2. Annotation Guidelines

This section provides the instructions given to the annotators during the annotation procedure. Each document was annotated by two annotators and in the case of competing annotations the final decision was made by a senior annotator.

The aim of the annotation

The aim of the annotation is the identification in Polish text one- and multi-word expressions, which have a corresponding page in the Polish Wikipedia. Such expressions should have the link added in the annotated text. Since the set of page types in Wikipedia is very broad (even words such as *nie* (*not*) have their corresponding site) the task is restricted to the information bearing elements, e.g.:

— names of people

— geographical names

— names of institutions

— names of species

— names of substances

— names of scientific procedures

— etc.

The full list of entity types is given in Appendix A. The set is broader than the set of proper names, since it includes names of species, substances, administrative titles, occupations, etc. The annotation should exclude the following elements (even if the document is pre-annotated with such expressions): punctuation symbols, mathematical symbols, references to dates, months, years, etc.

When considering the decisions whether to put an annotation, the annotator should ask themselves: „May anyone want to check the meaning of the expression in the Wikipedia?". If the answer is no, the annotation should not be placed. It might be rather strange to check words such as *car* or *house* for a typical language user. As a final rule yet, it is better to put the annotation rather than not.

Automatic annotation

The documents are prepared for the annotation – they were tokenized and some of the expressions have already one or more potential links to Wikipedia given for the last token of the expression. It does not mean that all of the identified expressions should be annotated, nor that some of them were not skipped. The annotator should read the document carefully and identify the potentially missing values. It should be noted that the pre-annotating algorithm does not contain all the name variants, so some names annotated in the other places, might not be annotated in a given text piece if the inflectional variant is different.

Annotation insertion

The annotator should check if there is a page in Wikipedia related to the given entity. If there is such a page, they should place it next to the first token and should provide a 'COPY' entry for all the other tokens being part of the expression. Otherwise _ (an underscore) should be placed in the column. There should be always only one link in the column, the remaining links should be removed.

If there is more than one Wikipedia link related to the entity, the one which better suites the context and better reflects the meaning of the expressions should be selected. A disambiguation page in Wikipedia should never be chosen as the destination for the link. If the entity is mentioned only on the disambiguation page, an underscore should be placed.

The scope of annotated expressions

Always the longest expressions which has a specific meaning and that has a page in Wikipedia should be annotated. The nested expressions should not be annotated as separate entities. It means that if a longer expressions has another expressions which has broader or other meaning, it has to treated as the indivisible element of the whole expression and as such it contains only COPY symbol (unless its the first token). E.g. all tokens of the expression *Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie* (first – by the link, the remaining – by the COPY symbol) should indicate the academy, even though some of them *Akademia*, *Staniasława Staszica*, and *Krakowie* have their corresponding Wikipedia pages.

The sub-expressions that are parts of names that do not have their corresponding Wikipedia page are treated the same way. E.g. *Masovia Maszewo soccer team* does not have its corresponding Wikipedia page, while *Maszewo* has, but the link should not be placed for it. Yet, for a common expression such as *math teacher* which does not have a Polish Wikipedia page, we annotate *math* and *teacher* separately. Similarly *maszewski klub* will have two links (to *Maszewo* and a *soccer club*), since the whole expression cannot be treated as a proper name and as a whole does not have a corresponding Wikipedia page. However, if such a page was present, the link should be attached to the whole expression and it should be treated as an atomic expression.

Semantic category of the expressions

It is admissible to allow for a slight change in the meaning of the expression and the link, under the assumption that it has a historical grounding. For instance the *Warsaw Voivodeship Court* was converted into the *Warsaw District Court* and there is no separate page for the former institution. On the contrary, *Poland* and *Second Polish Republic* have their separate pages, so in context when the word *Polska* refers to the second meaning, it should link to the second Wikipedia page.

It should be stressed that such changes in meaning **cannot change the semantic category of the entity**. If the text talks about *Poles* (people), the link cannot lead to *Poland* (country), since people and countries constitute separate semantic categories. If the text talks about the *marshal of the Małopolska province*, it cannot be linked to the *Małopolska province*, since the person and the institution are two separate types of entities.

As a final remark, the gender or sex of the expression should not be taken into consideration. *Nauczyciel* (teacher in masculine) and *nauczycielka* (teacher in feminine) should link to the same page, with the rare exclusion of the separate entries for the different gender versions of occupations (e.g. *king* and *queen*).

Adjectives

Name-based adjectives, such as *polski* (Polish), *warszawski* (Warsaw), *lekarski* (medical) should be annotated with a link to the source name (*Polska*, *Warszawa*, *Lekarz*).

Nested expressions within annotated expression

Elements that are not annotated separately, such as dates, when being a constituent of a larger names, e.g. *ul. 11 Listopada w Bielsku-Białej* (a specific street in a city that includes the date *11th of November*), should be treated the same as the other element of the expression, i.e. should have a COPY symbol or a link (if they are the first token of the expression). A special type of nested entities are punctuation marks. In the preceding example the dot should be annotated the same way as the other tokens. Yet, if the dot ends a sentence, it should not be annotated. Regarding the quotation marks – we assume that if the whole name is enclosed in quotation marks, they are not annotated. But if one of the marks is inside the name, the second one is annotated.

Specific vs. general links

If there is an expression such as *przekazania **Gminie** kompleksu* (referring to a regional institution) indicating that it is a reference to a specific institution, without direct indication of that institution, the annotation is dependent on the following condition. If the institution was indicated in the document previously, the link should be treated as a coreference and should point to that specific institution. Otherwise the link should point to the general page about the institution.

Selected types of non-annotated types

Some important types of entities that should be excluded from the annotation are listed below:

— disambiguation pages – `https://www.wikidata.org/wiki/Q16133130`

— time – `https://www.wikidata.org/wiki/Q11471`

— mental representations – `https://www.wikidata.org/wiki/Q2145290`

— numbers/amounts – `https://www.wikidata.org/wiki/Q11563`

— events – `https://www.wikidata.org/wiki/Q1656682`

— actions – `https://www.wikidata.org/wiki/Q4026292`

3.3. Testing Data

We annotated a part of the publicly available National Corpus of Polish to prepare the testing data. The dataset contains approx. 40 thousand tokens and includes 3195 identified entities. The sample of the test data (without the Wikipedia links and WD IDs) is given in Appendix D. During the disambiguation the algorithm is informed about 2 mentions, one spanning `Elwro` and another spanning `zakład`. The penultimate column is always empty, but kept in order to make the training and the test set look exactly the same. It should be noted that in the test data, mentions linking to the same entities have separate mention IDs, unless they form a continuous span of tokens. The test data is available for download under `http://poleval.pl/task3/task3_test.tsv.gz`.

4. Participating Systems

There were three systems participating in the competition: one baseline system (system-1), one system using vector representation for the words (Cheeky Mouse) and one heuristic-based system (zbronk.nlp.studio).

4.1. Baseline System

The first participating system was a baseline that selected the entity using simple statistical data from Wikipedia. At first a candidate entities were searched for using the length-based heuristic. Namely the mention to be linked was divided into continuous sub-expressions of length n (starting with the length of the expression) and the sub-expressions were searched for in the set of all links present in Wikipedia. If such links were present, the page with the largest number of links *from this particular sub-expression* was selected and the ID of the entity was determined by the Wikipedia – Wikidata mapping. If there wasn't any link found for a given n, sub-expressions with $n-1$ were explored until $n=0$.

4.2. Cheeky Mouse

The second participating system used word2vec embeddings (Mikolov et al. 2013) to disambiguate between the competing candidates for the expression. The candidates were generated as all permutations of the expressions appearing as the links in Wikipedia. The word2vec embeddings were trained on the National Corpus of Polish (Przepiórkowski et al. 2012, Pęzik 2012) and were used to compute the vector representation for the expressions appearing in Wikipedia (the context included the preceding, the central and the following sentences of the link). The vectors were averaged to compute one representation of the link and the surrounding context. Similar representation was computed when the expression was disambiguated and the candidate with the highest cosine-similarity was selected as the winning entity.

4.3. zbronk.nlp.studio

The third participating system used a cascade of heuristic rules for reaching the disambiguation decision. First of all the system used the lemmatized version of the phrase. Then the phrase was checked against the *dictionary of predefined links* (described later). If it matched, the predefined link was selected as the winning entity. The next heuristic checked if the link refers to a nation. If so, the process was finished and the Wikidata node referring to that nation was selected as the winning candidate. The next heuristic checked if the lemma of the name is adjectival form of one of the entries in *Multidictionary*. If so, the process was completed, with that entry (having a mapping to the Wikidata entry) was selected as the winning entity. The next heuristic checked for links present in Wikipedia. If there was a link or a number of links, the target page (and the related Wikidata entry) with the largest number of links was selected as the winning entry. Similarly next heuristic checked for the lemmatized variant of the name in the Wikipedia page titles (with the disambiguation part removed). The next heuristic checked if the part of the name is a name of a person appearing in Wikipedia – a match finished the process. At the end, the system used Wikidata search function for finding the candidates.

The dictionary of predefined links was constructed on the basis of **manual annotation of the test set**, i.e. expressions disambiguated manually were added to the dictionary and served as the first choice for ambiguous entries. Although such an approach was not directly stated as forbidden by the task creator, it could not be accepted as a solution for the task. Such an approach was not far away from manually annotating the test set and sending such an annotation as the solution, thus solving the task manually. The idea of PolEval is the promotion of systems that may be applied in a broad spectrum of applications, rather than for the particular dataset. Providing manual annotations and using them will not lead to general applicability of the solution.

5. Evaluation

The number of correctly identified entities divided by the total number of entities to be identified was used as the evaluation measure. If the system does not provide an ID for the entity mention, the case is treated as an invalid answer.

6. Results

The results achieved by the systems are given in Table 1. The result of zbronk.nlp.studio is the best (91.9%), yet it should be stressed that the system creator annotated the test data and used that annotation in the final solution. The second score (77.2%) was achieved by the baseline system that used the most probable target Wikipedia page as the disambiguation result. The most sophisticated system i.e. Cheeky Mouse achieved the worst result (26.2%).

Table 1: The results achieved by the participating systems

System	**Score**
zbronk.nlp.studio*	91.9%
system-1	77.2%
Cheeky Mouse	26.7%

*the system included knowledge from the test set in the solution

7. Conclusion

Inspecting the number of participating systems and their performance, we may observe that the solution sent for the competition were not much elaborated. We hope that the availability of the annotated data and the PolEval competition will make the task more interesting for a larger number of participating teams. We consider submitting the same task for the next PolEval edition to spark more interest in the problem. Yet, we also acknowledge the difficulty of the task and the fact that providing a solution for it might be time consuming.

Acknowledgments

This work was supported by the Polish National Centre for Research and Development – LIDER Program under Grant LIDER/27/0164/L-8/16/NCBR/2017 titled „Lemkin – intelligent legal information system".

References

Mendes P. N., Jakob M., García-Silva A. and Bizer C. (2011). *DBpedia Spotlight: Shedding Light on the Web of Documents*. In *Proceedings of the 7th International Conference on Semantic Systems*, pp. 1–8. ACM.

Mikolov T., Sutskever I., Chen K., Corrado G. and Dean J. (2013). *Distributed Representations of Words and Phrases and Their Compositionality*. In *Proceedings of the 26th International Conference on Neural Information Processing Systems – Volume 2*, pp. 3111–3119. Curran Associates Inc.

Milne D. and Witten I. H. (2013). *An Open-source Toolkit for Mining Wikipedia*. „Artificial Intelligence", 194, p. 222–239.

Moro A. and Navigli R. (2015). *SemEval-2015 Task 13: Multilingual All-words Sense Disambiguation and Entity Linking*. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pp. 288–297.

Pęzik P. (2012). *Wyszukiwarka PELCRA dla danych NKJP*. In Przepiórkowski A., Bańko M., Górski R. and Lewandowska-Tomaszczyk B. (eds.), *Narodowy Korpus Języka Polskiego*, pp. 253–279. Wydawnictwo Naukowe PWN.

Pohl A. (2012). *Improving the Wikipedia Miner Word Sense Disambiguation Algorithm*. In *Proceedings of the 2012 Federated Conference on Computer Science and Information Systems*, pp. 241–248. IEEE.

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

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

Rosales-Méndez H., Hogan A. and Poblete B. (2018). *VoxEL: A Benchmark Dataset for Multilingual Entity Linking*. In *International Semantic Web Conference*, pp. 170–186. Springer.

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.

Appendices

A. Entity types

Type name	Wikidata link
human	https://www.wikidata.org/wiki/Q5
geographic location	https://www.wikidata.org/wiki/Q2221906
academic discipline	https://www.wikidata.org/wiki/Q11862829
anatomical structure	https://www.wikidata.org/wiki/Q4936952
occupation	https://www.wikidata.org/wiki/Q12737077
vehicle model	https://www.wikidata.org/wiki/Q29048322
construction	https://www.wikidata.org/wiki/Q811430
written work	https://www.wikidata.org/wiki/Q47461344
astronomical body	https://www.wikidata.org/wiki/Q6999
clothing	https://www.wikidata.org/wiki/Q11460
taxon	https://www.wikidata.org/wiki/Q16521
mythical entity	https://www.wikidata.org/wiki/Q24334685
type of sport	https://www.wikidata.org/wiki/Q31629
supernatural being	https://www.wikidata.org/wiki/Q28855038
liquid	https://www.wikidata.org/wiki/Q11435
political system	https://www.wikidata.org/wiki/Q28108
group of living things	https://www.wikidata.org/wiki/Q16334298
chemical entity	https://www.wikidata.org/wiki/Q43460564
publication	https://www.wikidata.org/wiki/Q732577
landform	https://www.wikidata.org/wiki/Q271669
language	https://www.wikidata.org/wiki/Q34770
unit	https://www.wikidata.org/wiki/Q2198779
physico-geographical object	https://www.wikidata.org/wiki/Q20719696
intellectual work	https://www.wikidata.org/wiki/Q15621286
tool	https://www.wikidata.org/wiki/Q39546
organism	https://www.wikidata.org/wiki/Q7239
food	https://www.wikidata.org/wiki/Q2095

B. Sample of the training data

Doc. #	Token	Space	Wikipedia page	Wikidata ID
2	Nazwa	1	_	_
2	języka	1	_	_
2	pochodzi	1	_	_
2	od	1	_	_
2	pierwszych	1	_	_
2	liter	1	_	_
2	nazwisk	1	_	_
2	jego	1	_	_
2	autorów	1	_	_
2	Alfreda	1	Alfred V. Aho	Q62898
2	V	1	Alfred V. Aho	Q62898
2	.	0	Alfred V. Aho	Q62898
2	Aho	1	Alfred V. Aho	Q62898
2	,	0	_	_
2	Petera	1	Peter Weinberger	_
2	Weinbergera	1	Peter Weinberger	_
2	i	1	_	_
2	Briana	1	Brian Kernighan	Q92608
2	Kernighana	1	Brian Kernighan	Q92608
2	i	1	_	_
2	czasami	1	_	_
2	jest	1	_	_
2	zapisywana	1	_	_
2	małymi	1	_	_
2	literami	1	_	_
2	oraz	1	_	_
2	odczytywana	1	_	_
2	jako	1	_	_
2	jedno	1	_	_
2	słowo	1	_	_
2	awk	1	_	_
2	.	0	_	_

C. Sample data with lemma and morphosyntactic tags

The Wikipedia page title and the Wikidata ID were skipped in order to preserve space.

#	Token	Lemma	Space	Tags
2	Nazwa	nazwa	0	subst:sg:nom:f
2	języka	język	1	subst:sg:gen:m3
2	pochodzi	pochodzić	1	fin:sg:ter:imperf
2	od	od	1	prep:gen:nwok
2	pierwszych	pierwszy	1	adj:pl:gen:f:pos
2	liter	litera	1	subst:pl:gen:f
2	nazwisk	nazwisko	1	subst:pl:gen:n
2	jego	on	1	ppron3:sg:gen:m1:ter:akc:npraep
2	autorów	autor	1	subst:pl:gen:m1
2	Alfreda	Alfred	1	subst:sg:gen:m1
2	V	V	1	subst:sg:nom:n
2	.	.	0	interp
2	Aho	Aho	1	subst:sg:gen:m1
2	,	,	0	interp
2	Petera	Peter	1	subst:sg:gen:m1
2	Weinbergera	Weinbergera	1	subst:sg:gen:m1
2	i	i	1	conj
2	Briana	Brian	1	subst:sg:gen:m1
2	Kernighana	Kernighana	1	subst:sg:gen:m1
2	i	i	1	conj
2	czasami	czas	1	subst:pl:inst:m3
2	jest	być	1	fin:sg:ter:imperf
2	zapisywana	zapisywać	1	ppas:sg:nom:f:imperf:aff
2	małymi	mały	1	adj:pl:inst:f:pos
2	literami	litera	1	subst:pl:inst:f
2	oraz	oraz	1	conj
2	odczytywana	odczytywać	1	ppas:sg:nom:f:imperf:aff
2	jako	jako	1	prep:nom
2	jedno	jeden	1	adj:sg:nom:n:pos
2	słowo	słowo	1	subst:sg:nom:n
2	awk	awk	1	subst:pl:gen:n
2	.	.	0	interp

D. Sample of the testing data

#	Token	Lemma	Space	Tags	_	Mention ID
2240	Pracownice	pracownica	0	subst:pl:nom:f	_	_
2240	Elwro	Elwro	1	subst:sg:gen:n	_	e1
2240	:	:	0	interp:	_	_
2240	-	-	1	interp:	_	_
2240	To	to	1	subst:sg:nom:n	_	_
2240	boli	boleć	1	fin:sg:ter:imperf	_	_
2240	,	,	0	interp:	_	_
2240	bo	bo	1	comp:	_	_
2240	nam	my	1	ppron12:pl:dat:f:pri	_	_
2240	się	się	1	qub:	_	_
2240	ciągle	ciągle	1	adv:pos	_	_
2240	wydawało	wydawać	1	praet:sg:n:imperf	_	_
2240	,	,	0	interp:	_	_
2240	że	że	1	comp:	_	_
2240	to	to	1	pred:	_	_
2240	jest	być	1	fin:sg:ter:imperf	_	_
2240	nasz	nasz	1	adj:sg:nom:m3:pos	_	_
2240	zakład	zakład	1	subst:sg:nom:m3	_	e2
2240	.	.	0	interp:	_	_

PolEval 2019: Entity Linking

Szymon Roziewski, Marek Kozłowski, Łukasz Podlodowski
(National Information Processing Institute)

Abstract

This work presents our results of participation in Entity Linking task at PolEval 2019. The goal of the task was to identify the meaning of entities from a knowledge base in Polish Wikipedia texts. The data contain texts from Polish Wikipedia, given in a structured form. Each data entity consists of specific information, regarding word entity itself, its exact part of speech, and for text mentions, the Wikipedia link, and Wikidata id, in addition. We have used a hybrid approach for solving this task. The main idea was to filter out entities that suit for simple mapping, and the rest which was "hiding" behind different context or textual form, were directed to another model. After mention candidates were found, we proceeded with semantic filtering of them, with respect to the entity context. This procedure was performed by using word2vec model, trained on a train set.

Keywords

entity linking, natural language processing, computational linguistics

1. Introduction

PolEval is a challenge dedicated to tasks of Natural Language Processing of Polish. There are several tasks announced, and the competitors work on models and tools that provide a solution to a specific problem. In this work, our efforts are targeted at Task 3, related to Entity Linking problem. Every highlighted word in the sentence is called an entity, and it has a specific contextual meaning, marked by Wikidata label. Therefore, besides Wikipedia text, Wikidata set is concerned. The task was to find a certain semantic definition for a given entity. All entities within repeated ones, have to be mapped to their meanings. The organizers provided structured textual dataset consisting of Polish Wikipedia texts.

1.1. Data Set

Two data sets are available, one for training and another for final model evaluation. The training set contains documents consisting of sentences, in which mentions are specified. The training data are specifically organized in a structure. Each document is made of sentences, those are built from tokens. The token is the smallest item forming a chunk of data. The data file contains rows of data, where each row expresses a token within some extended information i.e. document id, token, preceding space, Wikipedia link title, entity id from Wikidata. There is also an extended training data set, in which one can find some more knowledge, token lemma and part of speech. The files weights are 9.6 GB and 18 GB, respectively. The raw file contains 1.88M of documents, 25.6M of sentences, 348M of tokens and 42M of entities to be linked to their meanings. There are other data sets which provide some additional material i.e. a list of entity types (Wikidata IDs) and structured information of Wikidata documents with relations between them. In additional file, 3M of entities definitions are available. Each item in this set contains id label, English and Polish entity name and classifying type from Wikidata, concerning hierarchical knowledge.

The test set contains similar data, consists of 2 documents, 3628 sentences, and 33179 tokens, 4071 entities to be linked to meanings, and 1.5 MB in weight. Mentions are already detected, the entity labels are given as a sequence of naming elements $e_1 \dots e_n$, where n is the number of all entities.

A file in JSON format[1], describing Wikidata entities (referred later as meanings) consists of more than 3M items. One can find some helpful information in it, for a given entity, e.g. entity ID, both Polish and English: entity labels, and Wikipedia page name. There is also some hierarchical knowledge provided as traits e.g. lists of subclasses and instances of classes, to which a given entity belongs.

As a preprocessing step, stopwords and very short sentences (shorter than three words) were removed.

1.2. Entity Linking Approach

Our solution to Entity Linking task is made of three steps. The first stage is a simple filtering. Unique entities with no disambiguation issues that have been discovered from the training set were stored in a map within their Wikidata labels. The elements of the map were formed of key and value. The keys were made of entity value. The entities that could not be linked to their meanings were stacked with their value and missing label from Wikidata, and we used them to enrich our dictionary for better lining performance. If the entity consisted of several words, we put permutated patterns as keys with value as Wikidata label. Thus, the first step was filtering out those entities that were unique on the basis of the training set. When the entity had more than one meaning (e.g. word *Unix* had three different meanings depending on the given context: 1. family of computer operating systems that derive from the original AT&T Unix; 2. Unix command; 3. standard UNIX utility) we treated all detected meanings as candidates. After training word2vec (Mikolov et al. 2013b) model we selected the best

[1] https://en.wikipedia.org/wiki/JSON

candidate based on its context in the given Wikipedia text. As context, we reckon surrounding sentences i.e. preceding, current and following. As we can imagine, giving the correct answer is quite a complex task.

2. Related Work

Raiman and Raiman (2018) outperformed multilingual entity linking task. They incorporated symbolic structures into the reasoning process of a deep neural network without attention. Thus, the loss function is defined with a label hierarchy. The entity meaning is disambiguated based on the type system. They constructed a type system, and subsequently, and made use of it to constrain the results of a neural network and to respect the symbolic structure. The design problem is reformulated into a mixed integer problem, the type system is created and a neural network is trained with it. Now, the discrete variables pick which parent-child relations from an ontology are types within the type system. In the meantime, continuous variables adjust a classifier fit to the type system (Raiman and Raiman 2018). This classifier discovers long-term dependencies in the input data that let it reliably predict types. They use a bidirectional-LSTM (Lample et al. 2016), fed with word, prefix and suffix embeddings as in (Andor et al. 2016).

2.1. Word2Vec

Word2vec computes semantic neural network based vector representation of words, which has been shown to help the machine learning algorithms to boost their scores in natural language processing tasks (Mikolov et al. 2013a). Such computed word vectors are positioned in the vector space such that words that share common contexts in the corpus are located in close proximity to one another in the space. Continuous bag-of-words and skip-gram architectures yield vector representations of words, which are useful in various natural language processing applications such as machine translation, named-entity recognition, word sense disambiguation, tagging, parsing, etc. Skip-gram model is an idea of learning word vector representations, which are useful in predicting contextual words given a target word.

The target word is fed to the input layer, and the context words are produced by the output layer.

The aim of the skip-gram model is to maximize the function of average log-likelihood, given a sequence of training contextual words $w_1, w_2, \ldots, w_T$

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{j=-k}^{k}\log p(w_{t+j}|w_t), \tag{1}$$

where k stands for the size of the training window.

The skip-gram model learns vector representations of words based on their patterns of co-occurrence in the training corpus as follows: it assigns to each word in the vocabulary V a “context” and a “target” vector, respectively u_w and v_w, which are to be used in order to

predict the words that appear around each occurrence of w_i within a window of k tokens. Specifically, the log probability of any target word w_i to occur at any position within distance k of a context word w_j is taken to be proportional to the inner product between u_w and v_w. The probability of accurately predicting word w_i given w_j is described by the softmax model

$$p(w_i|w_j) = \frac{\exp(u_{w_i}^\top v_{w_j})}{\sum_{l=1}^{V} \exp(u_l^\top v_{w_j})} \tag{2}$$

where V is the size of the vocabulary. The skip-gram model featured with softmax is computationally expensive; computing $\log p(w_i|w_j)$ is proportional to vocabulary size, which can reach the size of tens of millions. For boosting model efficiency, we use hierarchical softmax with lower computation demands bounded by $O(\log(V))$ as shown by Mikolov et al. (2013a).

Negative sampling technique is used to reduce noise in the resulting vectors; it also affects the training process by improving speed and convergence. In this procedure, more frequent words are more likely to be chosen as negative samples. Mikolov et al. (2013a) claim that equation (3) outperformed other candidates for probability function of selecting a word w_i from the corpus.

$$P(w_i) = \frac{f(w_i)^{3/4}}{\sum_{j=0}^{n} f(w_j)^{3/4}} \tag{3}$$

The function (3) has the tendency to promote less frequent words and to neglect more frequent ones.

2.2. Learning Phrases

Learning Phrases is an innovation added to word2vec in comparison to (Mikolov et al. 2013b). Phrase detection is covered in (Mikolov et al. 2013a). The Learning Phrase tool passes through the corpora and looks at a combination of 2 words, though by running multiple times, one can catch longer phrases. The relative number of combinations of two words appearing in the training text is used to determine which word combinations to convert to phrases. The algorithm is made to select a combination as a phrase, when words together occur often relative to the number of individual appearances.

3. Solution

3.1. Dataset Preprocessing

The dataset preprocessing was focused mainly on removing Polish stop words and punctuation marks. In order to complete the first task, we used large and rich Polish corpora, formed by the Polish community of linguists and computational linguistics scientists. We extracted a list of Polish stop words, which contains 350 items. We make use of The National Corpus of Polish[2]

[2] http://nkjp.pl/

(Pol. Narodowy Korpus Języka Polskiego; NKJP; Przepiórkowski 2012, Przepiórkowski et al. 2008). NKJP is a shared initiative of four institutions: Institute of Computer Science at the Polish Academy of Sciences (coordinator), Institute of Polish Language at the Polish Academy of Sciences, Polish Scientific Publishers PWN, and the Department of Computational and Corpus Linguistics at the University of Łódź. In particular, NKJP contains a list of sources for the corpora: classic literature, daily newspapers, specialist periodicals and journals, transcripts of conversations, and a variety of short-lived and internet texts. NKJP comes in two variants, the first one is closed, since it contains copyrighted material (∼1.5 billion words), and the second one is open and contains about 1 million words. A smaller part of NCP is distributed under a GNU GPL license in TEI-compliant XML format.

After making the efforts of the preprocessing step, we truncated some unnecessary elements i.e. punctuation marks, empty items. We obtained a list of sentences, that was used to train the semantic model. Besides, some interesting insights were gained. In Table 1, we state the overview of datasets. In the training set, there are 1.1M unique entities out of all 42M entities, 5.7M unique words out of 348M, and 25.6M sentences. Whereas, in the test set, there are 2.7K unique entities out of 4K, 13K unique words out of 33K, and 3.6K sentences.

Table 1: Datasets exploration

	Training set	**Test set**
Entities	42.27×10^6	4071
Unique entities	1.1×10^6	2707
Tokens	348.39×10^6	33179
Unique tokens	5.74×10^6	13315
Stopwords	69.98×10^6	9907
Punctuation marks	66.79×10^6	3646
Sentences	25.58×10^6	3628

3.2. Word2vec Training

We trained a semantic model using Gensim (Řehůřek and Sojka 2010) package, which is a Python (van Rossum and Drake 2011) implementation of word2vec. We use continuous bag-of-words algorithm approach, which is dedicated to word prediction based on the context. Using a CBOW model is beneficial for some other reasons i.e. training is several times faster than the skip-gram, and it has slightly better accuracy for the frequent words (Mikolov et al. 2013b). Now we discuss some of the model parameters. We set up the size of the semantic space for word vectors to 300. The maximum distance between the current and predicted word within a sentence is 2. Subsequently, negative sampling technique was used within the training process, so as to ease the training process itself and to improve the quality of the final semantic vectors, by limiting the "noise words". Learning phrases mode was also applied, in order to create more concise semantic representations.

In our application, we have used the Gensim (Řehůřek and Sojka 2010) implementation of word2vec. Gensim implementation is fast enough to process the sentences coming from Wikipedia training text, in less than one day. Additionally, in an effort to reduce memory usage, our training pipeline takes advantage of iterators. The model is being trained in an online fashion, by fetching documents one after another from the database (a file containing all sentences extracted from the training set). Finally, the resulting model is about 1.5 GB large, which makes it possible to train it even on machines with a modest amount of available RAM.

Table 2 presents words with entries closest in meaning. In addition, the distance between those two semantic vectors is attached. The output is considered to be semantically coherent. We arbitrarily chose a bunch of vectors to be shown here.

Table 2: Examples of semantic similarities based on word2vec trained on Polish Wikipedia Data (PolEval 2019 Entity Linking train set)

Word	**Most similar**	**Distance**
owies (*oat*)	jęczmień (*barley*)	0.87
pietruszka (*parsley*)	marchew (*carrot*)	0.83
kubek (*mug*)	filiżanka (*cup*)	0.83
apple	macintosh	0.81
dziewczyna (*girl*)	chłopak (*boy*)	0.80
Bach	Mozart	0.76
islam (*Islam*)	chrześcijaństwo (*Christianity*)	0.74
Kubica (*Polish F1 driver*)	Michael Schumacher	0.73
spragniony (*thirsty*)	głodny (*hungry*)	0.73
Gdańsk	Gdynia	0.64

A few examples of linguistic computations based on vector space representation are shown in Table 3. Thanks to word2vec we have a direct link between the mathematical representation and the semantic meaning of a word. We post an overview of selected candidates, for which one can find semantic regularities appealing.

3.3. Entity Linking

In order to solve the entity linking task, as we stated before, we extracted entities without ambiguity issues from training text, to form a big dictionary for meaning mapping. Since we knew which entities meanings were missing during the naive approach, we added them within the entity to the map, to gain more accuracy. The key was made of entity's Wikipedia page name, and the value was a tuple of Wikidata ID, and a Polish label for ID. Afterwards, performing a simple meaning matching with respect to Wikipedia page name of an entity, for items having only one meaning, yielded 58% accuracy on the training text. Using Levenshtein (Miller et al. 2009) distance, more possible definition candidates were involved. We put a

Table 3: Linguistic regularities in vector space

Expression	**Nearest token**	**Distance**
samochód + rower (*car* + *bicycle*)	motocykl (*motorcycle*)	0.71
jezioro + las (*lake* + *forrest*)	bagno (*swamp*)	0.68
ptak − zwierzę + samolot (*bird* − *animal* + *airplane*)	myśliwiec (*fighter plane*)	0.65
sosna − roślina + zwierzę (*pine* − *plant* + *animal*)	żubr (*aurochs*)	0.60
król − mężczyzna + kobieta (*king* − *man* + *woman*)	królowa (*queen*)	0.58
dobry − zły (*good* − *bad*)	najlepszy (*best*)	0.58

restriction for meanings fetched by Levenshtein measure to have a similar prefix to given entity value. Entities with more than one meaning (Wikidata ID) were treated by word2vec model to choose a candidate based on the entity context. The context was made of words extracted from surrounding sentences, with a word restriction of having the same part of speech as an entity had. We took into account three sentences for entity context formulation, i.e. preceding, current and succeeding. In order to make it happen, we demanded definitions of meanings, thus we perceived them from Wikidata by making use of their API, and receiving the desired definitions for given Wikidata IDs. Searching for accurate applicants was done in a specific way. We had a bunch of candidates for entities with ambiguous meaning, i.e. having more than one definition. For each meaning candidate, we computed a mean semantic vector of words coming from its Wikidata definition, by taking the arithmetic mean. Similarly, a mean vector related to the entity was created from semantic vectors of words belonging to the entity context, and having the same part of speech as the entity. After performing the *distance* function on trained word2vec model on these two mean semantic vectors, for each meaning candidates, we chose the candidate with the closest distance. In such a way, we selected semantically best candidates for their meaning, regarding a specific entity context, and different meanings definitions.

4. Summary

Our model managed to link 97% of entities to their meanings, but the precision on the test set was only the 26.7%. The possible reason was related to miserable semantic context extraction by applying only a mean over selected words, both for entity context and meanings definitions. Neglecting the hierarchical structure of entities could have a major impact as well.

Currently, much more efficient semantic models have been developed i.e. ELMo (Embeddings from Language Models; Peters et al. 2018) and BERT (Bidirectional Encoder Representations from Transformers; Devlin et al. 2018). The ELMo model is a new type of deep contextualized word representation that can model complex characteristics and cross-linguistic variations. All in all, BERT model obtains new state-of-the-art results on eleven natural language processing tasks (Devlin et al. 2018).

We believe that making use of entities hierarchical structure along with one of those state-of-the-art models, would be crucial in gaining much higher accuracy on this Entity Linking task.

References

Andor D., Alberti C., Weiss D., Severyn A., Presta A., Ganchev K., Petrov S. and Collins M. (2016). *Globally Normalized Transition-Based Neural Networks*. „CoRR", abs/1603.06042.

Devlin J., Chang M., Lee K. and Toutanova K. (2018). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. „CoRR", abs/1810.04805.

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

Mikolov T., Sutskever I., Chen K., Corrado G. S. and Dean J. (2013a). *Distributed Representations of Words and Phrases and their Compositionality*. In Burges C., Bottou L., Welling M., Ghahramani Z. and Weinberger K. (eds.), *Advances in Neural Information Processing Systems 26*, pp. 3111–3119. Curran Associates, Inc.

Mikolov T., Chen K., Corrado G. and Dean J. (2013b). *Efficient Estimation of Word Representations in Vector Space*. „CoRR", abs/1301.3781.

Miller F. P., Vandome A. F. and McBrewster J. (2009). *Levenshtein Distance: Information Theory, Computer Science, String (Computer Science), String Metric, Damerau Levenshtein Distance, Spell Checker, Hamming Distance*. Alpha Press.

Peters M., Neumann M., Iyyer M., Gardner M., Clark C., Lee K. and Zettlemoyer L. (2018). *Deep Contextualized Word Representations*. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Przepiórkowski A. (2012). *Narodowy Korpus Języka Polskiego*. Wydawnictwo Naukowe PWN.

Przepiórkowski A., Górski R. L., Lewandowska-Tomaszczyk B. and Łaziński M. (2008). *Towards the National Corpus of Polish*. In Calzolari N., Choukri K., Maegaard B., Mariani J., Odijk J., Piperidis S. and Tapias D. (eds.), *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC 2008)*, Marrakech. ELRA.

Raiman J. and Raiman O. (2018). *DeepType: Multilingual Entity Linking by Neural Type System Evolution*. In McIlraith S. A. and Weinberger K. Q. (eds.), *Proceedings of the 32nd AAAI*

Conference on Artificial Intelligence, (AAAI-18), the 30th Innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), pp. 5406–5413. AAAI Press.

Řehůřek R. and Sojka P. (2010). *Software Framework for Topic Modelling with Large Corpora*. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pp. 45–50, Valletta, Malta. ELRA. `http://is.muni.cz/publication/884893/en`.

van Rossum G. and Drake F. L. (2011). *The Python Language Reference Manual*. Network Theory Ltd.

Results of the PolEval 2019 Shared Task 4: Machine Translation

Krzysztof Wołk (Polish-Japanese Academy of Information Technology)

Abstract

This article summarizes the first PolEval shared task on machine translation into Polish. The focus of the task was to develop translation systems between Polish-English and Polish-Russian language pairs. It was possible to implement such systems with any technology, with or without automatic pre- or post-processing tools. Machine translation system authors were allowed to submit training data constrained result or open systems (those did not take part in competition). The evaluation was done using four most popular automatic metrics (BLEU, NIST, TER, METEOR) in their original form. The best constrained system between PL and EN scored 16.29 BLEU point whereas best unconstrained 28.23. For PL to RU it was 5.38 and 12.71 respectively and for RU to PL 5.73 and 11.45 respectively.

Keywords

Polish, natural language processing, machine translation, neural translation, MT evaluation, SMT

1. Introduction

The PolEval series is an annual Polish language processing contest organized by Institute of Computer Science in collaboration with other institutions companies. The third edition of the PolEval (2019) contest featured six shared tasks one of which was Machine Translation. Machine Translation is a translation of text with a computer, with no human involvement. Pioneered in the 1950s, Machine Translation may also be known as automatic or instantaneous translation. Currently, there are three most popular forms of machine translation system that usually operate individually or as hybrid solutions: rules-based, statistical and neural:

— Rules-based systems use a combination of language and grammar rules plus dictionaries for common words. Specialist dictionaries are created to focus on certain industries or

disciplines. Rules-based systems typically deliver consistent translations with accurate terminology when trained with specialist dictionaries.

— Statistical systems have no knowledge of language rules. Instead they "learn" to translate by analyzing large amounts of data for each language pair. They can be trained for specific industries or disciplines using additional data relevant to the sector needed. Typically, statistical systems deliver more fluent-sounding but less consistent translations.

— Neural Machine Translation (NMT) is a new approach that makes machines learn to translate through one large neural network (multiple processing devices modelled on the brain). The approach has become increasingly popular amongst MT researchers and developers, as trained NMT systems have begun to show better translation performance in many language pairs compared to the phrase-based statistical approach.

2. Task Description

As the training data set, we have prepared a set of bi-lingual corpora aligned at the sentence level. The corpora were saved in UTF-8 encoding as plain text, one language per file. We divided the corpora as in-domain data and out-domain data. Using any other data was not permitted. The in-domain data was rather hard to translate because of its topic diversity. In-domain data were lectures on different topics. As out of domain data we accepted any corpus from `http://opus.nlpl.eu` project. Any kind of automatic pre- or post- processing was also accepted. The in-domain corpora statistics are given in Table 1.

Table 1: Task 4 corpora statistics

	Segments		**Unique tokens**			
	TEST	**TRAIN**	**TEST**		**TRAIN**	
			INPUT	**OUTPUT**	**INPUT**	**OUTPUT**
EN to PL	10 000	129 254	9 834	16 978	49 324	100 119
PL to RU	3 000	20 000	6 519	7 249	32 534	32 491
RU to PL	3 000	20 000	6 640	6 385	32 491	31 534

The participants were asked to translate with their systems test files and submit the results of the translations. The translated files should be aligned at the sentence level with the input (test) files. Submissions that were not aligned were not accepted. If any pre- or post-processing was needed for the systems, it was supposed to be done automatically with scripts. Any kind of human input into test files was strongly prohibited.

3. Participating Systems

As part of the evaluation preparation we prepared baseline translation systems. For this purpose we used out of the box and state of the art ModernMT machine translation system (Jelinek 2004). It was created in cooperation of Translated, FBK, UEDIN and TAUS. ModernMT also has secondary neural translation engine. From the code on Github we know that it is based on PyTorch (Paszke et al. 2017) and OpenNMT (Klein et al. 2017); it also uses BPE for sub-word units (Sennrich et al. 2015) generation as default. More detailed information is unknown and not stated on the project manual pages. The project probably has many more default optimizations. We did not do any kind of data pre- or post-processing nor any system adaptation. We simply used our data with default ModernMT settings. Out of the curiosity we also translated our test samples with the Google engine. Please note that these results are not comparable because Google engine was not restricted to any data. For EN to PL translation ModernMT obtained 16.29 BLEU points, whereas Google engine scored 16.83. For PL to RU we obtained 12.71 versus 15.78 for Google; for RU to PL the scores were 11.45 and 13.54 respectively. The highest scores were obtained by the Samsung team but the system was not open. The authors focused on data pre-processing by preparing a clean high quality Polish-English parallel corpora, supplementing the large out-of-domain data, over-sampled scarce in-domain data. They tried adding back-translation data but unfortunately in that particular case it did not improve the performance. Authors experimented with a variety of training schedules as the transformer architecture that was very sensitive to hyper-parameters. The obtained BLEU score equals to 28.23 in EN to PL task. The competition winner was neural based translation system prepared by National Information Processing Institute. For the task, the authors prepared two different approaches based on neural computing and rule-based system. For EN-PL subtask they applied a deep neural network called the Transformer. This method is a state-of-the-art solution for a WMT 2014 English-to-German task. For PL-RU (in both directions) we prepared a rule-based model which statistically learned a dictionary of lemmas. We applied a few grammatical rules which allowed them to use translated lemmas and part of speech tags to conjugate words and transfer grammatical information to the translated sentence.

4. Evaluation

The evaluation itself was done with four main automatic metrics widely used in machine translation:

— BLEU (Papineni et al. 2002)

— NIST (Doddington 2002)

— TER (Snover et al. 2006)

— METEOR (Banerjee and Lavie 2005).

BLEU was developed on a premise similar to that used for speech recognition, described by Papineni et al. (2002) as: “The closer a machine translation is to a professional human

translation, the better it is." Hence, the BLEU metric is designed to measure how close SMT output is to that of human reference translations. It is important to note that translations, SMT or human, may differ significantly in word usage, word order, and phrase length (Papineni et al. 2002). To address these complexities, BLEU attempts to match phrases of variable length between SMT output and the reference translations. Weighted match averages are used to determine the translation score (Axelrod 2006). A number of variations of the BLEU metric exist. The basic metric requires calculation of a brevity penalty P_b as follows:

$$P_b = \begin{cases} 1, c > r \\ e^{1-\frac{r}{c}}, c \leq r \end{cases} \tag{1}$$

where r is the length of the reference corpus, and candidate (reference) translation length is given by c (Graves and Schmidhuber 2005). The basic BLEU metric is then determined as shown in (Axelrod 2006):

$$BLEU = P_b \exp \sum_{N}^{n=0} w_n \log p_n \tag{2}$$

where w_n are positive weights summing to one, and the n-gram precision p_n is calculated using n-grams with a maximum length of N. There are several other important features of BLEU. Word and phrase positions in the text are not evaluated by this metric. To prevent SMT systems from artificially inflating their scores by overuse of words known with high confidence, each candidate word is constrained by the word count of the corresponding reference translation. The geometric mean of individual sentence scores, by considering the brevity penalty, is then calculated for the entire corpus (Axelrod 2006). The NIST metric was designed to improve BLEU by rewarding the translation of infrequently used words. This was intended to further prevent inflation of SMT evaluation scores by focusing on common words and high-confidence translations. The NIST metric thus uses heavier weights for rarer words. The final NIST score is calculated using the arithmetic mean of n-gram matches between SMT and reference translations. In addition, a smaller brevity penalty is used for smaller variations in phrase length. The reliability and quality of the NIST metric has been shown to be superior to the BLEU metric (Tai et al. 2015). The Metric for Evaluation of Translation with Explicit Ordering (METEOR) considers more directly several factors that are indirect in BLEU. Recall (the proportion of matched n-grams to total reference n-grams) is used directly in this metric. Moreover, METEOR explicitly measures higher-order n-grams, considers word-to-word matches, and applies arithmetic averaging for a final score. The best matches against multiple reference translations are used (Snover et al. 2006). The METEOR method uses a sophisticated and incremental word alignment method that starts by considering exact word-to-word matches, word stem matches, and synonym matches. Alternative word order similarities are then evaluated based on these matches. The calculation of precision is similar in the METEOR and NIST metrics. Recall is calculated at the word level. To combine the precision and recall scores, METEOR uses a harmonic mean. It rewards longer n-gram matches (Snover et al. 2006). The METEOR metric is calculated as shown in (Snover et al. 2006):

$$METEOR = \frac{10PR}{R + 9P}(1 - P_M) \tag{3}$$

where the unigram recall and precision are given by R and P, respectively. The brevity penalty P_M is determined by:

$$P_M = 0.5\frac{C}{M_U} \tag{4}$$

where M_U is the number of matching unigrams, and C is the minimum number of phrases required to match unigrams in the SMT output with those found in the reference translations.

5. Results and Conclusions

The competition winner team was from National Information Processing Institute. They proposed translation solutions to all three translation tasks using only in-domain data. Those systems were better than in-domain baseline systems and obviously worse than system using additionally out of domain data. The best system in English to Polish task was prepared by the Samsung research team. The best scoring systems were neural-based and utilized the Transformer architecture. In the results we can observe big disproportion in scores between rule-based (SIMPLE_SYSTEMS) and neural systems (DeepIf and SRPOL). The results for EN to PL task are given in the Table 2, for PL to RU in the Table 3 and for RU to PL in the Table 4. Please note that Google results cannot be compared directly. Google Translate was trained with bigger and unknown amount of data.

Table 2: EN-PL Results

System name	BLEU	NIST	TER	METEOR
SRPOL	28.23	6.60	62.13	47.53
Google Translate	16.83			
ModernMT	16.29			
ModernMT (in-domain)	14.42			
DeepIf (in-domain)	4.92	2.27	86.56	21.74
SIMPLE_SYSTEMS	0.94	1.12	97.94	9.81

Table 3: PL-RU Results

System name	BLEU	NIST	TER	METEOR
Google Translate	15.78			
ModernMT	12.71			
DeepIf (in-domain)	5.38	2.53	83.02	53.54
SIMPLE_SYSTEMS	0.69	0.85	102.75	41.06

Table 4: RU-PL Results

System name	BLEU	NIST	TER	METEOR
Google Translate	13.54			
ModernMT	11.45			
ModernMT (in-domain)	5.73			
DeepIf (in-domain)	5.51	2.97	85.27	24.08
SIMPLE_SYSTEMS	0.57	1.29	109.43	8.35

Acknowledgments

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

References

Axelrod A. E. (2006). *Factored Language Models for Statistical Machine Translation*. Master's thesis, University of Edinburgh.

Banerjee S. and Lavie A. (2005). *METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments*. In *Proceedings of the ACL workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pp. 65–72.

Doddington G. (2002). *Automatic Evaluation of Machine Translation Quality Using N-gram Co-occurrence Statistics*. In *Proceedings of the 2nd International Conference on Human Language Technology Research*, pp. 138–145. Morgan Kaufmann Publishers Inc.

Graves A. and Schmidhuber J. (2005). *Framewise Phoneme Classification with Bidirectional LSTM and Other Neural Network Architectures*. „Neural Networks", 18(5), p. 602–610.

Jelinek R. (2004). *Modern MT Systems and the Myth of Human Translation: Real World Status Quo*. In *Proceedings of the International Conference Translating and the Computer 26*.

Klein G., Kim Y., Deng Y., Senellart J. and Rush A. M. (2017). *OpenNMT: Open-Source Toolkit for Neural Machine Translation*. „CoRR", abs/1701.02810.

Papineni K., Roukos S., Ward T. and Zhu W.-J. (2002). *BLEU: a Method for Automatic Evaluation of Machine Translation*. In *Proceedings of the 40th Annual Meeting of Association for Computational Linguistics*, pp. 311–318. Association for Computational Linguistics.

Paszke A., Gross S., Chintala S., Chanan G., Yang E., DeVito Z., Lin Z., Desmaison A., Antiga L. and Lerer A. (2017). *Automatic Differentiation in PyTorch*.

Sennrich R., Haddow B. and Birch A. (2015). *Neural Machine Translation of Rare Words with Subword Units*. „CoRR", abs/1508.07909.

Snover M., Dorr B., Schwartz R., Micciulla L. and Makhoul J. (2006). *A Study of Translation Edit Rate with Targeted Human Annotation*. In *Proceedings of Association for Machine Translation in the Americas*, pp. 223–231.

Tai K. S., Socher R. and Manning C. D. (2015). *Improved Semantic Representations from Tree-Structured Long Short-Term Memory Networks*. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 1556–1566, Beijing, China. Association for Computational Linguistics.

The Samsung's Submission to PolEval 2019 Machine Translation Task

Marcin Chochowski, Paweł Przybysz (Samsung Research Institute Poland)

Abstract

This paper describes the submission to the PolEval 2019 Machine Translation task by Samsung R&D Institute, Poland. We focused on preparing a clean high quality Polish-English parallel corpora, supplementing the large out-of-domain data, over-sampled scarce in-domain data. Unfortunately adding back-translation data in that particular case did not improve the performance. We also experimented with a variety of training schedules as the transformer architecture we used is very sensitive to hyper-parameters. Our submission was ultimately produced by a single system not using system ensembles.

Keywords

neural machine translation, transformer, natural language processing, computational linguistics, linguistic engineering

1. Introduction

Neural machine translation (NMT) is a deep learning based approach for machine translation, which yields the state-of-the-art translation performance for translation directions where large-scale parallel corpora are available. Moreover, vanilla NMT performs poorly in scenarios where domain-specific translations are required and there is no or only scarce in-domain corpora. In such cases it is very important to prepare high-quality and well balanced training corpora so that the model can improve in its domain keeping good quality in general domain. In this paper, we give a description of our submission to PolEval 2019 Machine Translation task, with exactly that constraints i.e. there is a specific domain to adapt, but only scarce in-domain corpora and significantly bigger out-of-domain corpora are available.

2. Training Data

In brief, we used all of the provided in-domain parallel training data along with permissible parallel data from Opus[1] repository. Table 1 lists the individual parallel corpora that made up our training data. Note that the in-domain PolEval corpora was over-sampled 10 times so that in-domain data constitutes noticeable fraction of the training set. Note also that not all corpora were used in all experiments. In particular the final submission did not utilize the synthetic data.

Table 1: Statistics for the parallel training corpora used in our submission. The raw sentence pair count corresponds to original size, while the filtered sentence pair count to corpus size after realignment and removing pairs not classified as proper translations. All corpora except PolEval are considered out-of-domain.

Corpus	**Sentence pairs (raw)**	**Sentence pairs (filtered)**	**Filtering rate**
Bible	0.03 M	0.02 M	79.81%
Books	0.002 M	0.001 M	40.09%
DGT	3 M	2.8 M	92.57%
ECB	0.07 M	0.04 M	66.74%
EMEA	0.9 M	0.7 M	78.36%
Eubookshop	0.5 M	0.3 M	65.68%
Euconst	0.008 M	0.007 M	91.40%
Europarl.7	0.6 M	0.6 M	98.09%
GlobalVoices	0.04 M	0.03 M	87.38%
GNOME	0.006 M	0.005 M	81.83%
IATE	0.1 M	0.05 M	44.81%
JRC-Acquis	1.3 M	1.2 M	93.08%
KDE4	0.1 M	0.1 M	63.62%
OpenSubtitles.2018	41.3 M	34.7 M	83.95%
Paracrawl.v1	1.3 M	0.9 M	75.76%
PHP	0.03 M	0.02 M	75.76%
Tanzil	0.1 M	0.1 M	88.35%
Tatoeba	0.002 M	0.002 M	98.20%
TED	0.2 M	0.2M	93.54%
Ubuntu	0.006 M	0.003 M	46.16%
Wikipedia	0.1 M	0.1 M	68.50%
wikititles	0.7 M	0.1 M	18.79%
PolEval 2019 in-domain	10 x 0.1 M	10 x 0.1 M	81.35%
Synthetic in-domain	5 x 0.1 M	5 x 0.1 M	100.00%
Total	52.3 M	43.7 M	81.69%

[1] `http://opus.nlpl.eu`

2.1. In-Domain Data

We used all of the available in-domain data, which was rather scarce compared to permissible out-of-domain data. We noticed that the in-domain data were poorly aligned. Manual inspection showed that it probably comes from subtitles to movies of lectures as the sentences were split into several lines, thus the alignment was often lost. We also noticed that there is a significant overlap between the training set and the development set. Out of 10.000 sentences of the development set, 2620 were also present in the training set. These were not only short, common sentences but also long complex ones. Not to allow drawing wrong conclusion during training we cut out all leaking sentences from the development set, making it 7380 sentence pairs. We also considered realigning the whole training corpus by re-splitting it to sentences – one sentence per line not per few lines – and aligning it with target side. Yet, since the test set was supposed to look similarly to development set, we decided not to, so that the model during training sees similar examples as during evaluation.

2.2. Out-of-Domain Data

As any corpus from OPUS was permissible, we crawled the OPUS website for Polish-English corpora. Table 1 summarizes parallel training corpora we used in our submission. Although at first glance most of these corpora, maybe except TED talks, are in different domain than the development set, there might be some potential overlap in terminology. There are some texts on medical science that can be covered by EMEA corpus or political or historical text that can be close to Europarl. All in all we decided not to prune available out-of-domain corpora in other way than quality filtering.

2.3. Data Filtering

As the quality of parallel corpora is often very poor, and rubbish data have significantly stronger impact on neural models than it used to have for statistical models we put a lot of effort in filtering out all spurious sentence pairs from training set. Our filtering mechanism is based on alignment scores we assign to every sentence pair. To obtain a sentence alignment score, we follow the idea that using parallel corpora we can train a statistical fast-align model to predict probable alignments (Dyer et al. 2013). We trained an IBM Model2 statistical alignment model for Polish-English and applied it to score each parallel training sentence. We also scored each sentence pair with a sentence-level language recognition tools. After these operations each sentence pair had assigned fast-align score and language recognition scores. We selected small subset of 3k sentences from the corpora and performed manual evaluation for each sentence pairs scoring from 1 (very bad) to 5 (very good). Then we trained a regression model to predict human score based on fast-align and language recognition scores. Finally, we used the regression model to score whole parallel corpora and select potentially good sentences (predicted score above 3). We also removed empty lines, lines with only dots and with Wiki markup as we observed negative impact of such lines in our baseline model. Corpus sizes after these steps are shown in column filtered. The filtering method removed less than

10% of high quality corpora like Europarl.7 or DGT, but it removed over 50% of Books or wikititles corpus.

Even in the small, available in-domain data set we found spurious sentence pairs like the ones showed in Table 2. We filtered out almost 20% of PolEval 2019 in-domain corpus, mostly due to misalignment caused by sentence breaks.

Table 2: Examples of spurious sentence pairs filtered out from in-domain training set

Polish	**English**
$2 * 16 \prod cm^2$	and now we have to figure out the surface area of this thing that goes around
Prawda? 1+2+3+...+k+k+1 jest sumą wszystkich liczb z włączeniem k+1	Well we are assuming that we know what this already is.

2.4. Synthetic Data

Back-translated monolingual in-domain data has been shown to be very beneficial when added to the parallel training data (Sennrich et al. 2016, Przybysz et al. 2017, Williams et al. 2018). From our experience, it is also sometimes useful to back-translate the target side of parallel in-domain corpora to produce additional synthetic source so that we get two corpora with the same target side but slightly different source side. For the back-translation we trained two models. Both were self-attention Transformer models (Vaswani et al. 2017). We used an efficient, C++ written open-source Marian library (Junczys-Dowmunt et al. 2018) for training and decoding. The detailed Transformer parameters correspond to what usually is called transformer-big, i.e. 6 -layered encoder and decoder, hidden size of size 1024, feed-forward layer of size 4096, 16 attention heads and all embeddings tied. First we trained a baseline model Polish to English using only out-of-domain corpora to get a reference point. On the reversed development set it reached 29.28 BLEU. In the second model training we added in-domain corpora oversampled 10 times. With the second model we got significantly better performance reaching 33.51 BLEU. We used the letter model to produce back-translation of the in-domain corpora and we did not filter the resulting corpus.

3. System Description

Our submission is a self-attention Transformer model (Vaswani et al. 2017). Again in all experiments we used Marian for training and decoding. All models were the same as the ones used for back-translation i.e. transformer-big with 6 -layered encoder and decoder, hidden size of size 1024, feed-forward layer of size 4096, 16 attention heads and all embeddings tied. The only difference between experiments were slightly different hardware settings that only influence the speed and did not effect on the final performance.

We trained our models on multi-GPU server with 8 Tesla v100 32GB cards using synchronous Adam optimizer with parameters $\beta_1 = 0.9$, $\beta_2 = 0.98$ and norm clipping on 5. From our previous experience with training Transformer models we also set learning rate warmup to 32000 updates and after that inverted square learning rate decay. We have experimented with increasing the effective batch size by summing gradients over increasing number of batches before performing model update. So on the one hand we used 8 GPU cards with huge RAM size reserving 25GB for data. This alone made the effective batch size of more than 20k sentences. On the other we multiplied this by a factor of up to 16 by setting the "optimizer-delay" parameter of Marian. We noticed however that using the delay may lead to unstable training. We experienced this several times, in particular the best performing model was trained in two steps. The first, was run for 30000 updates after which the training broke down (blue lines on Figure 1). We restarted the training starting from the best model, cleared the Adam optimizer history matrices and lowering the learning rate slightly. The second step also experienced similar but significantly less disturbing break down, and again it seems to correspond to learning rate inflection with multi-step update (see lower panel of Figure 1). We are investigating this phenomena presuming this might be a implementation issue. The convergence condition was set to 20 updates without improving the cost function. The training converged after 220 hours reaching maximum of 27.32 BLEU on the development set (note that development set was trimmed as described in Section 2). Unfortunately, adding synthetic data did not help, but rather decreased the performance (see green plots on Figure 1). The model with synthetic data reached only 21.97 BLEU which is over 5 BLEU point worse than the best model with no synthetic data. We account this fact to poor quality of the in-domain data that we back-translated. On the input there are often only parts of sentences as sentences are often split into several lines. The back-translating model was trained on set where the vast majority of the data was properly segmented to sentences, so the decoder learned to produce sentence-like output. This results in synthetic corpus which is actually out-of-domain since its syntax differs from the real in-domain data.

4. Conclusions

This paper describes the Samsung's R&D Institute Poland Machine Translation team to the PolEval 2019 task for English-Polish translation direction. We report strong results that are significantly better than on-line available translators. We assign this on one hand to particular domain that we were able to adapt to, and on the other to specific format of the test data that often contains only parts of sentences. This 'specificity' of in-domain corpus is also evident in fact that synthetic data did not improve the performance. Our experimental results confirm the effectiveness of using larger batch sizes by accumulating gradients over several batches before doing model update.

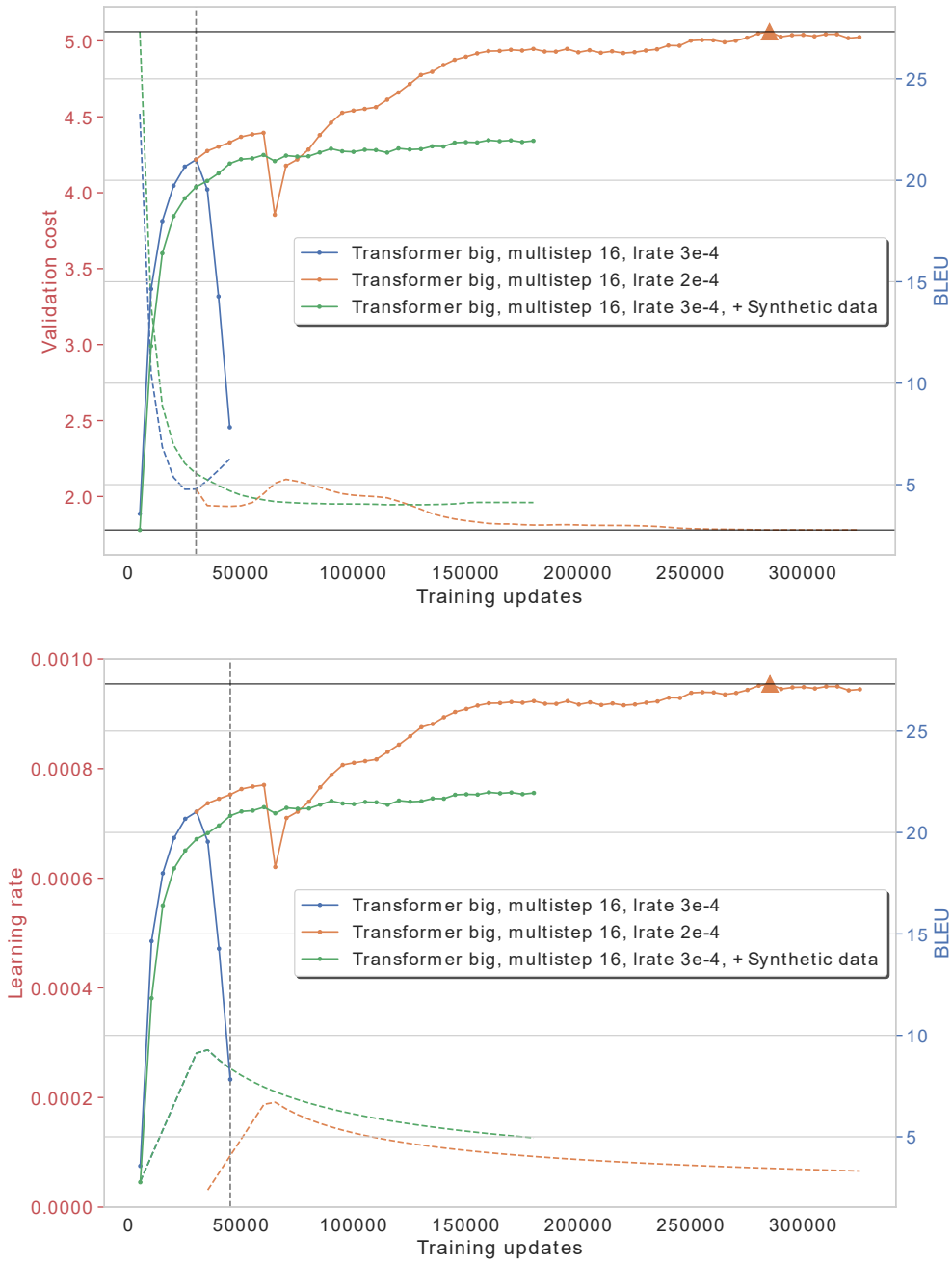

Figure 1: Submitted model's training progress. The upper panel plots the progress of BLEU on the development set with corresponding cross entropy, while the lower panel shows the same BLEU with value of the learning rate during training. The blue lines depict first step of the training, and the orange lines depict the second step – continuation.

References

Dyer C., Chahuneau V. and Smith N. A. (2013). *A Simple, Fast, and Effective Reparameterization of IBM Model 2*. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 644–648, Atlanta, Georgia. Association for Computational Linguistics.

Junczys-Dowmunt M., Grundkiewicz R., Dwojak T., Hoang H., Heafield K., Neckermann T., Seide F., Germann U., Fikri Aji A., Bogoychev N., Martins A. F. T. and Birch A. (2018). *Marian: Fast Neural Machine Translation in C++*. In *Proceedings of ACL 2018, System Demonstrations*, pp. 116–121, Melbourne, Australia. Association for Computational Linguistics.

Przybysz P., Chochowski M., Sennrich R., Haddow B. and Birch-Mayne A. (2017). *The Samsung and University of Edinburgh's submission to IWSLT17*. In Sakti S. and Utiyama M. (eds.), *Proceedings of the 14th International Workshop on Spoken Language Translation*, pp. 23–28.

Sennrich R., Haddow B. and Birch A. (2016). *Improving Neural Machine Translation Models with Monolingual Data*. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 86–96, Berlin, Germany. Association for Computational Linguistics.

Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser Ł. and Polosukhin I. (2017). *Attention Is All You Need*. In *Advances in Neural Information Processing Systems*, pp. 5998–6008.

Williams P., Chochowski M., Przybysz P., Sennrich R., Haddow B. and Birch A. (2018). *Samsung and University of Edinburgh's System for the IWSLT 2018 Low Resource MT Task*. In *Proceedings of the 15th International Workshop on Spoken Language Translation*, pp. 118–123.

English-Polish and Polish-Russian Translation Systems

Łukasz Podlodowski, Marek Kozłowski, Szymon Roziewski (National Information Processing Institute)

Abstract

This paper presents a proposed solution for a PolEval 2019 Task 4: Machine translation[1]. We propose two models to solve the problem of natural language translation. For English to Polish translation deep attention-based neural network was used. For Russian-Polish (both directions) rule-based model was applied. For the second language pair, we trained a statistically based dictionary using typical word's alignment models using the distributions in training data.

Keywords

natural language processing, machine translation, rule-based system, deep neural network, attention mechanism, Transformer

1. Introduction

This paper presents a proposed solution for a PolEval 2019 Task 4: Machine translation[1]. The task was divided into two sub-tasks based on pairs of languages:

1. English to Polish, in this paper we will call this task EN-PL
2. Polish-Russian (both directions), PL-RU.

Organizers have prepared a set of bi-lingual in-domain corpora aligned at the sentence level for each sub-task. Additionally, participants could use out-of-domain data provided by OPUS – open parallel corpus[2].

[1] http://poleval.pl/tasks/task4
[2] http://opus.nlpl.eu/

2. Problem Details

EN-PL in-domain set contains approximately 130 thousand pairs of sentences. However, the PL-RU in-domain set contained only 23 000 pairs. The training of the model could be extended through out-of-domain OPUS data. Due to limited time and computation power, we decided to focus only on in-domain data.

The cardinality of training sets was one of the main aspects taken into account for a suitable model selection. Because of the difference between English and Polish languages and big enough training set we decided to use the deep attention-based neural network called Transformer for EN-PL. A small number of training examples, the high similarity of grammar between Russian and Polish lead us to choose a rule-based model for PL-RU.

To measure the quality of solution organizers proposed four metrics:

1. BLEU (Papineni et al. 2002)
2. NIST (Doddington 2002)
3. TER (Snover et al. 2006)
4. METEOR (Banerjee and Lavie 2005).

Datasets were extracted from the subtitles of movies. This the reason that some of the data could be noised. Movies subtitles were focused on providing the meaning of the text. Some sentences could be translated by taking into account a wider context, not available on a sentence level.

3. Related Work

Since a decade, machine translation and computer-aided translations have become an active field of research in the machine learning community. Most of the approaches are based on effective translation memory population, or statistical and neural methods consuming parallel corpora. In machine translation (MT), translation systems are trained on large quantities of parallel data (from which the systems learn how to translate small segments), as well as even larger quantities of monolingual data (from which the systems learn what the target language should look like). Parallel data is a collection of corresponding texts in two different languages, which is sentence-aligned, in that each sentence in one language is matched with its corresponding translated sentence in the other language. It is also known as a bitext.

Given a parallel text, i.e. the same text in two languages, aligning the different language versions of that text at a sentence level is the necessary first step for building a corpus-based machine translation system (e.g. statistical MT (SMT), neural-based ones (NMT) or example-based MT). Also building a translation memory from existing parallel corpora typically requires text aligned at the sentence level.

The training process in SMT takes in the parallel data and uses co-occurrences of words and segments (known as phrases) to infer translation correspondences between the two languages

of interest. In phrase-based machine translation, these correspondences are simply between continuous sequences of words, whereas in hierarchical phrase-based machine translation or syntax-based translation, more structure is added to the correspondences (Koehn et al. 2007). One of the most popular java based SMT toolkit is Joshua (Li et al. 2009). It is an open-source toolkit of statistical machine translation decoder for phrase-based, hierarchical, and syntax-based machine translation, written in Java. The Joshua 6 release introduces a phrase-based decoder that uses the standard priority-queue-based decoding algorithm to construct a hypergraph whose format is shared with the existing CKY+-based hierarchical decoding algorithms.

Neural machine translation (NMT) is an approach based on a large artificial neural network to predict the likelihood of a sequence of words. Deep neural machine translation is an extension of neural machine translation, that processes multiple neural network layers instead of just one.

Traditional phrase-based statistical translation systems are breaking up source sentences into multiple segments and then translated them iteratively. The most crucial disadvantage of such an approach is the disfluency of the outputs in a human matter. In NMT we read the entire source sentence, understand its meaning, and then produce a translation. NMT systems first read the source sentence using an encoder to build a fixed size vector, a sequence of float numbers that represent the input sentence. Then a decoder processes the sentence vector to infer a translation. This is known as the encoder-decoder architecture (Cho et al. 2014). NMT resolves some typical statistical translation problems as long-range dependencies e.g., gender agreements; syntax structures. Neural based models produce much more fluent translations as demonstrated by Google Neural Machine Translation systems (Wu et al. 2016).

Neural machine translation (NMT) uses of vector representations (continuous space representations) for words and internal states. The structure of the models is simpler than statistical phrase-based models. There is no separate language model, translation model, and reordering model, but just a single sequence model that predicts one word at a time. The translation prediction is conditioned on the entire source sentence and the entire already produced target sequence (Bahdanau et al. 2014).

NMT models vary in terms of their exact architectures. A natural choice for sequential data is the recurrent neural network (RNN), used by most NMT models. Usually, an RNN is used for both the encoder and decoder. The RNN models, however, differ in terms of: (a) directionality – unidirectional or bidirectional; (b) depth – single- or multi-layer; and (c) type – often either a vanilla RNN, a Long Short-term Memory (LSTM), or a gated recurrent unit (GRU; Luong et al. 2015).

In this paper, if we have a statistically significant number of training data we consider a neural network based Transformer model. The Transformer follows this overall architecture using stacked self-attention and point-wise, fully connected layers for both the encoder and decoder (Vaswani et al. 2017). We show an example of such models in Section 4. Otherwise, if we do not have enough training instances, we decided to use the dictionary and rule-based approach.

4. Model Description

For the task, we prepared two different approaches based on neural computing and rule-based system.

4.1. PL-RU

For this subtask, we decided to use the rule-based model. We used popular lemmatizers to prepare data:

— for Russian: pyMorphy2[3] (Korobov 2015) which allows to lemmatize Russian words, provides set of tags of grammatical forms and additionally allows to conjugate words based on lemma and set of tags

— for Polish: Morfologik[4] to lemmatize words and Morfeusz[5] (Woliński 2006) to generate conjugation of words.

Our system determines the translation of lemmatized words used in a sentence. Then tried to construct a correct grammatical form of the translated sentence. Because of the complexity of a problem we did not use a morphosyntactic identification of lemmas. This fact provided additional noise to data processing. We were aware of the possible quality of system improvement and we consider adding this aspect in our future works.

The system works in the following steps:

1. lemmatize an input sentence
2. collect lemmas and part of speech tags
3. translate each lemma based on a dictionary
4. apply grammatical rules
5. apply postprocessing rules
6. return translated sentence.

In this section, we will describe translation only in one direction: from Russian to Polish. However we used the same, but inverted rules in an opposite direction.

Preparing dictionary

The first step was preparing a dictionary of lemmatized words. We removed stopwords from datasets then we used a simple formula to score candidates for a translation of a term:

$$\text{score}(\text{w}_{\text{pl}}) = \frac{|RU_{w_{ru}} \cap PL_{w_{pl}}|^2}{|RU_{w_{ru}}| + |PL_{w_{pl}}|}$$

[3] https://github.com/kmike/pymorphy2
[4] https://github.com/morfologik/
[5] http://sgjp.pl/morfeusz

where $RU_{w_{ru}}$ is a set of sentence pairs when the Russian term occurred and $PL_{w_{pl}}$ – set of sentence pairs where the Polish term occurred.

We used the formula above as a criterion function on which we based choosing a translation for a term. This computation could be done efficiently by using an inverted index concept.

Two dictionaries were evaluated one based on the whole possible matching and a second one with a limited set of candidates. We chose the most probable translation for a term based on the part of speech tags returned by lemmatizators. Groups of tags were linked based on domain level knowledge. Prepared groups are presented in Table 1.

Table 1: Matched groups of tags from pyMorphy2 and Morfologik lemmatizers

pyMorphy2	**Morfologik**
NOUN, NUMR	subst, depr, burk, ger, num
ADJF, ADJS, COMP	adj, adja, adjc, adv
VERB, INFN	verb, ger
PRTF, PRTS, GRND	pant, ppas, pcon
NPRO	ppron12, ppron3, siebie
ADVB	adv,adjp
PRED	pred
PREP	prep
CONJ	conj, comp
PRCL	qub
INTJ	interj

Finally, we used a third dictionary of stopwords evaluated in a similar way, but on the dataset with removed all terms which was not on the stopwords lists. However some of stopwords occurred much often (i.e. „что", „nie") in one of the languages or did not occur at all („się"). Because of that this small stopwords dictionary was manually corrected by humans. Choosing a dictionary for a translated word was based on the order:

1. part of speech limited
2. dictionary without stopwords
3. stopwords.

When the dictionary did not contain translated term next in order dictionary was chosen. Finally, if none of the dictionaries had Russian term we did not translate word but returned its original form.

Grammar rules

In our system we used only six grammatical rules which were utilized based on the lemmatizers grammatical tags:

1. **Future and present tense** – we assumed the translation of the verbs in the future or present tense could be done one-to-one between both languages. We kept information about plurality and person.
2. **Imperative mood** – similar too previous rule we translated verbs in the imperative mood in a one-to-one way.
3. **Past participle** – translate a sequence of a personal pronoun and full participle into a past participle form. Include all grammar changes from a Nouns rule.
4. **Short participle** – translate short participle into a Polish passive adj. participle. Include all grammar changes from a Nouns rule.
5. **Nouns** – translate nouns directly, however, recognize lemma grammar form changes between Polish and Russian versions, i.e. different plurality or gender. Keep this information in cache until the next noun is detected. If the previous word was an adjective, conjugated it again with new grammar tags.
6. **Adjectives** – translate adjectives directly, but include all grammar changes from a Nouns rule.

Postprocessing

Our method contained a sequence of postprocessing rules associated with:

1. Translate „что" as Polish conjunction „że" if after „что" occurred a personal pronoun and it is not a first word in a sentence. The Russian term „что" could be also translated as asking pronoun, however in that case it usually occurs at the begging of sentence.
2. Removing a personal pronoun for a pair: pronoun + verb, i.e for Russian sentence „Я знал" our system generate Polish translation „Ja znałem". It's a correct translation, however using the personal pronoun in that combination in the Polish language is rare. Instead of that often used is only a short version of conjugated verb (in previous example: „Znałem.")
3. Removing „давать" in a pair „давать" + verb.
4. Restore an original punctuation of the sentence and a capitalization of each word.

4.2. EN-PL

For EN-PL subtask we used a neural network based Transformer model. The Transformer follows this overall architecture using stacked self-attention and point-wise, fully connected layers for both the encoder and decoder. There are two kinds of attention units. Scaled Dot-Product Attention and Multi-Head Attention. In addition to attention sub-layers, each of the layers in our encoder and decoder contains a fully connected feed-forward network, which is applied to each position separately and identically. It could be interpreted as two convolutions with kernel size 1.

This method is a state-of-the-art solution for a WMT 2014 English-to-German task (Vaswani et al. 2017). Model architecture is presented in Figure 1.

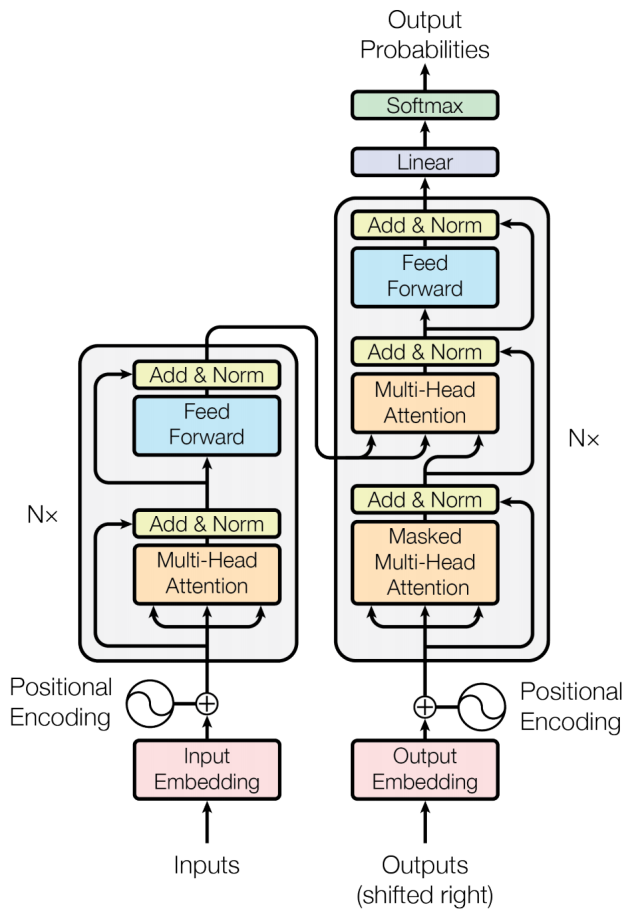

Figure 1: Transformer model (Vaswani et al. 2017)

Attention mechanism

The attention weights are the relevance scores of the input encoder hidden states (values with size d_v), in processing the decoder state (query with size d_k). This is calculated using the encoder hidden states (keys with size d_k) and the decoder hidden state. For the Scaled Dot-Product Attention computation is based on the dot products of the query with all keys, divide each by $\sqrt{d_k}$ and apply a softmax function to obtain the weights on the values.

For the Multi-Head Attention authors of the Transformer proposed utilizing Multi-Head Attention as a beneficial alternative to a single attention function. This approach linearly projects the queries, keys, and values h times with different, learned linear projections to d_k,

d_k and d_v dimensions, respectively. On each of these projected versions of queries, keys, and values we then perform the attention function in parallel, yielding d_v-dimensional output values. These are concatenated and once again projected, resulting in the final values. Both attention layers are presented in Figure 2.

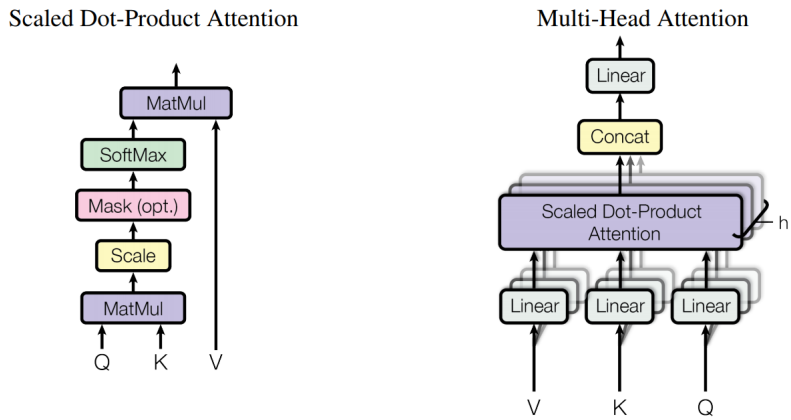

Figure 2: Scaled Dot-Product Attention (left), Multi-Head Attention (right) consists of several attention layers running in parallel (Vaswani et al. 2017)

Transformer parametrization was based on the original base parameters set in (Vaswani et al. 2017). Due to the more complex morphosyntactic nature of the Polish language, we increase the number of hidden units in feedforward layers to 4096. Additionally to prevent overfitting and reduce data noisiness we cut off all words which occurred less than 5 times in the dataset.

5. Results and Conclusion

For each of the subtask of the challenge, we prepared two different models based on significant different approaches. The metrics values of the solution are presented in Table 2.

Table 2: Results of the presented systems

Subtask	BLEU	NIST	TER	METEOR
English to Polish	4.92	2.27	86.56	21.74
Polish to Russian	5.38	2.53	83.02	53.54
Russian to Polish	5.51	2.97	85.27	24.08

For PL-RU subtask we showed that a small number of grammatical rules could provide promising results even for a small dataset. These rules could be used to improve or support machine learning systems. Since Polish grammar contains plenty of grammatical forms for

a single lemma directly approaching neural computation based solutions used in machine translation between English and German is limited. Since these models do not recognize the semantic and grammatical similarity to lemma classification space is significantly bigger. The more complicated problem leads to more data needs and more complex neural networks.

References

Bahdanau D., Cho K. and Bengio Y. (2014). *Neural Machine Translation by Jointly Learning to Align and Translate*. „CoRR", abs/1409.0473.

Banerjee S. and Lavie A. (2005). *METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments*. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pp. 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Cho K., Van Merriënboer B., Bahdanau D. and Bengio Y. (2014). *On the Properties of Neural Machine Translation: Encoder-Decoder Approaches*. „CoRR", abs/1409.1259.

Doddington G. (2002). *Automatic Evaluation of Machine Translation Quality Using N-gram Co-occurrence Statistics*. In *Proceedings of the 2nd International Conference on Human Language Technology Research (HLT 2002)*, pp. 138–145, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Koehn P., Hoang H., Birch A., Callison-Burch C., Federico M., Bertoldi N., Cowan B., Shen W., Moran C., Zens R. et al. (2007). *Moses: Open Source Toolkit for Statistical Machine Translation*. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (Companion volume: Proceedings of the Demo and Poster Sessions)*, pp. 177–180.

Korobov M. (2015). *Morphological Analyzer and Generator for Russian and Ukrainian Languages*. In Khachay M. Y., Konstantinova N., Panchenko A., Ignatov D. I. and Labunets V. G. (eds.), *Analysis of Images, Social Networks and Texts*, vol. 542 of *Communications in Computer and Information Science*, pp. 320–332. Springer International Publishing.

Li Z., Callison-Burch C., Dyer C., Ganitkevitch J., Khudanpur S., Schwartz L., Thornton W. N., Weese J. and Zaidan O. F. (2009). *Joshua: An Open Source Toolkit for Parsing-based Machine Translation*. In *Proceedings of the 4th Workshop on Statistical Machine Translation*, pp. 135–139. Association for Computational Linguistics.

Luong M.-T., Pham H. and Manning C. D. (2015). *Effective Approaches to Attention-based Neural Machine Translation*. „CoRR", abs/1508.04025.

Papineni K., Roukos S., Ward T. and Zhu W.-J. (2002). *BLEU: A Method for Automatic Evaluation of Machine Translation*. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL 2002)*, pp. 311–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

Snover M., Dorr B., Schwartz R., Micciulla L. and Makhoul J. (2006). *A Study of Translation Edit Rate with Targeted Human Annotation*. In *Proceedings of Association for Machine Translation in the Americas*, pp. 223–231.

Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser L. and Polosukhin I. (2017). *Attention Is All You Need*. „CoRR", abs/1706.03762.

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

Wu Y., Schuster M., Chen Z., Le Q. V., Norouzi M., Macherey W., Krikun M., Cao Y., Gao Q., Macherey K. et al. (2016). *Google's Neural Machine Translation System: Bridging the Gap Between Human and Machine Translation*. „CoRR", abs/1609.08144.

Results of the PolEval 2019 Shared Task 5: Automatic Speech Recognition Task

Danijel Koržinek (Polish-Japanese Academy of Information Technology)

Abstract

This article summarizes the first PolEval shared task on Automatic Speech Recognition in Polish. The goal of the task is to build a system for transcribing sessions of the Polish Parliament. The participants were allowed and encouraged to use any method to solve this problem and were provided with all the data necessary to accomplish this task. Because some participants had access to data not available to others, the competition was divided into two sub-tasks: fixed using only the data provided during the competition and open using any data available. The main evaluation metric was Word Error Rate. Four teams participated in the competition and achieved scores ranging from 41.8% to 11.8% WER.

Keywords

Polish, natural language processing, automatic speech recognition, speech recognition evaluation, ASR

1. Introduction

Automatic speech recognition (ASR) is the problem of converting an audio recording of speech into its textual representation. For the purpose of this evaluation campaign, the transcription is considered simply as a sequence of words conveying the contents of the recorded speech. Here, words are defined simply as space delimited tokens. ASR is a very common and well known problem. It has many practical uses in both commercial and non-commercial setting. Applications of ASR include among other things: dictation, human-computer interaction, dialog systems, speech transcription, speech corpus analytics, etc. There are many evaluation campaigns associated specifically with ASR, for example ASpIRE in (Harper 2015), CHiME in (Vincent et al. 2016), NIST Rich Transcription Evaluation in (Fiscus et al. 2006).

A common method of solving ASR is by utilizing the following formula for maximum a posteriori (MAP) optimization:

$$w* = \arg\max_i P(w_i|O) \approx \arg\max_i P(O|w_i) \cdot P(w_i) \quad (1)$$

The formula defines the task of ASR as the most likely sequence of words w_i, given a sequence of acoustic observation O of data. This equation is furthermore broken into two essential components by Bayesian inference: the estimation of the acoustic-phonetic realization $P(O|w_i)$, also known as acoustic modeling (AM), and the probability of word sequence realization $P(w_i)$, also known as language modeling (LM).

Each element of the O sequence is usually defined as a constant size vector of distinctive acoustic features calculated using well known signal processing techniques, for example Short-Term Fourier Transform, Mel-Frequency Cepstral Coefficients, Linear Prediction Coefficients (Young et al. 2002). It is worth noting that the length of the O sequence can be very different from the length of the w sequence and usually it is much longer – most often around 100 elements per second. Historically this problem used to be solved using Dynamic Time Warping (DTW Berndt and Clifford 1994).

A more common framework for solving ASR is the Hidden Markov Model (Young et al. 2002). Currently, this concept was expanded to a more useful implementation based on Weighted Finite-State Transducers (Mohri et al. 2002). Some of the most recent solutions try to bypass the individual sub-steps by modeling the whole process in a single end-to-end model usually based on artificial neural-networks (Graves and Jaitly 2014).

2. Task Description

The goal of the Poleval 2019 ASR competition was to transcribe sessions of the Polish Parliament. This task is very common in many countries due to easy access to the training material – both the audio and the transcripts are public domain and mandated by law. The quality of the recordings is fairly high and predictable with minimal cocktail speech and a single person speaking long passages of text.

The participants were given data collected before January 1st 2019. They were given several weeks notice in order to prepare the systems for the evaluation campaign. The evaluation was performed around one week after publishing the evaluation data, which was collected from a random date after the January 1st cutoff date. The participants were asked to transcribe the data themselves and send the transcriptions of the systems only. The transcription should consist of regular UTF-8 encoded text files, one text file per evaluation audio file. Each file should contain a single line of text, all lower-case, without any digits or special characters.

The competition encourages participation on all levels of expertise, from students to whole organization specializing in the problem, many of which use proprietary systems and are not allowed to disclose the details of their intellectual property. That is why it was decided to provide to ways of participating in the competition. In the fixed competition, the participants can only use the provided data. This would allow a level-playing field, especially for the junior

participants with low resources and capabilities. If a participant wishes to use systems trained on data outside of the provided corpora, they can still participate in the open competition.

3. Data

The organizers provided the following open corpora related to the competition:

— Clarin-PL speech corpus (Koržinek et al. 2017) – `https://mowa.clarin-pl.eu/korpusy`

— PELCRA parliamentary corpus (Pęzik 2018) – `https://tinyurl.com/PELCRA-PARL`

— a collection of 97 hours of parliamentary speeches published on the ClarinPL website (Marasek et al. 2014) – `http://mowa.clarin-pl.eu/korpusy/parlament/parlament.tar.gz`

— Polish Sejm Corpus for language modeling (Ogrodniczuk 2012) – `http://clip.ipipan.waw.pl/PSC`.

For the open competition, the participants were encouraged to use any data available, apart for the data published on the websites of the Polish Parliament after January 1st 2019. This was because this data was reserved for evaluation. The users could still acquire data from other sources from the same time period, e.g. news, forums and other media.

The evaluation data was collected all from a single day, January 31st 2019. It consisted of 29 audio files with a mean duration of ∼99 seconds, minimum duration of ∼5 seconds and maximum duration of ∼362 seconds. The whole collection contained ∼48 minutes of hand-annotated data. Audio was encoded as uncompressed, linearly encoded 16-bit per sample, 16 kHz sampling frequency, mono signals encapsulated in WAV formatted files.

4. Evaluation

The evaluation was performed using the common Word Error Rate (WER) metric. Word error rate is defined as the number of edit-distance errors (deletions, substitutions, insertions) divided by the length of the reference:

$$WER = \frac{N_{del} + N_{sub} + N_{ins}}{N_{ref}} \tag{2}$$

In order to avoid minor differences between various computation methods the evaluation was based on the well-known NIST SCLITE program for evaluating speech recognition results (Fiscus 1998), made available on `https://github.com/usnistgov/SCTK`.

Table 1: Overall results for the Poleval 2019 ASR competition. Some team names were modified to protect the authors' identities.

Team	System	WER%	CORR%	SUB%	DEL%	INS%	Type
MS	GOLEM	**12.8**	90.1	6.9	3.0	2.9	Fixed
ML	ARM-1	**26.4**	77.0	16.5	6.5	3.4	Open
AWSR	SGMM2	**41.3**	65.2	27.1	7.7	6.5	Open
	tri2a	**41.8**	62.9	26.8	10.3	4.7	Open
PJATK	clarin-pl/studio	**30.9**	71.4	16.0	12.6	2.4	Open
	clarin-pl/sejm	**11.8**	89.7	5.4	5.0	1.4	Fixed

Table 2: The per-file statistics for the individual systems

Team	System	Mean	StdDev	Median
MS	GOLEM	13.3	8.8	11.9
ML	ARM-1	27.2	13.5	24.7
AWSR	SGMM2	41.3	18.1	38.8
	tri2a	41.4	16.9	38.5
PJATK	clarin-pl/studio	30.4	13.6	25.9
	clarin-pl/sejm	12.0	7.9	9.8

5. Participating Systems

Three different teams sent their results for evaluation. The organizer of the campaign also provided two sample systems, but didn't participate in the reward portion of the competition.

To the best knowledge of the organizers, all the systems, except the clarin-pl (developed by the organizer) used acoustic models based on Gaussian Mixture Models (GMMs). The clarin-pl system was neural-network based, which gave it a competitive advantage. All the systems apart from ARM-1 were based on the open-source Kaldi project.

6. Conclusion

To the best knowledge of the organizers, this was the first open strictly Polish evaluation campaign in automatic speech recognition. Due to time constraints and overall limited knowledge of the subject in Poland, not too many teams participated in this campaign. The results were still very promising and consistent.

The only two systems that took part in the fixed competition were also the only two that used in-domain data. It is no wonder that these systems achieved the best results. The only reason that the PJATK system performed slightly better is the much more elaborate neural network based acoustic model.

There is still lots of room for improvement. The presented systems use only the most basic speech recognition technology, for example rescoring using Recurrent Neural Network Language Models, better adaptation techniques, transfer learning from larger acoustic corpora and open vocabulary are just some of the methods that would be worthwhile to verify. The organizers hope that this competition will serve as an inspiration for further research on this topic.

Acknowledgments

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

References

Berndt D. J. and Clifford J. (1994). *Using Dynamic Time Warping to Find Patterns in Time Series*. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, pp. 359–370. Seattle, WA.

Fiscus J. (1998). *Sclite Scoring Package version 1.5*. `http://www.itl.nist.gov/iaui/894.01/tools`.

Fiscus J. G., Ajot J., Michel M. and Garofolo J. S. (2006). *The Rich Transcription 2006 Spring Meeting Recognition Evaluation*. In *International Workshop on Machine Learning for Multimodal Interaction*, pp. 309–322. Springer.

Graves A. and Jaitly N. (2014). *Towards End-to-end Speech Recognition with Recurrent Neural Networks*. In *Proceedings of the 31st International Conference on International Conference on Machine Learning – Volume 32*, pp. 1764–1772.

Harper M. (2015). *The Automatic Speech Recognition in Reverberant Environments (ASpIRE) challenge*. In *Proceedings of the 2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pp. 547–554. IEEE.

Koržinek D., Marasek K., Brocki Ł. and Wołk K. (2017). *Polish Read Speech Corpus for Speech Tools and Services*. „CoRR", abs/1706.00245.

Marasek K., Koržinek D. and Brocki Ł. (2014). *System for Automatic Transcription of Sessions of the Polish Senate*. „Archives of Acoustics", 39(4), p. 501–509.

Mohri M., Pereira F. and Riley M. (2002). *Weighted Finite-State Transducers in Speech Recognition*. „Computer Speech & Language", 16(1), p. 69–88.

Ogrodniczuk M. (2012). *The Polish Sejm Corpus*. In Calzolari N., Choukri K., Declerck T., Doğan 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. 2219–2223, Istanbul, Turkey. European Language Resource Association.

Pęzik P. (2018). *Increasing the Accessibility of Time-Aligned Speech Corpora with Spokes Mix*. In Calzolari N., Choukri K., Cieri C., Declerck T., Hasida K., Isahara H., Maegaard B., Mariani J., Moreno A., Odijk J., Piperidis S., Tokunaga T., Goggi S. and Mazo H. (eds.), *Proceedings of the 11th International Conference on Language Resources and Evaluation (LREC 2018)*, pp. 4297–4300, Miyazaki, Japan. European Languages Resources Association.

Vincent E., Watanabe S., Barker J. and Marxer R. (2016). *The 4th CHiME Speech Separation and Recognition Challenge*. `http://spandh.dcs.shef.ac.uk/chime_challenge/` (last accessed on August 1, 2018).

Young S., Evermann G., Gales M., Hain T., Kershaw D., Liu X., Moore G., Odell J., Ollason D., Povey D. et al. (2002). *The HTK Book*, vol. 3. Cambridge University Press.

ARM-1: Automatic Speech Recognition Engine

Jerzy Jamroży, Marek Lange, Mariusz Owsianny, Marcin Szymański[1]
(Poznań Supercomputing and Networking Center)

Abstract

The paper presents ARM-1 engine – a Large Vocabulary Continuous Speech Recognition system for Polish. It is a HMM/GMM based system with unsupervised speaker adaptation. ARM-1 competed in PolEval 2019 Automatic speech recognition task in open competition category.

Keywords

natural language processing, speech recognition, ASR, LVSCR

1. Introduction

First attempts to automatically recognize speech were made in the 1950s. Since then there was a continuous progress towards achieving higher accuracy of that process. One of the milestones in the work on ASR (Automatic Speech Recognition) was the use of statistical modeling with Hidden Markov Models (HMM). In this approach machine learning algorithms are used to build models based on training data in order to process and identify speech signal. HMMs allow one to combine various types of information such as acoustics, language and syntax in one probabilistic system. This technique dominated till late 1990s, when HMMs were combined with artificial neural networks. Gradually deep learning methods are taking over various aspects of speech recognition process.

With the success of the ASR techniques, the application areas were growing, as were the users' expectations. Gradually the goals of ASR were changing from merely understanding an utterance through processing spontaneous speech recorded in a noisy environment with many speakers' voices overlapping to transcribing audio signal which is inaudible for a human. In other words, the expectation are that ASR systems should be able to understand speech not just as well as a human can but better than that. Such systems are needed especially wherever the quality or the volume of data makes it impossible for a human operator to process the data.

[1] Currently at Samsung Electronics. The work presented in the paper was done while the author was at PSNC.

Speech recognition for some languages are more challenging than for others. Polish, being inflectional language with a flexible word order, poses a bigger challenge than English (Nouza et al. 2010). For such language certain assumptions concerning the acoustic-phonetic database structure need to be modified in order to provide adequate material for both acoustic and language modeling (Demenko et al. 2012). In this paper we present ARM-1[2], a Large-Vocabulary Continuous Speech Recognition system for Polish. ARM-1 is based on HMM and statistical acoustic and linguistic modeling. It implements an ASR workflow that includes, beside the standard blocks, also unsupervised speaker adaption modules that can significantly reduce Word Error Rate (WER). ARM-1 engine has a number of application. In most cases the engine is used to process data in large repositories whose size makes manual processing impossible.

ARM-1 has been submitted to compete in open competition subtask of PolEval Task 5, i.e., training data set used in acoustic and linguistic modeling was not specified. The engine, the principles of its operation and modifications of the standard HMM based workflow are described in Section 2. The training data used to generate acoustic and linguistic models used to perform the competition task, are characterized in Section 3 followed by recognition results presentation in Section 4. We conclude the description in Section 5.

2. ARM-1 Engine

ARM engine has been developed since 2007. Research that laid foundation for ARM has been initiated by prof. Grocholewski and prof. Demenko. Initially work which concentrated on acoustic-phonetic database structure for acoustic and language modeling, resulted in JURISDICT (Demenko et al. 2008, Szymański et al. 2008) corpus for training and testing ASR systems, specifically in judiciary domain. JURISDICT consists of 2000 recordings from all over the country, i.e., recordings of speakers from various regions, with a total duration of over 1200 hours. Poznan Supercomputing and Networking Center team joined research and development activities in 2010. Since then the system is under continuous development with an objective to improve the results and adjust the solution to various application areas. ARM, and it subsequent version ARM-1, have been used to support document generation in judiciary and medical domains, to monitor media, perform spoken document retrieval and support IVR (Interactive Voice Response) systems. Such a wide range of application required diversified resources for acoustic and linguistic modeling. Consequently, ARM-1 engine can be equipped with universal models or models dedicated to a specific domain.

2.1. ARM-1 Principles

ARM-1 implements HMM-GMM (Gaussian Mixture Model) speech recognition methodology (El-emary et al. 2011). Speech recognition process is a sequence of the following three phases: digital signal processing (DSP), decoding and rescoring as shown in Figure 1.

[2] `https://speechlabs.pl/en/our-offer/arm/`

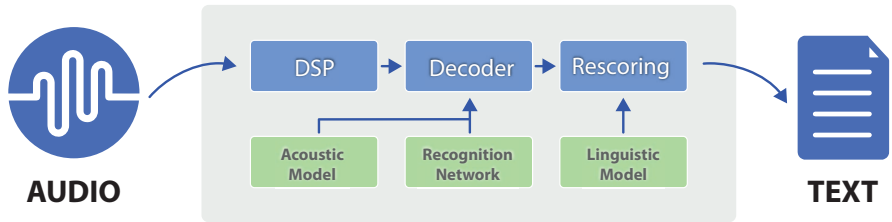

Figure 1: ARM-1 workflow

DSP is responsible for capturing audio signal from an input device or from a file and processing that signal. This stage may encompass normalization, frequency filtering and Voice Activity Detection (VAD). The main phase however, is signal parametrization which is done by dividing data into a number of observations and determining a set of values characterizing each observation. Each observation is represented by so-called feature vector. Observations that are classified by VAD as voice data undergo decoding.

Decoder processes feature vectors for each observation in order to recognize words that constitute a given utterance. This goal is achieved by searching for the best path matching a sequence of observations in a recognition network. At a high level, the recognition network consists of words pronunciations defined as sequences of triphones. Each triphone is modelled with a HMM, which in turn consists of a set of states. Each of the emitting states is described with sets (mixtures) of Gaussian probability densities. The densities are determined for each observation feature and assigned to a given state during model training. During decoding an observation is evaluated with the GMM in order to obtain acoustic likelihood used as a cost for best path finding algorithm. The result consists of a set of hypotheses, each of which matches the input data with a certain probability.

Rescorer evaluates each of the hypotheses by determining combined acoustic and linguistic probability, in order to find the best hypothesis. More specifically, a hypothesis is evaluated and ranked based on the product of weighted average of acoustic and linguistic probability of all its words. Linguistic model allows one to determine probability of a word occurring in a certain context.

2.2. Speaker Adaptation

Correctness of speech recognition results depends to a large degree on how well models used in the process represent observations. Acoustic and linguistic models are statistical models generated based on training data, (acoustic and text corpora) with machine learning algorithms. If there is mismatch between training and test data, the ASR performance is degraded. In other words, utterance spoken in a very specific and unusual way, will not be recognized correctly. The mismatch can be addressed with adaptation to a speaker, i.e., adjusting models so that they better represent previously unseen observations.

Adaptation can be performed in a supervised manner using annotated training data. Our tests show that it can improve correctness by as much as a dozen percent, depending on how well the original acoustic model represents the speaker. Usually that gain is bigger for speakers for whom speech recognitions results are poor. However, supervised adaptation is time-consuming and must be conducted in an off-line manner. Moreover, such an approach cannot be applied to recordings of more than one speaker. These problems can be addressed with unsupervised adaption. Adaptation of this type is based on segmentation provided by Decoder. Obviously, the segmentation is not perfect, but this problem can be mitigated by applying a quality filter at the word-end or sentence level. In order to address multiple speakers in a single recording challenge, we have introduced a mechanism differentiating speakers so that adaptation can be performed for each speaker separately. In addition to speaker identification, it is possible to predict by how much the new acoustic model can improve the result based on observations variance at HMM states. Having speaker utterance recognition along with the adapted models and the improvement prediction, Redecoder can perform optional decoding of the given utterance with the adapted models or present results from the first Decoder pass as the final output data.

2.3. Implementation

ARM-1 engine was developed with Microsoft .NET Framework 4.0 platform with the intense use of Task Parallel Library (TPL). The engine can work in an offline mode with speech signal obtained from an audio file, or in an on-line mode with speech signal obtained directly from an audio device.

DSP module analyzes audio data, and performs Voice Activity Detection. The input signal is divided into separate observations (25 ms window with 10 ms stepping). For each observation LDA (Linear Discriminant Analysis) transformation is computed over Mel Frequency Cepstral Coefficients (MFCC) and Filterbank parameters, giving a short feature vector as a result. The observation vectors are passed to a recognition based decoder. The decoder is built upon modified beam-search algorithm and works over a recognition network being a word-loop of approximately 280 thousand dictionary entries with imposed unigram probabilities. The decoder produces a hypotheses lattice as a result. The lattice elements are attributed with appropriate probabilities. All hypotheses are evaluated using N-Grams linguistic model in the Rescoring module. Hypothesis with best probability is returned as a recognized text. Postprocessing is performed on the this text in order to apply rules for transformation of the result into the final form, i.e., insert punctuation or abbreviations.

In order to use acoustic adaptation in on-line (live stream) scenarios, the ARM-1 pipeline was equipped with additional blocks: Speaker Tracker, Adaptation and Redecoder as shown in Figure 2. The Speaker Tracker block is responsible for collecting statistical data for each speaker identified in the input signal. This data is used by the Adaptation block to adapt the acoustic model to current speaker and to estimate possible improvement resulting from model adaptation. The Redecoder performs optional secondary decoding of the input data whenever the estimated improvement exceeds a certain threshold.

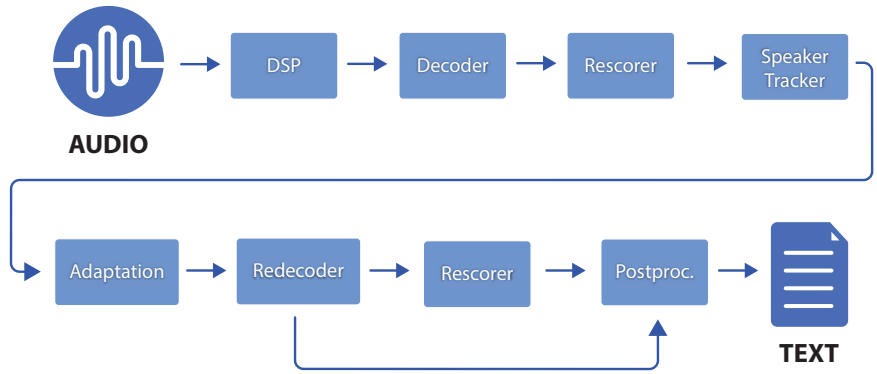

Figure 2: ARM-1 workflow with speaker adaptation

The Speaker Tracker module operates on the parametrized audio samples and on the hypothesis produced by the first decoding phase. Since each speech segment identified by VAD may, and often does, contain voices of more than one speaker, it is Speaker Tracker responsibility to detect speaker change and gather corresponding statistical data. Speaker change detection relies on the comparison between GMM components and audio parameters corresponding to HMM states.

The Adaptation module operates on statistical audio parameters data collected for a given speaker and calculates a set of transformations for GMM components using MLLR (Maximum Likelihood Linear Regression). This step reduces the mismatch between observation features and GMM components. Because the amount of adaptation data is very limited in this case, regression class tree, which groups GMM components, is used (Young et al. 2002).

The Redecoder module performs secondary decoding whenever estimated improvement exceeds certain threshold. The secondary decoding is performed in the same way as the primary decoding by the Decoder module but with the adapted models.

3. Training Data

ARM-1 engine was submitted to PolEval 2019 Task 5 in an open system category, which means that the training set could include any data with the exception of this year speech recordings from the lower and higher house of the Polish Parliament. Given various domains of ARM-1 engine applications and the fact we have data from various source acquired over time at our disposal, we have decided to test the solution with universal models, i.e., models that are not dedicated to a specific domain but rather represent various speech styles and acoustic characteristics.

3.1. Acoustic Modeling

The training acoustic set consisted of approximately 457 hours of recordings of various types as far as acoustic characteristics and speech styles are concerned. The main components of this set are as follows:

— 205 hours of dictated speech

— 58 hours parliamentary addresses

— 20 hours of radio programming

— 40 hours of TV programming

— 40 hours of phone conversations.

The above-listed sets were complemented by a smaller amount of recordings of court hearings and various meetings. All recordings are monophonic 16kHz with PCM audio signal representation.

The training set recordings were segmented at the phone-level. The phonetic dictionary consisted of 48 elements including stressed as well as unstressed vowels, element representing silence pause (sil), filled pauses, i.e. pauses filled by a speaker with non-verbal speech such as “um”, “ehh” (speaker fill), and noises such as grunting (speaker noise).

Since a large part of the training data contains dictated speech which is usually characterized by low tempo, a procedure for increasing tempo was used. An average speech tempo was determined for the entire training set and expressed as an average number of phonemes produced within a given amount of time. Recordings with speech tempo below 66th percentile of that average were sped up 1.2 times. The change of tempo affected the average fundamental voice frequency.

The acoustic speech model was trained with HTK (Young et al. 2002) tool set, namely HInit, HRest, HERest and HHEd. A standard training procedure for triphone Continuous Density Hidden Markov Model was used. A list of approximately 60 contextual questions formulated on the basis of phoneme articulation features (mostly articulation manner and placement) were used for state and triphone clustering.

3.2. Language Modeling

Text corpus used for language modelling contained approximately 20 GB of data. The corpus consisted of electronic editions of a number of newspapers (such as Wprost, Newsweek, Gazeta Wyborcza, Rzeczpospolita, Puls Biznesu), parliamentary speech transcripts and Wikipedia articles. The preprocessing besides standard partitioning into sentences and capitalization of all letters, included also number and date conversion into word representation and abbreviation expansion.

A statistical 3rd degree linguistic model was built with SRILM (Stolcke 2002) package using Witten-bell discounting and Katz back-off for smoothing. The cutoff frequency for bigrams was 2 and for trigrams was 4. The final model included of 283,262 unigrams, 74,054,429 bigrams and 60,918,030 trigrams.

4. Test Results

Performance of ASR systems submitted to the competition was evaluated and compared based on the results obtained for a set of test recordings of parliamentary speech. Most of them were interpellations or answers delivered in a more or less formal way. The test set consisted of 29 files with a total duration of nearly 48 minutes. The shortest recording was 5 second long and the longest over 6 minutes. There were PCM files recorded at 256 kb/s with the sampling rate of 16 kHz. There were some variations in the volume of the recordings. Figure 3 presents average volume for each recording. There was usually just one speaker recorded per file with the exception of short interruptions by the House Speaker leading the debate.

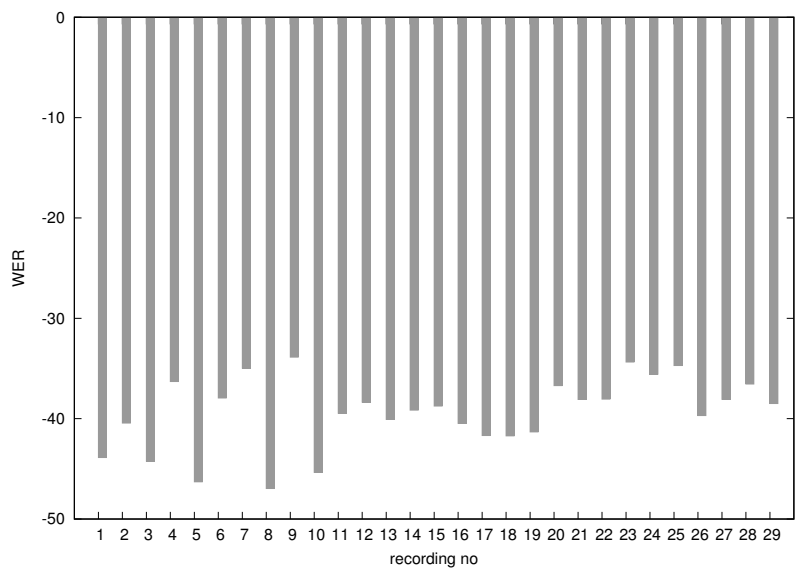

Figure 3: Average recording volume

The speech recognition results were normalized, i.e., number and dates were written in words with no abbreviations and no punctuation. The reference text for each recording was not provided to the competitors. Hence, the results presented in this section were obtained with references generated for each file and therefore, there may be some slight differences with the official competition results. The differences may occur in places where an utterance was inaudible due to overlap of two voices for example. The basic evaluation metric was Word Error Rate (WER) defined as the number of edit-distance errors (deletions, substitutions, insertions) divided by the length of the reference:

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

Figure 4 presents WER value obtained for each test file with speaker adaptation. The total WER for the entire set was slightly above 26%. Table 1 presents detailed results for each recording and the total for the entire set.

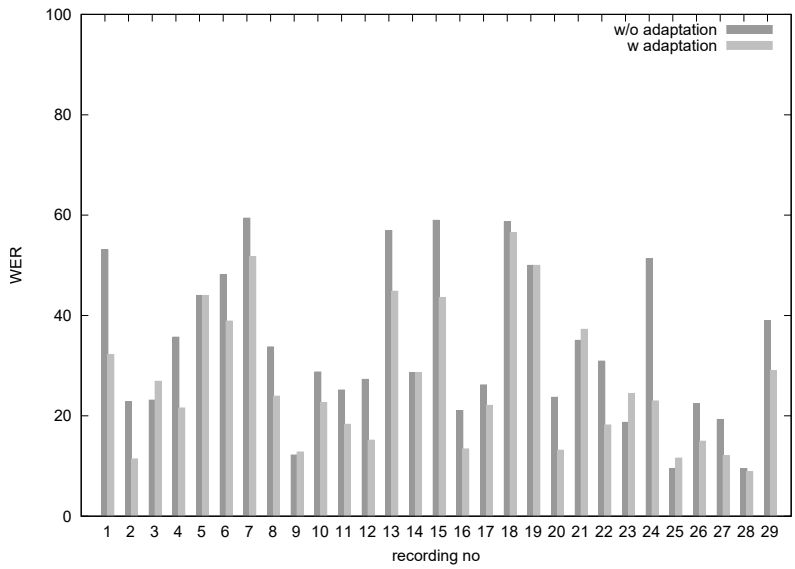

Figure 4: Test data recognition results

In order to assess the influence of the speaker adaption on WER, we have also performed tests on the same set without speaker adaptation. The total WER in that case was 34,49%. Figure 4 presents comparison between WER obtained with and without speaker adaptation for each of the files and for the entire set. For the majority of files speaker adaptation resulted in a reduction of WER.

5. Conclusion

We have presented ARM-1 engine developed for continuous speech recognition for Polish and the results obtained for PolEval Task 5 test data. ARM-1 is a HMM/GMM based recognition system with a speaker adaptation module. WER obtained for the test data is not low enough to use speech recognition results as a transcript of the recordings. There are however, areas of application where speech recognition with WER at this level can still be used. Nonetheless, the system is still under development with an objective to improve the accuracy.

Table 1: Detailed tests results

Label	Corr [%]	Ins [%]	Del [%]	Sub [%]	WER [%]
224	71.43	3.57	5.80	22.77	32.14
35	88.57	0.00	0.00	11.43	11.43
160	76.88	3.75	4.38	18.75	26.88
807	80.05	1.61	6.20	13.75	21.56
25	56.00	0.00	4.00	40.00	44.00
432	63.66	2.55	13.89	22.45	38.89
261	77.39	29.12	5.75	16.86	51.72
255	81.96	5.88	4.31	13.73	23.92
164	89.63	2.44	3.05	7.32	12.80
362	83.43	6.08	4.14	12.43	22.65
191	82.72	1.05	4.19	13.09	18.32
99	84.85	0.00	4.04	11.11	15.15
58	58.62	3.45	6.90	34.48	44.83
98	72.45	1.02	13.27	14.29	28.57
195	57.95	1.54	13.33	28.72	43.59
209	89.47	2.87	1.91	8.61	13.40
172	80.23	2.33	4.65	15.12	22.09
92	44.57	1.09	16.30	39.13	56.52
14	50.00	0.00	21.43	28.57	50.00
38	92.11	5.26	5.26	2.63	13.16
137	67.88	5.11	8.76	23.36	37.23
110	84.55	2.73	1.82	13.64	18.18
139	81.29	5.76	3.60	15.11	24.46
148	81.76	4.73	3.38	14.86	22.97
147	92.52	4.08	2.72	4.76	11.56
174	89.66	4.60	2.87	7.47	14.94
166	90.36	2.41	2.41	7.23	12.05
180	93.33	2.22	1.67	5.00	8.89
541	73.38	2.40	7.76	18.85	29.02
5633	78.18	4.19	6.14	15.68	26.01

Acknowledgments

This work was partially supported by the Polish Center for Research and Development, project no. POIR.01.01.01-00-1532/15-00.

References

Demenko G., Grocholewski S., Klessa K., Ogórkiewicz J., Wagner A., Lange M., Śledzinski D. and Cylwik N. (2008). *LVCSR Speech Database — JURISDIC*. In *New Trends in Audio and Video/Signal Processing Algorithms, Architectures, Arrangements, and Applications SPA*, pp. 67–72.

Demenko G., Szymański M., Cecko R., Kuśmierek E., Lange M., Wegner K., Klessa K. and Owsianny M. (2012). *Development of Large Vocabulary Continuous Speech Recognition for Polish*. „Acta Physica Polonica A", 121(1-A), p. 86–91.

El-emary I. M. M., Fezari M. and Attoui H. (2011). *Hidden Markov Model/Gaussian Mixture Models (HMM/GMM)-based Voice Command System: A Way to Improve the Control of Remotely Operated Robot Arm TR45*. „Scientific Research and Essays", 6, p. 341–350.

Nouza J., Zdansky J., Cerva P. and Silovsky J. (2010). *Challenges in Speech Processing of Slavic Languages (Case Studies in Speech Recognition of Czech and Slovak)*, pp. 225–241. Springer Berlin Heidelberg, Berlin, Heidelberg.

Stolcke A. (2002). *SRILM – An Extensible Language Modeling Toolkit*. In *INTERSPEECH*. ISCA.

Szymański M., Ogórkiewicz J., Lange M., Klessa K., Grocholewski S. and Demenko G. (2008). *First evaluation of Polish LVCSR acoustic models obtained from the JURISDIC database*. In *Speech and Language Technology*, pp. 39–46.

Young S., Evermann G., Kershaw D., Moore G., Odell J., Ollason D., Valtchev V. and Woodland P. (2002). *The HTK book*. „Cambridge University Engineering Department", 3.

Results of the PolEval 2019 Shared Task 6: First Dataset and Open Shared Task for Automatic Cyberbullying Detection in Polish Twitter

Michal Ptaszynski (Kitami Institute of Technology),
Agata Pieciukiewicz (Polish-Japanese Academy of Information Technology),
Paweł Dybała (Jagiellonian University in Kraków)

Abstract

In this paper we describe the first dataset for the Polish language containing annotations of harmful and toxic language. The dataset was created to study harmful Internet phenomena such as cyberbullying and hate speech, which recently dramatically gain on numbers in Polish Internet as well as worldwide. The dataset was automatically collected from Polish Twitter accounts and annotated by both layperson volunteers under the supervision of a cyberbullying and hate-speech expert. Together with the dataset we propose the first open shared task for Polish to utilize the dataset in classification of such harmful phenomena. In particular, we propose two subtasks: 1) binary classification of harmful and non-harmful tweets, and 2) multiclass classification between two types of harmful information (cyberbullying and hate-speech), and other. The first installment of the shared task became a success by reaching fourteen overall submissions, hence proving a high demand for research applying such data.

Keywords

cyberbullying, automatic cyberbullying detection, hate-speech, natural language processing, machine learning

1. Introduction

Although the problem of humiliating and slandering people with the use of Internet communication measures has existed almost as long as the communication via the Internet between people itself, the appearance of new handheld mobile devices, such as smartphones and tablet computers, which allow using the Internet not only at home, work or school but also in commute, has further intensified the problem. Especially recent decade, during which

Social Networking Services (SNS) such as Facebook and Twitter, rapidly grew in popularity, has brought to light the problem of unethical behaviors in Internet environments, which has been greatly impairing public mental health in adults and, for the most, in younger users and children. It is the problem of *cyberbullying* (CB), defined as exploitation of open online means of communication, such as Internet forum boards, or SNS to convey harmful and disturbing information about private individuals, often children and students.

To deal with the problem, researchers around the world have begun to study the problem with a goal of automatic detection of Internet entries containing harmful information and reporting them to SNS service providers for further analysis and deletion. After ten years of research (Ptaszynski et al. 2010b,a, Nitta et al. 2013a,b, Hatakeyama et al. 2015, Ptaszynski et al. 2015, Lempa et al. 2015, Hatakeyama et al. 2016a, Ptaszynski et al. 2016a, Hatakeyama et al. 2016b, Ptaszynski et al. 2016b, 2017, 2018, Ptaszynski and Masui 2018, Ptaszynski et al. 2019), a sufficient knowledge base on this problem has been collected for languages of well-developed countries, such as the US, or Japan. Unfortunately, still close to nothing in this matter has been done for the Polish language. With the presented here dataset and the initial experiments performed with the dataset, we aim at filling this gap.

The dataset, as well as open shared task supplementing the dataset, allows the users to try their classification methods to determine whether an Internet entry is classifiable as part of cyberbullying narration or not. The entries contain tweets collected from openly available Twitter discussions. Since much of the problem of automatic cyberbullying detection often relies on feature selection and feature engineering (Ptaszynski et al. 2017, 2019), the tweets are be provided as such, with minimal preprocessing. The preprocessing, if used, is applied mostly for cases when information about a private person is revealed to the public.

The goal of the main task is to classify the tweets into cyberbullying/harmful and non-cyberbullying/non-harmful with the highest possible Precision, Recall, balanced F-score and Accuracy. In an additional subtask, the goal is to differentiate between various types of harmful information, in particular cyberbullying and hate-speech, and non-harmful[1].

The rest of the paper is organized in the following way. In Section 2 we describe how the data for the dataset was collected. In Section 3 we explain the whole annotation process, including our working definition of cyberbullying and guidelines for annotation used in training the annotators. In Section 4 we perform an in-depth analysis of the created dataset, which includes both general statistical analysis as well as deeper example-based specific analysis. In Section 5 we describe the task we propose together with the dataset, in particular two subtasks for classification of 1) harmful information in general and 2) two specific types of harmful information. We also propose the default means for evaluation and introduce the participants that took part in the first installment of the shared task. In Section 4 we present the results of the participants in comparison to a number of baselines. Finally, in Sections 7 and 8 we conclude the paper and set up plans and directions for the near future.

[1]The dataset, together with the two subtasks proposed for it, is available under the following URL: `https://github.com/ptaszynski/cyberbullying-Polish`

2. Data Collection and Preprocessing

2.1. Collection

In order to collect the data, we used Standard Twitter API[2]. It has a number of limitations, which we had to work around. For example, the number of requests per 15-minute window and the number of tweets that could be downloaded in one request is limited by Twitter API. We respected those limits, and after exhausting the limit of requests the download script simply waited for another download window. Twitter API was used via the python-twitter library (`https://github.com/bear/python-twitter/`). Another obstacle was the time limit for searching tweets. In Standard (non-paid) Twitter API the user is allowed to search for tweets from past 7 days. That is why we were not able to collect all answers to tweets made from our initial starting accounts. Our script saved data received from Twitter in MongoDB using the pymongo library (`https://github.com/mongodb/mongo-python-driver`). Twitter provides tweet data in JSON format, so the use of a document database was convenient for further handling of data.

The script, written in Python, has been used to download tweets from nineteen official Polish Twitter accounts. Those accounts were chosen as the most popular Polish Twitter accounts in the year 2017[3]. By popular we understand those with the largest number of observers, those with a rapidly growing number of observers, those who collected the most user activity, those most often mentioned and those who themselves tweeted most often. In particular, we initially looked at the following accounts: `@tvn24`, `@MTVPolska`, `@lewy_official`, `@sikorskiradek`, `@Pontifex_pl`, `@donaldtusk`, `@BoniekZibi`, `@NewsweekPolska`, `@PR24_pl`, `@tvp_info`, `@rzeczpospolita`, `@AndrzejDuda`, `@lis_tomasz`, `@K_Stanowski`, `@R_A_Ziemkiewicz`, `@pisorgpl`, `@Platforma_org`, `@RadioMaryja`, `@RyszardPetru`.

In addition to tweets from those accounts, we have collected answers to any tweets from the accounts mentioned above (from the past 7 days). In total, we have received over 101 thousand tweets from 22,687 accounts (as identified by screen_name property in the Twitter API). Using bash random functions ten accounts were randomly selected to become the starting point for further work.

Next, using the same script as before, we downloaded tweets from these 10 accounts and all answers to their tweets that we were able to find using the Twitter Search API (again, limited to the past 7 days). Using this procedure we have selected 23,223 tweets from Polish accounts for further analysis. Data downloading was finished on 20.11.2018. (Last downloaded tweet was created at 18:12:32). These 23,223 tweets became the base for the dataset presented in this paper.

[2] `https://developer.twitter.com/en/docs/tweets/search/api-reference/get-search-tweets.html`

[3] According to `https://www.sotrender.com/blog/pl/2018/01/twitter-w-polsce-2017-infografika/`

2.2. Preprocessing and Filtering

Since in this initial dataset, we did not follow the conversation threads (as the official Twitter API does not provide such information), we considered each tweet separately.

At first, we randomized the order of tweets in the dataset to get rid of any consecutive tweets from the same account. This would help decrease the anchoring bias (Tversky and Kahneman 1974) in annotations since when a human annotator reads tweets from the same account they could become prone to assigning the same score to many messages.

Next, we got rid of all tweets containing URLs. This was done due to the fact that URLs often take space and limit the contents of the tweets, which in practice often resulted in tweets being cut in the middle of the sentence or with a large number of *ad hoc* abbreviations. Next, we got rid of all tweets which were exactly the same in contents, which eliminated most of the duplications. Tweets consisting only of at-marks (@) or hashtags (#) were also removed, as they do not convey any intrinsic linguistic value as a whole, but rather are used as unrelated keywords. Finally, we removed tweets with less than five words and those written in languages other than Polish. This left us with 11,041 tweets. From this group, we randomly extracted 1,000 tweets to be used as test data and the rest (10,041) was used as training data. The exact step-by-step preprocessing procedure and analysis of how many tweets were discarded at each time is presented below.

1. Deleted tweets with URLs and retaining only the text of tweets (no meta-data, timestamps, etc.) (retained: 15,357/23,223, or 66.13% of all).
2. Deleted exact duplicates (retained:15,255, only 102 deleted, or 0.44% of all).
3. Deleted tweets containing only @atmarks and #hashtags (retained: 15,223, only 32 deleted, or 0.14% of all).
4. Deleted tweets that, except @atmarks or #hashtags consist of only a single or a few words or emoji, etc.:
   (a) Deleted tweets with only one word (retained: 14,492 tweets, 731 deleted, or 3.1% of all).
   (b) Deleted tweets with only two words (retained: 13,238 tweets, 1254 deleted, or 5.4% of all).
   (c) Deleted tweets with only three words (retained: 12,226 tweets, 1012 deleted, or 4.4% of all).
   (d) Deleted tweets with only four words (retained: 11,135 tweets, 1091 deleted, or 4.7% of all).

After the above operations, we were left with 11,135 tweets containing five or more words, not counting @atmarks or #hashtags.

The reasoning behind deleting short tweets was the following:

1. For a human annotator a tweet that is too short will contain an insufficient amount of context, and thus will be difficult to appraise, thus creating many ambiguous annotations.

2. It is also better for machine learning models to have more contents (features) to train on, which also suggests longer sentences will help training more accurate machine learning models. Although one can imagine short tweets also containing aggression, we can assume that if a system is trained on a larger data it will also be able to cover shorter tweets.

In the remaining 11,135 tweets we also noticed a few samples written in a language other than Polish, mostly in English. To solve this problem we used a Text::Guess::Language Perl module[4], which detects the language of a sentence based on top 1000 words from that language. Initial manual analysis of a small sample of tweets revealed that the module sometimes erroneously guessed tweets written in Polish as written in Slovak or Hungarian, due to strangely sounding account names (@atmarks) and #hashtags sometimes used in tweets, but was never wrong when detecting tweets written in English. Therefore as a rule of thumb, we discarded only all English tweets, which in practice left us with only tweets written in Polish. After this final preprocessing operation we were left with 11,041 tweets, from which we used as training data 10,041 tweets and as 1000 tweets as test data.

Together with the dataset we also released a short Perl script used to discard tweets in English from Polish data (onlypolish.pl), as well as tweets that contain only @atmarks or #hashtags (extractnohashatmarks.pl).

3. Annotation Schema

3.1. Cyberbullying – a Working Definition

To develop the annotation schema for annotating the downloaded tweets we firstly prepared our working definition of cyberbullying. Although there is a number of general definitions of the problem, most definitions (Ptaszynski and Masui 2018) agree that

> *cyberbullying happens when modern technology, including hardware, such as desktop or tablet computers, or, more recently, smartphones, in combination with software, such as Social Networking Services (later: SNS, e.g., Twitter, Facebook, Instagram, etc.), is used in a repeated, hostile and, in many times, deliberate attempt to embarrass or shame a private person by sending messages, consisting of text or images, with contents that is malicious and harmful for the victim, such as, shaming the person's appearance or body posture, or revealing the person's private information (address, phone number, photos, etc.)*

Also, social science studies (Dooley et al. 2009) agree that there are both similarities between cyberbullying and traditional face-to-face bullying, as well as differences, which make cyberbullying a problem more difficult to mitigate. Similarities, which make the problem classifiable as a kind of bullying, include: peer group, such as classmates in face-to-face bullying and friends from groups on SNS, which in reality also often overlap; repetitiveness of bullying

[4] `https://metacpan.org/pod/Text::Guess::Language`

acts, which especially on the Internet, could occur more often than in face-to-face bullying; imbalance of power, where one person or a small group becomes bullied by an overwhelming number of bullies and their supporters.

It would be ideal to be able to analyze the data in a wider context, such as threads of conversations on Twitter. Unfortunately, Twitter API does not allow for grouping of conversations, thus in this dataset, we consider each tweet separately. This approach is also similar to all of the previous studies, where each Internet entry was considered as a separate example (Ptaszynski and Masui 2018). In future, however, it is desirable to find a way to automatically group the tweets into conversations to be able to annotate roles of participants in cyberbullying, such as a victim, bully, or bystanders (supporters, defenders).

3.2. Annotation Guidelines

To help annotators perform their task efficiently and to limit the subjective bias of each annotator, we prepared the guidelines for annotations of tweets for harmful information. The guidelines include the following:

English version		**Polish version**
phishing, disclosure or threat of disclosure of private information (phone number, e-mail, address, account name, school name/number, class at school, private identification number (PESEL), credit card number, etc.)	—	wyłudzanie, ujawnienie lub groźba ujawnienia prywatnych informacji (numer tel., e-mail, adres, nazwa konta, nazwa/numer szkoły, klasy, PESEL, karta kredytowa, itd.)
personal attack (“Kill yourself, bitch!”, etc.)	—	atak personalny (“Powieś się, gnoju!”, etc.)
threats (“I will find you and I will kill you”, etc.)	—	groźby (“znajdę cię i zajebię”, etc.)
blackmail (“I will tell everyone where you live if you do not pay me”, etc.)	—	szantaże (“powiem wszystkim gdzie mieszkasz, jeśli mi nie zapłacisz”, etc.)
mocking/ridiculing (“Look how fat this guy is”, “you pimple-face”, etc.)	—	szyderstwa/wyśmiewanie (“Patrzcie na tego grubasa”, “ty pryszczata mordo”, etc.)
gossip/insinuations (“Hey, apparently he’s a zoophiliac!”, etc.)	—	plotki/insynuacje (“Ej, podobno to zoofil!”, etc.)
the accumulation of profanity (single profane and vulgar words appear in conversations fairly often, but a longer “bundle” can be considered as harmful)	—	nagromadzenie wulgaryzmów (pojedyncze występują dość często, ale ich nagromadzenie może być potraktowane jako niepożądane)
various combinations of all of the above	—	różne kombinacje wszystkich powyższych

The scope of the collection of tweets

Cyberbullying is usually addressed at private individuals, thus for the dataset, we used only tweets from private Twitter accounts. We did not include tweets from public accounts (politicians, celebrities) since these are usually from the definition exposed to criticism and personal attacks due to their profession, and often provoke themselves such criticism to raise their popularity. There is no doubt that a public person might also feel privately offended, but even in such case, public persons have the means to deal with such a problem (e.g., employees who massively report abuses in the Twitter system, exert pressure in a number of different ways, even sue an aggressive user).

Harmful, but not cyberbullying

Despite limiting the scope of search to private accounts, there is always a possibility that a harmful tweet addressed at a public person will appear in such collection, Therefore, we decided to also annotate all tweets that do not represent cyberbullying, but are harmful in any other way, e.g., represent hate speech, racism, sexism, but are not addressed at a private person, or a specific small group (e.g. not "you" or "a few people from the class"), but rather a public person, or a specific community in general (e.g., "gays and lesbians", or "Paki" (Pakistanians)/"ciapaty" in Polish).

3.3. Annotation Process

Annotators were provided with only the contents of the tweets and performed annotation one tweet at a time. Each tweet was annotated by at least two, at most three layperson annotators and one expert annotator. Layperson annotators were trained for cyberbullying and hate-speech detection with the guidelines described in this section. Layperson annotators were a group of seven people, all female, in their early twenties. The one expert annotator was a male in his late thirties with a 10-year experience in research on cyberbullying and cyberbullying detection.

After layperson annotators performed their annotations, the expert annotator looked through all annotations and either approved or corrected them. The annotations consisted of the following type of information:

A) harmfulness score:

Score	Label type
0	non-harmful
1	cyberbullying
2	hate-speech and other harmful contents

B) specific tag if possible to specify (see next page)

C) specific phrases if possible to specify in the text.

Abbreviation	Full description	Explanation
pry	prywatne	disclosure or threat of disclosure of private information, phishing
atk	atak	personal attack
gro	groźba	threat
sza	szantaż	blackmail
szy	szyderstwo	mocking/ridiculing
plo	plotka	gossip/insinuations
wul	wulgaryzmy	accumulation of profanity and vulgarities
szy, wul, pry	(etc.)	various combinations of the above

3.4. Examples of Tweets with Annotations

In Table 1 we show a number of examples. Since the dataset contained tweets from various private sources, the annotators were trained to annotate the tweets regardless of their political sentiments. Thus one can see tweets with assigned harmfulness score for both anti-alt-right (Example 2, 4, 6) and anti-left (Example 5), as well as of unknown addressee (Example 1). Some tweets contained typos (Example 5, "endekdu" instead of "endeku" – from "National Democracy supporter"; Example 10 "czulem" instead of "czułem", "głow" instead of "głowa"). Some tweets, which, although contained vulgar vocabulary, were not considered harmful as were not directed at a particular person or a group (Example 12, "dupa"/"ass"). On the other hand, some tweets, although also not being directed at anyone in particular, were encouraging the use of illegal substances, thus were considered as harmful (Example 3).

4. Dataset Analysis and Discussion

4.1. General Statistical Analysis

The overall number of tweets the final dataset contained was 11,041 with 10,041 included in the training set and 1000 in the test set. The layperson annotators agreed upon most of the annotations, with overall 91.38% of agreements, with a very small number of tweets which either of the annotators was unable to tag (84, or 0.76%). This was is a high percentage of agreements, however, this high percentage was mostly due to the fact that most of the annotators agreed upon non-harmful tweets, which comprised most of the dataset (over 89.76%). Among the final number of harmful tweets, the annotators fully agreed on the cyberbullying class (1) for only 106 (0.96%) and on the hate-speech class (2) for only 73 tweets (0.66%). Moreover, even some of the tweets with full agreement ended up being corrected by the expert annotator to other class, thus making the agreements even weaker. In general, it can be said, that layperson annotators can specify with fair amount of confidence that a tweet is not harmful (even if it contains some vulgar words), and to some extent can spot when there is something wrong or undesirable with the tweet, but in majority struggle to specify, what exactly is the reason of a tweet being undesirable. This provides proof that

for specific problems such as cyberbullying, an expert annotation is required. This has been highlighted by Ptaszynski and Masui (2018) for their ten years of research, although there has been a large number of research applying in annotation laypeople, such as undergraduate students or Mechanical Turk workers (Cano et al. 2013, Dinakar et al. 2012). When it comes to the comparison between the training and the test set, the latter contained a slightly higher ratio of harmful tweets (8.48% for training set vs. 13.40% for test set), which could cause some problems for the participants.

Table 1: Examples of tweets with annotated information

No.	Tweet contents	Score	Tag	Phrases
1	@anonymized_account Jebać, jebać, nic się nie bać.	2		jebać
2	@anonymized_account @anonymized_account Ty się oszuście i kłamco od nas odpierdol. My sobie damy radę bez twoich kłamstw	2	atk, wul, atk	oszuście, kłamco, odpierdol się
3	Picie to przyszłość, piękna i nieznana, szukajcie zapomnienia, próbujcie różnych trunków, łączcie środki psychoaktywne, Powodzenia	2		
4	@anonymized_account Ale ty jesteś żenujący. Dno metr mułu.	2		
5	@anonymized_account Jaki hejt. Hejt to jest na Tuska, Komorowskiego, przecież to nie Polacy, śmierć wrogom ojczyzny. Obudź się	2	gro	śmierć wrogom ojczyzny
6	@anonymized_account Wio endekdu, ścierwa, zdrajcy, szubienica	1	szy	ścierwa, endek, szubienica, zdrajcy
7	@anonymized_account Jeszcze was zjemy i wysramy	1	atk, gro	zjemy, wysramy, jeszcze
8	@anonymized_account @anonymized_account A ty wieś kretynie CONTI jest Acta2 i czego dotyczy? Najpierw przeczytaj a potem się wypowiadaj.	1	atk	kretynie
9	@anonymized_account @anonymized_account Ty pajacu, zmień sobie herb na pusty łeb.	1	atk	pajacu, pusty łeb
10	jak ja się źle czulem jak byłem dzieckiem w kościele to głow mala, szopka do kwadratu, nie mogłem tego wytrzymać	0		
11	Kiedy Christina wychodzi za mąż	0		
12	@anonymized_account kot też się załapał na fotke, a raczej jego dupa :)	0		

Table 2: General statistics of the dataset

	#	% of all	% of set
Overall # of tweets	11041	100.00%	
# of tweets annotator 1 was unable to tag	38	0.34%	
# of tweets annotator 2 was unable to tag	46	0.42%	
# of tweets where annotators agreed	10089	91.38%	
# of tweets where annotators agreed for 0	9910	89.76%	
# of tweets where annotators agreed for 1	106	0.96%	
# of tweets where annotators agreed for 2	73	0.66%	
# of tweets where annotators disagreed	952	8.62%	
# of retweets (RT) which slipped through	709		
# of final 0	10056	91.08%	
# of final 1	278	2.52%	
# of final 2	707	6.40%	
# of all harmful	985	8.92%	
Training set	10041	90.94%	
# of final 0	9190	83.24%	91.52%
# of final 1	253	2.29%	2.52%
# of final 2	598	5.42%	5.96%
# of all harmful	851	7.71%	8.48%
Test set	1000	9.06%	
# of final 0	866	7.84%	86.60%
# of final 1	25	0.23%	2.50%
# of final 2	109	0.99%	10.90%
# of all harmful	134	1.21%	13.40%

Apart from the above statistics, there was also a fairly large number of retweets that slipped through both the data preparation process as well as a later annotation (709 or 6.42%). All of those tweets were not official retweets, but tweet quotations starting with a short comment "RT". This situation will need to be taken into consideration when creating the second, improved version of the dataset in the future.

4.2. Discussion on Specific Tweet Examples

The whole annotation process provided a number of valuable insights reported by the annotators. For example, many annotators noticed that the meaning of most tweets depended on the context, and when the context was unclear, it was difficult to evaluate them in the given categories (especially for the harmful category). The entire conversation between Twitter users would facilitate better assessment, and show the context in which the given tweet was published. This problem could be solved by clustering tweets into conversation threads. We will propose a method for automatic clustering of tweets into coherent threads. This could be done by incorporating, a specific meta-information about at which tweet the message is addressed at, provided by the API (`in_reply_to_status_id`), or taking additional

advantage of user quotations (`@user`), which appear at the beginning of tweets usually as responses, together with time between tweets, which could additionally suggest the tweet being a response with the higher confidence the shorter the time between tweets.

When it comes to the tweets regarding the authorities or public figures, in cases where the tweet represented only an opinion without insult or defamation, most annotators assigned them with the non-harmful label. This was due to the general common sense that expressing an opinion is not punishable in itself. The annotators also highlighted the need for constant awareness for separating one's own worldviews from the criticisms about the authorities in order to preserve the objectivity during the annotation process. Also, although there was a clear difference between the language of supporters of the right and the left wing (e.g., "lemingi"/"lemmings" vs. "pisiory"/"PiS-supporters"), more general patterns appeared on both sides, also outside of the political topics.

4.3. Examples of Tweets with Additional Explanations of Reasoning Behind Annotation

Not harmful

1. "500+ bardzo na plus jednak ten rzad wykorzystał dorobek poprzednich rządów do swojego populizmu chorego"
2. "Mamy do czynienia z najgorszym prezydentem RP w historii. Kropka."
   — Both samples considered a general opinion. Score: 0.
3. "@anonymized_account I kurwa mamy ta wolność"
   — Despite the appearance of a strong vulgar language ("kurwa"), the expression itself does not indicate any punishable harmful action, thus the tweet was annotated as non-harmful. Score: 0.
4. "@anonymized_account @anonymized_account @anonymized_account Matka Boska była półką i Jezus też."
   — Although at a first glance this tweet might look like a blasphemy, the harmful effect is caused mostly by an error of a spellchecker ("Polka" wrongly corrected to "półka"). Score: 0.
5. "Biało-Czerwoni brawo, brawo, brawo! Zbigniew Boniek i Adam Nawałka - wyrazy szacunku. Robert Lewandowski - wielkie podziękowania!"
   — Score: 0.
6. "@anonymized_account WISŁA KRAKÓW !! brawo za dzisiejszy mecz :)"
   — Score: 0.

Cyberbullying

1. "[tel. no. anonymized] w Bułgarii numer ten uważany jest za przeklęty ponieważ podobno każdy z jego właścicieli umierali po kilku dniach"

— Considered a type of phishing. By trying to check whether a given telephone number is actually cursed (and it is fair to assume there will be such people) it is possible to fall into a trap and risk of being robbed of personal information. Score: 1.

2. "@anonymized_account Tu stary chuju PZPRowski zajmij się swoimi komuchami z PiSu."
   — A typical attack with accusations of supporting communism. Despite originally being a response to a public account, it looks like an attack at a specific person. Score: 1.

Hate-speech/other harmful

1. "@anonymized_account @anonymized_account @anonymized_account Rozumiem, że jutro w sejmie powie to pani protestującym. Załgane pisowskie skurwysyny."
   — Accumulation of profanity. Score: 2.
2. "@anonymized_account @anonymized_account Was, gnidy powinno się zaorać na metr w głąb i grubo posypać niegaszonym wapnem. A dla pewności zbombardować napalmem."
   — A typical case of hate-speech consisting of over exaggerated death threats aimed at a public person. Score: 2.
3. "KAIN TEŻ ZABIŁ BRATA ALE NIE ŚWIĘTOWAŁ TEGO CO MIESIĄC I NIE STAWIAŁ POMNIKÓW NA TĘ OKOLICZNOŚĆ."
   — A good example of a context-dependent mocking/ridiculing of a public person. Although the name of the object of ridicule is not explicitly stated, thus it is difficult to grasp by a computational method, it is easy to understand by a normal person following political events in Poland. Score: 2.
4. "@anonymized_account MILIONY POLAKÓW CZEKA NA BADANIA PSYCHIATRYCZNE LISA PO WPISACH WIDAĆ NIE ZRUWNOWARZENIE PSYCHICZNE I CIĄGŁĄ DEPRESJE"
   — The tweet, originally directed at a specific public person (a TV announcer), violates privacy and can be considered a public slander. Moreover, the expression of the need for a psychiatric checkup and the use of mental illness (depression), which typically are diagnosed by a psychiatrist, is in the area of a private matter, while in this case are used as a slur. Score: 2 (slandering of a public person).
5. "@anonymized_account Elo swastyka na ryju kiedy będzie, sorry że ciągle pytam?"
   — Although the tweet mainly represents a general negative opinion about the UK, it also focuses on a specific person (originally a right-wing publicist) accusing him of fascism. Score: 2 (accusing a public person of fascism).
6. "@anonymized_account Lzy ogromne, kiedyś usunąłem ciążę, nie mów nikomu"
   — Considered a harmful tweet written with an aim of provocation, signaled by the phrase "nie mów nikomu", since the tweet is publicly viewable, in a socially controversial topic (abortion). Score: 2.

5. Task Description

In the pilot task designed for the dataset, the participants are to determine whether an Internet entry is classifiable as harmful (cyberbullying, hate-speech) or not. The particular goal is to classify the provided tweets into cyberbullying/harmful and non-cyberbullying/non-harmful with the highest possible Precision, Recall, balanced F-score and Accuracy. There are two subtasks.

5.1. Description of Subtasks

Subtask 1. Harmful vs non-harmful

In this subtask, the participants are to distinguish between normal/non-harmful tweets (class: 0) and tweets that contain any kind of harmful information (class: 1). This includes cyberbullying, hate speech and related phenomena.

Subtask 2. Type of harmfulness

In this subtask, the participants are to distinguish between three classes of tweets: 0 (non-harmful), 1 (cyberbullying), 2 (hate-speech). There are various definitions of both cyberbullying and hate-speech, some of them even putting those two phenomena in the same group. The specific conditions on which we based our annotations for both cyberbullying and hate-speech have been worked out during ten years of research (Ptaszynski and Masui 2018). However, the main and definitive condition to distinguish the two is whether the harmful action is addressed towards a private person(s) (cyberbullying), or a public person/entity/larger group (hate-speech). Other specific definitions and guidelines applied in creation were described in Section 3.

5.2. Evaluation

The scoring for the first subtask is done based on standard Precision (P), Recall (R), Balanced F-score (F1) and Accuracy (A), on the basis of the numbers of True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN), according to the below equations. The winning condition would be to have the highest balanced F-score. However, in the case of F-score equal for two or more participants, the one with higher Accuracy would be considered as the winner. Furthermore, in case of the same F-score and Accuracy, a priority shall be given to the results as close as possible to BEP (break-even-point of Precision and Recall).

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1 = \frac{2 \cdot P \cdot R}{P + R}$$

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

The scoring for the second subtask is based on two measures, namely, Micro-Average F-score (*microF*) and Macro-Average F-score (*macroF*). Micro-Average F-score is calculated similarly as in standard equation for F-score, but on the basis of Micro-Averaged Precision and Recall, which are calculated according to the below equations. Macro-Average F-score is calculated on the basis of Macro-Averaged Precision and Recall, which are calculated according to the following equations. The winning condition would mean at first the highest *microF*. This measure treats all instances equally, which is a fair approach since the number of instances is different for each class. However, in the case of equal results for

5.3. Task Participants

There were fourteen overall submissions to the task sent by nine unique teams. All the submitting teams attempted to solve the first subtask (6-1), which was a computationally simpler problem of binary classification of tweets into harmful and non-harmful, while there were only eight attempts at solving the second subtask (6-2), which was the three-class classification problem. Below we briefly describe the systems proposed by each team. All teams and submitted systems were summarized in Tables 4 and 5. Below we present short descriptions of teams and systems, for which the authors decided to describe their systems in this volume.

Korzeniowski et al. (2019) from Sigmoidal team presented three approaches, namely, fine-tuning of a pre-trained ULMFiT, fine-tuning a pre-trained BERT model, and using the TPOT library to find the optimal pipeline. The last of the proposed approaches, namely, TPOT with a logistic regression classifier with non-trivial feature engineering, scored as second in the Subtask 6-2 (detection of different types of harmful information).

Wróbel (2019) after, firstly preprocessing the data, tested two classifiers, namely, Flair trained on character-based language model and FastText.

Prońko (2019) compared some of the popular text classification models, such as Ngrams and MLP, word embedding and sepCNN, and Flair with different embeddings, in combination with LSTM and GRU with word embeddings trained from scratch.

Ciura (2019) applied Przetak, a tool wich identifies abusive and vulgar speech in Polish, to detect cyberbullying. Przetak is a dynamically-linked library written in Go, which uses logistic regression over character 5-grams of words. This approach scored as second in the first subtask (6-1) on detection of any type of harmful information.

Biesek (2019) presented three approaches with different architectures and level of complexity, namely, a standard machine learning SVM classifier with TF-IDF, a bidirectional GRU network, and a deep Flair framework model with Contextual String Embeddings. The model applying SVM outperformed all other submissions for Subtask 6-2.

Krasnowska-Kieraś and Wróblewska (2019) proposed a simple neural network setup with various feature sets, including LASER embeddings, stylistic qualifiers signalling various informal modifications (e.g., vulgar, colloquial, depreciating, etc.), a list of offensive words, and character n-grams. To supplement for the imbalanced data samples they also divided separate tweets into separate sentences with full stop, and added artificially created back-translations (Polish-Russian-Polish, etc.) of tweets containing insufficient number of classes.

Czapla et al. (2019) from n-waves team based their approach on transfer learning, which uses large amounts of unlabelled text to reduce data necessary for a target task. They also showed that initial weights of language model play an important role in model performance on the target task, and proposed a mechanism to test if the sampled initial weights are suitable for the target task. Their solution proposed for Subtask 6.1 achieved state-of-the-art performance and took first place.

6. Results of First Shared Task for Automatic Cyberbullying Detection in Polish Twitter

6.1. Baselines

The dataset was not balanced, namely, the ratio of each class was different (see Table 1). Therefore to get a more objective view on how participants of the task managed to classify the data, we first prepared a number of simple baselines.

The first set of baselines consisted of simple classifiers assigning scores without any insight into data:

A. classifier always assigning score 0

B. classifier always assigning score 1

C. classifier always assigning score 2 (only for Subtask 2)

D. classifier assigning random score: 0/1 (for Subtask 1)

E. classifier assigning random score: 0/1/2 (for Subtask 2).

As a result, all simple baselines scored very low. For Subtask 1, baseline A (always 0) scored $F1 = 0$, which was predictable and simply means it is not possible to simply disregard the problem as too easy. Baseline D (random) also scored F1=0, which additionally means that it is not possible to solve to problem of cyberbullying detection by simply flipping a coin. Baseline B (always 1), by the definition, was able to catch all harmful samples (Recall = 100%), but such a simplistic assumption results in a very low Precision (13.4%), thus causing the F-score to be also very low (23.63%).

As for the second subtask, for the same reasons as in subtask 1, baselines B (always 1), C (always 2), and E (random) also achieved very low scores. Baseline A (always 0), achieved a high *microF* (86.6%) due to automatically winning for non-harmful cases, which were the

majority in the dataset. However, *macroF* provided a sufficient clarification of the score, is in fact very low (30.94%).

Table 3: Results of simple baselines for Subtask 1

Subtask 1	**P**	**R**	**F1**	**A**
Baseline A	0.00%	0.00%	0.00%	86.60%
Baseline B	13.40%	100.00%	23.63%	13.40%
Baseline D	0.00%	0.00%	0.00%	86.60%

Table 4: Results of simple baselines for Subtask 2

Subtask 2	**microF**	**macroF**
Baseline A	86.60%	30.94%
Baseline B	2.50%	1.63%
Baseline C	10.90%	6.55%
Baseline E	31.20%	31.16%

6.2. Results of Task Participants

Subtask 6-1

In the first subtask, out of fourteen submissions, there were nine unique teams: n-waves, Warsaw University of Technology, Sigmoidal, CVTimeline, AGH & UJ, IPI PAN, UWr, and two independent researchers. Some teams submitted more than one system proposal, in particular: Sigmoidal (3 submissions), independent (3 by one researcher), CVTimeline (2). Participants used a number of various techniques, usually widely available open source solutions, trained and modified to match the Polish language and the provided dataset when it was required. Some of the methods used applied, e.g., fast.ai/ULMFiT (`http://nlp.fast.ai/`), SentencePiece (`https://github.com/google/sentencepiece`), BERT (`https://github.com/google-research/bert`), tpot (`https://github.com/EpistasisLab/tpot`), spaCy (`https://spacy.io/api/textcategorizer`), fasttext (`https://fasttext.cc/`), Flair (`https://github.com/zalandoresearch/flair`), neural networks (in particular with GRU) or more traditional SVM. There were also original methods, such as Przetak (`https://github.com/mciura/przetak`). The most effective approach was based on recently released ULMFiT/fast.ai, applied for the task by the n-waves team. The originally proposed Przetak was second-best, while third place achieved a combination of ULMFiT/fast.ai, SentencePiece and BranchingAttention model. The results for of all teams participating in Subtask 6-1 were represented in Table 5.

Subtask 6-2

In the second subtask, out of eight submissions, there were five unique submissions. The teams that submitted more than one proposal were: independent (3 submissions) and Sigmoidal (2). Methods that were the most successful for the second subtask were based on: svm (winning method proposed by independent researcher Maciej Biesek), a combination of ensemble of classifiers from spaCy with tpot and BERT (by Sigmoidal team), and fasttext (by the AGH & UJ team). The results for of all teams participating in Subtask 6-2 were represented in Table 6. Interestingly, although the participants often applied new techniques, most of them applied only lexical information represented by words (words, tokens, word embeddings, etc.), while none of the participants attempted more sophisticated feature engineering and incorporate other features such as parts-of-speech, named entities, or semantic features.

7. Conclusions

We presented the first dataset in the Polish language, together with an open shared task for automatic cyberbullying detection, to contribute to solving the recently growing problem of cyberbullying and hate-speech appearing on the Internet.

The dataset, together with the open shared task supplementing the dataset, allows the users to try their classification methods to determine whether an Internet entry (e.g., a tweet) is classifiable as harmful (cyberbullying/hate-speech) or non-harmful. The entries contain tweets collected from openly available Twitter discussions and were provided as such, with minimal preprocessing. The only applied preprocessing was for anonymization of mentions so private persons mentioned in tweets were not revealed to the public.

The goal of the main subtask was to classify the tweets into harmful (cyberbullying or hate-speech) and non-harmful with the highest possible Precision, Recall, balanced F-score and Accuracy. In an additional subtask, the goal was to differentiate between various types of harmful information, in particular cyberbullying and hate-speech, as well as non-harmful.

There were fourteen submissions from nine unique teams. All submissions attempted to solve the first binary classification subtask, while only eight submissions were for the second subtask. The participants mostly used widely available solutions for text classification, such as fast.ai/ULMFiT, SentencePiece, BERT, spaCy, fasttext, or more traditional SVM. Original methods were in minority, although appeared quite successful. Best methods were based, either on recently proposed solutions (fast.ai) or original methods (Przetak) for the first subtask, as well as more traditional machine learning methods (SVM) for the second subtask.

8. Future Work

As this was the first task of this kind for the Polish language, and one of the few first in general, we acknowledge that there is room for improvement. In particular, we plan on enlarging the dataset. At this time the dataset contains 11 thousand tweets with only about 9% of harmful

Table 5: Results of participants for Subtask 6-1

Submission author(s)	Affiliation	Submitted system	Precision	Recall	F-score	Accuracy
Piotr Czapla, Marcin Kardas	**n-waves**	**n-waves ULMFiT**	**66.67%**	**52.24%**	**58.58%**	**90.10%**
Marcin Ciura	independent	Przetak	66.35%	51.49%	57.98%	90.00%
Tomasz Pietruszka	Warsaw University of Technology	ULMFiT + SentencePiece + BranchingAttention	52.90%	54.48%	53.68%	87.40%
Sigmoidal Team (Renard Korzeniowski, Przemysław Sadowski, Rafał Rolczyński, Tomasz Korbak, Marcin Możejko, Krystyna Gajczyk)	Sigmoidal	ensamble spacy + tpot + BERT	52.71%	50.75%	51.71%	87.30%
Sigmoidal Team	Sigmoidal	ensamble + fastai	52.71%	50.75%	51.71%	87.30%
Sigmoidal Team	Sigmoidal	ensenble spacy + tpot	43.09%	58.21%	49.52%	84.10%
Rafał Prońko	CVTimeline	Rafal	41.08%	56.72%	47.65%	83.30%
Rafał Prońko	CVTimeline	Rafal	41.38%	53.73%	46.75%	83.60%
Maciej Biesek	independent	model1-svm	60.49%	36.57%	45.58%	88.30%
Krzysztof Wróbel	AGH, UJ	fasttext	58.11%	32.09%	41.35%	87.80%
Katarzyna Krasnowska-Kieraś, Alina Wróblewska	IPI PAN	SCWAD-CB	51.90%	30.60%	38.50%	86.90%
Maciej Biesek	independent	model2-gru	63.83%	22.39%	33.15%	87.90%
Maciej Biesek	independent	model3-flair	81.82%	13.43%	23.08%	88.00%
Jakub Kuczkowiak	UWr	Task 6: Automatic cyberbullying detection	17.41%	32.09%	22.57%	70.50%

Table 6: Results of participants for Subtask 6-2

Submission author(s)	**Affiliation**	**Name of the submitted system**	**Micro-Average F-score**	**Macro-Average F-score**
Maciej Biesek	independent	**model1-svm**	**87.60%**	**51.75%**
Sigmoidal Team (Renard Korzeniowski, Przemysław Sadowski, Rafał Rolczynski, Tomasz Korbak, Marcin Możejko, Krystyna Gajczyk)	Sigmoidal	ensamble spacy + tpot + BERT	87.10%	46.45%
Krzysztof Wróbel	AGH, UJ	fasttext	86.80%	47.22%
Maciej Biesek		model3-flair	86.80%	45.05%
Katarzyna Krasnowska-Kieraś, Alina Wróblewska	IPI PAN	SCWAD-CB	83.70%	49.47%
Maciej Biesek	independent	model2-gru	78.80%	49.15%
Jakub Kuczkowiak	UWr	Task 6: Automatic cyberbullying detection	70.40%	37.59%
Sigmoidal Team	Sigmoidal	ensamble + fastai	61.60%	39.64%

ones. In the future, we plan to at least double the size to contain at least a comparable number of harmful tweets, as in research in other languages (Ptaszynski and Masui 2018, Cano et al. 2013, Dinakar et al. 2012). We also need to improve the procedure for the preprocessing of the dataset to make sure no noise or redundant information is contained. In particular, the present dataset contained a number of unofficial retweets (tweets starting with RT). A thorough analysis also revealed some remaining tweets with unusual URLs, which slipped through the URL filtering stage.

Moreover, in a future version of the dataset we also plan to annotate on the tweets roles of participants in cyberbullying, such as: 1) victim, 2) bully and 3) bystanders (3–1 bully-supporter, and 3–2 victim–defender) to get a wider grasp on the problem of bullying as a process taking place on the Internet.

Finally, when it comes to the classification methods, although the participants used new widely available techniques, only lexical information was applied (words, tokens, word embeddings, etc.). Since it has been shown that a thorough feature engineering is useful in cyberbullying detection (Ptaszynski et al. 2017), we encourage future participants to incorporate other features, except words/tokens, e.g., parts-of-speech, named entities, or semantic features.

References

Biesek M. (2019). *Comparison of Traditional Machine Learning Approach and Deep Learning Models in Automatic Cyberbullying Detection for Polish Language*. In Ogrodniczuk and Kobyliński (2019), pp. 121–126.

Cano E., He Y., Liu K. and Zhao J. (2013). *A Weakly Supervised Bayesian Model for Violence Detection in Social Media*. In *Proceedings of the 6th International Joint Conference on Natural Language Processing (IJCNLP 2013)*, Nagoya, Japan.

Ciura M. (2019). *Przetak: Fewer Weeds on the Web*. In Ogrodniczuk and Kobyliński (2019), pp. 127–133.

Czapla P., Gugger S., Howard J. and Kardas M. (2019). *Universal Language Model Fine-Tuning for Polish Hate Speech Detection*. In Ogrodniczuk and Kobyliński (2019), pp. 149–159.

Dinakar K., Jones B., Havasi C., Lieberman H. and Picard R. (2012). *Commonsense Reasoning for Detection, Prevention and Mitigation of Cyberbullying*. „ACM Transactions on Intelligent Interactive Systems", 2(3).

Dooley J. J., Pyżalski J. and D. C. (2009). *Cyberbullying Versus Face-to-Face Bullying: A Theoretical and Conceptual Review*. „Zeitschrift für Psychologie/Journal of Psychology", 217(4), p. 182–188.

Hatakeyama S., Masui F., Ptaszynski M. and Yamamoto K. (2015). *Improving Performance of Cyberbullying Detection Method with Double Filtered Point-wise Mutual Information*. In *Proceedings of the Demo Session of The 2015 ACM Symposium on Cloud Computing 2015 (ACM-SoCC 2015)*, Kohala Coast, Hawai'i.

Hatakeyama S., Masui F., Ptaszynski M. and Yamamoto K. (2016a). *Statistical Analysis of Automatic Seed Word Acquisition to Improve Harmful Expression Extraction for Cyberbullying Detection*. „Proceedings of the International Conference on Advanced Technology Innovation 2016 (ICATI2016)".

Hatakeyama S., Masui F., Ptaszynski M. and Yamamoto K. (2016b). *Statistical Analysis of Automatic Seed Word Acquisition to Improve Harmful Expression Extraction in Cyberbullying Detection*. „International Journal of Engineering and Technology Innovation", 6(2), p. 165–172.

Korzeniowski R., Rolczyński R., Sadownik P., Korbak T. and Możejko M. (2019). *Exploiting Unsupervised Pre-training and Automated Feature Engineering for Low-resource Hate Speech Detection in Polish*. In Ogrodniczuk and Kobyliński (2019), pp. 141–148.

Krasnowska-Kieraś K. and Wróblewska A. (2019). *A Simple Neural Network for Cyberbullying Detection*. In Ogrodniczuk and Kobyliński (2019), pp. 161–163.

Lempa P., Ptaszynski M. and Masui F. (2015). *Cyberbullying Blocker Application for Android*. In *Proceedings of 7th Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics (LTC'15), The First Workshop on Processing Emotions, Decisions and Opinions (EDO 2015)*, pp. 408–412, Poznań, Poland.

Nitta T., Masui F., Ptaszynski M., Kimura Y., Rzepka R. and Araki K. (2013a). *Cyberbullying Detection Based on Category Relevance Maximization*. In *Proceedings of the Demo Session of 20th International Conference on Language Processing and Intelligent Information Systems (LP & IIS 2013)*, Warsaw, Poland.

Nitta T., Masui F., Ptaszynski M., Kimura Y., Rzepka R. and Araki K. (2013b). *Detecting Cyberbullying Entries on Informal School Websites Based on Category Relevance Maximization*. In *Proceedings of the 6th International Joint Conference on Natural Language Processing (IJCNLP 2013)*, pp. 579–586, Nagoya, Japan.

Ogrodniczuk M. and Kobyliński Ł., editors (2019). *Proceedings of the PolEval 2019 Workshop*, Warsaw. Institute of Computer Science, Polish Academy of Sciences.

Prońko R. (2019). *Simple Bidirectional LSTM Solution for Text Classification*. In Ogrodniczuk and Kobyliński (2019), pp. 111–119.

Ptaszynski M., Dybala P., Matsuba T., Masui F., Rzepka R., Araki K. and Momouchi Y. (2010a). *In the Service of Online Order: Tackling Cyber-Bullying with Machine Learning and Affect Analysis*. „International Journal of Computational Linguistics Research", 1(3), p. 135–154.

Ptaszynski M., Dybala P., Matsuba T., Masui F., Rzepka R. and Araki K. (2010b). *Machine Learning and Affect Analysis Against Cyber-Bullying*. In *Proceedings of The 36th Annual Convention of the Society for the Study of Artificial Intelligence and Simulation of Behaviour (AISB-10)*, pp. 7–16. De Montfort University, Leicester, UK.

Ptaszynski M., Masui F., Kimura Y., Rzepka R. and Araki K. (2015). *Extracting Patterns of Harmful Expressions for Cyberbullying Detection*. In *Proceedings of 7th Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics*

(LTC'15), The First Workshop on Processing Emotions, Decisions and Opinions (EDO 2015), pp. 370–375, Poznan, Poland.

Ptaszynski M., Masui F., Nakajima Y., Kimura Y., Rzepka R. and Araki K. (2016a). *Detecting Cyberbullying with Morphosemantic Patterns*. In *Proceedings of the Joint 8th International Conference on Soft Computing and Intelligent Systems and 17th International Symposium on Advanced Intelligent Systems (SCIS-ISIS 2016)*, pp. 248–255, Sapporo, Japan.

Ptaszynski M., Masui F., Nitta T., Hatakeyama S., Kimura Y., Rzepka R. and Araki K. (2016b). *Sustainable Cyberbullying Detection with Category-Maximized Relevance of Harmful Phrases and Double-Filtered Automatic Optimization*. „International Journal of Child-Computer Interaction (IJCCI)", 8, p. 15–30.

Ptaszynski M., Kalevi J., Eronen K. and Masui F. (2017). *Learning Deep on Cyberbullying is Always Better Than Brute Force*. In *Proceedings of the IJCAI 2017 3rd Workshop on Linguistic and Cognitive Approaches to Dialogue Agents (LaCATODA 2017)*, Melbourne, Australia.

Ptaszynski M., Masui F., Kimura Y., Rzepka R. and Araki K. (2018). *Automatic Extraction of Harmful Sentence Patterns with Application in Cyberbullying Detection*. In *Human Language Technology. Challenges for Computer Science and Linguistics*, pp. 349–362. Springer International Publishing. Lecture Notes in Computer Science (LNCS) vol. 10930.

Ptaszynski M., Masui F., Kimura Y., Rzepka R. and Araki K. (2019). *Brute Force Sentence Pattern Extortion from Harmful Messages for Cyberbullying Detection*. „Journal of the Association for Information Systems (JAIS)".

Ptaszynski M. E. and Masui F. (2018). *Automatic Cyberbullying Detection: Emerging Research and Opportunities*. IGI Global Publishing.

Tversky A. and Kahneman D. (1974). *Judgment under Uncertainty: Heuristics and Biases*. „Science, 185(4157)", pp. 1124–1131.

Wróbel K. (2019). *Approaching Automatic Cyberbullying Detection for Polish Tweets*. In Ogrodniczuk and Kobyliński (2019), pp. 135–140.

Simple Bidirectional LSTM Solution for Text Classification

Rafał Prońko (CVTimeline)

Abstract

In this paper I present a summary of my results from the competition that took place this year and was organized by PolEval[1]. One of the tasks of this competition was the detection of offensive comments in social media. By joining this competition, I set myself a goal to compare some of the popular text classification models used on Kaggle or recommended by Google. That's why during the competition I went through models such as: Ngrams and MLP, word embedding and sepCNN, Flair from Zalando with different embedding, combination of LSTM and GRU with word embedding trained from scratch.

Keywords

Natural Language Processing, Bidirectional LSTM, Text Classification, Neural Network

1. Introduction

Text classification is the process of assigning certain labels / classes to it depending on the content you have. It is one of the most popular NLP tasks with many applications such as: sentiment detection, topic labeling, spam detection, classification of ads, classification of work titles (Javed et al. 2016), detection of toxic comments[2].

Unstructured data in the form of text is everywhere: emails, chats, web pages, social media, request from the clients, survey responses, and more. Text can be an extremely rich source of information, but extracting insights from it can be hard and time-consuming due to its unstructured nature. Businesses are turning to text classification for structuring text in a fast and cost-efficient way to enhance decision-making and automate processes.

From year to year, more and more often hear about the problem of humiliating people on the Internet and especially in social media, in which you can quickly and easily put any

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

[2] `https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge`

content. Humiliating / offending other people with the use of social networks greatly affects the mental health of society – however, our youngest generation is always the one who is the most vulnerable when faced with an aggressive society. That is why it is so important to protect them and us all from the impact of toxic comments. It is not surprising, therefore, that more and more popular systems are being used to detect dangerous behaviors on the Internet. These solutions are created within research groups such as Conversation AI[3] or competitions on Kaggle[4] or even the last competition on PolEval.

2. Solution

On the last PolEval there were two tasks regarding the detection of toxic comments. In the first of them, the task was to find out whether the comment is harmful or not (ordinary binary classification) in the second task was to assign one of the three possible classes to the text. I focused on solving to solve the first one.

2.1. Dataset

The data collection for this issue contained 10041 examples of comments from social networks. A few examples of comments:

```
"Ja mam dla ciebie lepszą propozycję : powieś się gdzieś pod lasem
UB-ecka gnido .",1

"macie jej numer zdissujcie ją 8)",1

"Gosia się bardzo nudzi i chętnie z wami porozmawia. macie jej numer -
[NUMER TEL.] dzwonić może każdy, ale sms tylko plus.",1

"huju jebany oddawaj server gnoju glubi frajezre kutasie oddawaj
bo cie zajebie huju zzglosilem cie i tak nie będziesz miec konta
hahahahahahahhahahahaahha",1

"Czerwone Gitary, Historia jednej znajomości... i parawany które
istniały zawsze...",0
```

The data set had two fields – text and target. After a few minutes of data analysis it is easy to notice that:

1. the collection is unbalanced
2. the texts contain a few unnecessary elements that should be removed (they may disrupt the classification attempts).

The test set contained 1000 elements. The F1 score was selected as the main metric for evaluating the model.

[3] https://conversationai.github.io/
[4] http://kaggle.com

2.2. Solution Steps

In tasks related to NLP, it usually starts with steps related to preprocessing (Joulin et al. 2016) of text such as: deleting numbers, deleting unnecessary URLs, removing punctuation, deleting stop words etc. However, I chose the minimum version of preprocessing, i.e.:

1. all texts are converted into lower case
2. I have removed all unwanted elements from the text such as: punctuation, emoji and numbers.

Then, to test all my models, I decided on a certain algorithm. First of all, because the data set was very unbalanced, I decided to subdivide it into balanced subsets. Using the imbalanced-learn (Lemaître et al. 2017) library and using the functions of random undersampling, I created 5 subsets in which all the elements were marked as 1 and random elements that were marked as 0, selected in this way to balance the data set. Later, for each subset we create a separate model with separate embedding. I have determined the final result as the average of the individual models. The algorithm is presented in Figure 1.

```
read the data
clean the data
i=0
loop:
  undersample the data
  create sequence embedding
  fit the model with new embedding
  create the prediction for this model
  i += 1
  if i > 5 then
    goto prediction
  end if
end loop
prediction:
prediction = avg(predictions)
```

Figure 1: Cyberbullying detection algorithm

2.3. Models

Based on my experience in the classification of short texts, the first models which I used was the model/algorithm presented by Google[5]. There you can find a very interesting algorithm how to choose models to build a proof of concept (Figure 2).

Because I used the algorithm in which I chose subsets of the whole set which caused different S/W coefficients, I, therefore, tried both approaches to solve this problem.

[5] https://developers.google.com/machine-learning/guides/text-classification/

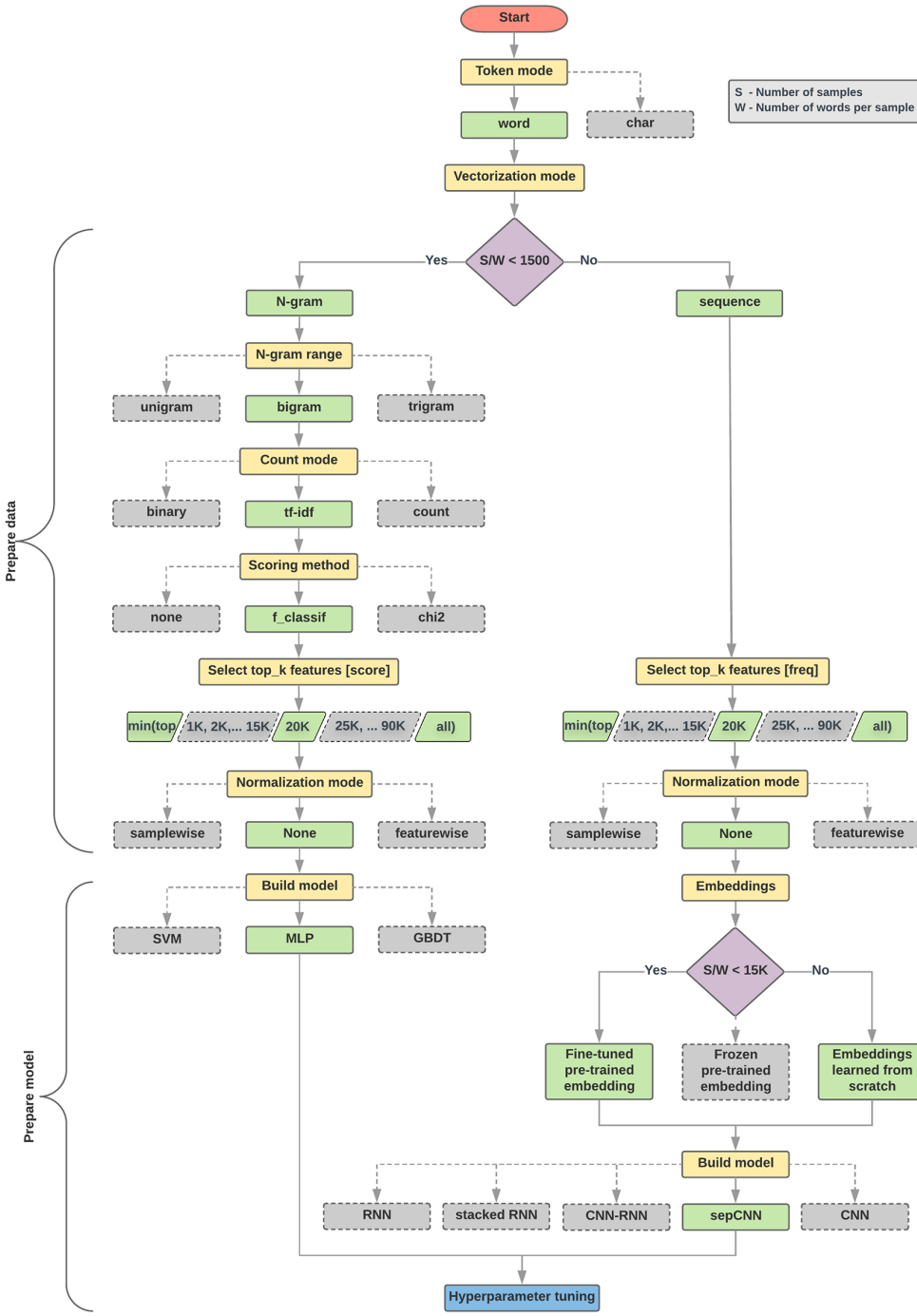

Figure 2: How to choose the model for text classification: `https://cloud.google.com/natural-language/docs/classify-text-tutorial`

NGrams and MLP

The text vectorization for this model was a TFIDF (Nyberg 2018) model based on words, ngrams in the 1–2 range. Then, using SelectKBest from the sklearn (Pedregosa et al. 2011) library for classification, I chose 20,000 best-matched ngrams. The MLP model itself was composed of three layers and a dropout (Srivastava et al. 2014) as a regularization method. The model is presented below.

```
Layer (type)                 Output Shape              Param #
=================================================================
dropout_36 (Dropout)         (None, 20000)             0
_________________________________________________________________
dense_36 (Dense)             (None, 1024)              20481024
_________________________________________________________________
dropout_37 (Dropout)         (None, 1024)              0
_________________________________________________________________
dense_37 (Dense)             (None, 1024)              1049600
_________________________________________________________________
dropout_38 (Dropout)         (None, 1024)              0
_________________________________________________________________
dense_38 (Dense)             (None, 1)                 1025
=================================================================
Total params: 21,531,649
Trainable params: 21,531,649
Non-trainable params: 0
_________________________________________________________________
```

sepCNN

This model was based on a neural network called sepCNN (Chollet 2016) and embedding trained from scratch. The model is presented below.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_32 (Embedding)     (None, 27, 300)           6000300
_________________________________________________________________
dropout_86 (Dropout)         (None, 27, 300)           0
_________________________________________________________________
separable_conv1d_221 (Separa (None, 27, 64)            20764
_________________________________________________________________
separable_conv1d_222 (Separa (None, 27, 64)            4480
_________________________________________________________________
max_pooling1d_86 (MaxPooling (None, 5, 64)             0
_________________________________________________________________
```

```
dropout_87 (Dropout)         (None, 5, 64)             0
_________________________________________________________________
separable_conv1d_223 (Separa (None, 5, 64)             4480
_________________________________________________________________
separable_conv1d_224 (Separa (None, 5, 64)             4480
_________________________________________________________________
max_pooling1d_87 (MaxPooling (None, 1, 64)             0
_________________________________________________________________
separable_conv1d_225 (Separa (None, 1, 128)            8640
_________________________________________________________________
separable_conv1d_226 (Separa (None, 1, 128)            17152
_________________________________________________________________
global_max_pooling1d_26 (Glo (None, 128)               0
_________________________________________________________________
dense_51 (Dense)             (None, 16)                2064
_________________________________________________________________
dense_52 (Dense)             (None, 1)                 17
=================================================================
Total params: 6,062,377
Trainable params: 6,062,377
```

Flair from Zalando

The next model I tried was the model created by Zalando (Akbik et al. 2019). This model is based on the PyTorch library. It has the ability to use many different word embedding like Glove (Pennington et al. 2014), FastText (Bojanowski et al. 2016), word2vec or language models like ELMo (Peters et al. 2018) or BERT (Devlin et al. 2018). It also has the ability to create embedding connections in a variety of ways. I tried to check all the possibilities. The best model based on BERT language model and classifier was build on bidirectional LSTM with dropout 0.2 and word dropout 0.2. I used only pretrained BERT multi language model.

Final solution

My final model was based on crossing Bidirectional networks (Pesaranghader et al. 2018) with GRU and LSTM. The final classifier was based on fully connected layers with concatenation of output from the language model part. Input to the network was word embedding trained from scratch.

The code is shown below.

```
x = SpatialDropout1D(0.3)(x)
x1 = Bidirectional(CuDNNGRU(512, return_sequences=True))(x)
x1 = Bidirectional(CuDNNGRU(512, return_sequences=True))(x1)
x1 = Bidirectional(CuDNNGRU(512, return_sequences=True))(x1)
```

```
x2 = Bidirectional(CuDNNLSTM(512, return_sequences=True))(x)
x2 = Bidirectional(CuDNNLSTM(512, return_sequences=True))(x2)
x2 = Bidirectional(CuDNNLSTM(512, return_sequences=True))(x2)
hidden = concatenate([
    GlobalMaxPooling1D()(x1),
    GlobalAveragePooling1D()(x1),
    GlobalMaxPooling1D()(x2),
    GlobalAveragePooling1D()(x2),
])
hidden = add([hidden, Dense(4096, activation='relu')(hidden)])
hidden = Dropout(0.2)(hidden)
hidden = Dense(2048, activation="relu")(hidden)
hidden = Dropout(0.2)(hidden)
hidden = Dense(2048, activation="relu")(hidden)
hidden = Dropout(0.2)(hidden)
hidden = Dense(1024, activation="relu")(hidden)
hidden = Dropout(0.2)(hidden)
hidden = Dense(1024, activation="relu")(hidden)
hidden = Dropout(0.2)(hidden)
result = Dense(op_units, activation=op_activation)(hidden)
```

3. Summary

Working on the PolEval solution I wanted to compare some of the popular text classification models. Looking at the overview I drew you could draw conclusions that LSTM/GRU type networks are the best for this type of task. However, it is always important to think about what it is that one wants at the end and whether the time devoted to the creation of the LSTM/GRU network is worth it. As I showed above, the differences between my final result and the results out of the box models are small. Table 4 presents the comparison of results for all systems.

Table 1: Summary of results

Model name	Precision	Recall	Balanced F-score	Accuracy
MLP	28.45	73.88	41.08	71.60
sepCNN	37.21	47.76	41.83	82.20
Flair from Zalando	46.97	23.13	31.00	86.20
LSTM/GRU	41.08	56.72	47.65	83.30

With proper tuning, it should be possible to get almost identical results in both cases.

However, as this competition and the results of other participants showed – it is worth paying attention to a relatively new approach to NLP, that is, ULMFiT (Howard and Ruder 2018), BERT and ELMo language models, because they give very good results. These models can be

trained on a generic body, for example, the Wiki, and then effectively reach the body that we have for our task. This is definitely a step towards using the popular transfer of learning in vision.

References

Akbik A., Bergmann T. and Vollgraf R. (2019). *Pooled Contextualized Embeddings for Named Entity Recognition*. In *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. to appear.

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

Chollet F. (2016). *Xception: Deep Learning with Depthwise Separable Convolutions*. „CoRR", abs/1610.02357.

Devlin J., Chang M., Lee K. and Toutanova K. (2018). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. „CoRR", abs/1810.04805.

Howard J. and Ruder S. (2018). *Fine-tuned Language Models for Text Classification*. „CoRR", abs/1801.06146.

Javed F., McNair M., Jacob F. and Zhao M. (2016). *Towards a Job Title Classification System*. „CoRR", abs/1606.00917.

Joulin A., Grave E., Bojanowski P. and Mikolov T. (2016). *Bag of Tricks for Efficient Text Classification*. „CoRR", abs/1607.01759.

Lemaître G., Nogueira F. and Aridas C. K. (2017). *Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning*. „Journal of Machine Learning Research", 18(17), p. 1–5.

Nyberg A. (2018). *Classifying Movie Genres by Analyzing Text Reviews*. „CoRR", abs/1802.05322.

Pedregosa F., Varoquaux G., Gramfort A., Michel V., Thirion B., Grisel O., Blondel M., Prettenhofer P., Weiss R., Dubourg V., Vanderplas J., Passos A., Cournapeau D., Brucher M., Perrot M. and Duchesnay E. (2011). *Scikit-learn: Machine Learning in Python*. „Journal of Machine Learning Research", 12, p. 2825–2830.

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

Pesaranghader A., Pesaranghader A., Matwin S. and Sokolova M. (2018). *One Single Deep Bidirectional LSTM Network for Word Sense Disambiguation of Text Data*. „CoRR", abs/1802.09059.

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

Srivastava N., Hinton G., Krizhevsky A., Sutskever I. and Salakhutdinov R. (2014). *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*. „Journal of Machine Learning Research", 15, p. 1929–1958.

Comparison of Traditional Machine Learning Approach and Deep Learning Models in Automatic Cyberbullying Detection for Polish Language

Maciej Biesek

Abstract

This paper presents systems for automatic cyberbullying detection in Polish tweets. Three approaches were proposed with different architectures and level of complexity, from standard machine learning SVM with TF-IDF technique through bidirectional GRU network and ending with deep Flair framework model with Contextual String Embeddings. They competed in both subtasks of Task 6 in PolEval 2019 and the first one, model1-svm, turned out to be the winning system of Task 6-2.

Keywords

natural language processing, cyberbullying detection, SVM, GRU

1. Introduction

Name-calling and rumor-spreading have been always an unpleasant and challenging aspect of adolescent life. But with the growing popularity of smartphones and social media, the problem went online and took a new form, even worst due to the anonymity – people hiding behind their nicknames tend to be more aggressive and abusive. A Pew Research Center survey finds that 59% of U.S. teens have personally experienced at least one of six types[1] of online harassment; for details see (Anderson 2018).

This is the problem social media companies like Facebook and Twitter have to deal on a daily basis, hiring people to filter out an inappropriate content. The recent research in Natural

[1] offensive name-calling, spreading of false rumors, receiving explicit images they did not ask for, having explicit images of them shared without their consent, psychical threats and constant asking where they are and what are they doing by someone other than a parent

Language Processing field focuses on automatic detection of cyberbullying (Ptaszynski et al. 2017). A system allowing fast annotation of different forms of online harassment would be particularly beneficial, resulting in creating a better Internet environment, especially for teenagers. Multiple datasets and competitions were created to build such systems, PolEval 2019 Task 6 is one of them and addresses the problem to the Polish language.

This paper is structured as follows: the dataset used in the task is presented in Section 2. Details about submitted models and their evaluation on the provided tweets can be found in Section 3 and Section 4, respectively. Conclusions and ideas for future research are described in Section 5.

2. Dataset

Dataset used to train and evaluate presented systems is the one provided by PolEval 2019 Task 6 organizers. It contains tweets collected from openly available Twitter discussions with applied anonymization of posts.

To make it useful for machine learning models dataset was preprocessed. From training data retweets (duplicated tweets with *RT* tag at the begging) were removed, in case of testing one only *RT* tag was deleted, but duplicated tweets were preserved. For both datasets each post was cleaned from whitespaces, digits, punctuation marks, emojis and *@anonymized_account* tags. They were lowercased and tokenized using spaCy tool[2] with Polish model[3]. Finally, posts with zero or only one token were deleted from the training dataset. Additionally, for **model1-svm** stopwords were removed, also using spaCy tool.

Statistics of number of tweets and their lengths in datasets after preprocessing were presented in Table 1.

Table 1: Statistics of training and testing datasets

	Training dataset	**Testing dataset**
Number of tweets	9362	1000
Avg of number of tokens in the sentence	10.49	10.44
Min of number of tokens in the sentence	2	2
Max of number of tokens in the sentence	29	27
Median of number of tokens in the sentence	10	10
75th percentile of number of tokens in the sentence	14	14
95th percentile of number of tokens in the sentence	19	19
99th percentile of number of tokens in the sentence	22	22

[2] `https://spacy.io/`
[3] `https://github.com/explosion/spaCy/pull/2974`

2.1. Task 6-1

The aim of this task was to distinguish between neutral tweets (class 0) and those which contain any kind of harmfulness (class 1), including cyberbullying, hate speech and so on. The dataset is highly imbalanced, the number of examples in each class is presented in Table 2.

Table 2: Frequencies of classes in Task 6-1

Type of tweets	Training dataset	Testing dataset
Neutral	8607	866
Harmful	755	134

2.2. Task 6-2

The goal of this task was to classify tweets into three classes: neutral (0), cyberbullying (1) and hate speech (2). The definition differentiating those two last phenomena was as follows: if the harmful action is addressed towards private persons we call it cyberbullying, if to a public person, entity or large group it is known as hate speech. As in Task 6-1, the dataset is imbalanced, Table 3 shows frequencies of three mentioned types of posts.

Table 3: Frequencies of classes in Task 6-2

Type of tweets	Training dataset	Testing dataset
Neutral	8607	866
Cyberbullying	243	25
Hate speech	512	109

3. Systems Description

Three different systems were proposed to tackle the problem of classifying tweets as neutral and harmful in Task 6-1 and as neutral, cyberbullying and hate speech in Task 6-2. Because of that in these tasks posts are the same (in Task 6-2 those previously annotated as harmful were further divided into cyberbullying and hate speech) architectures of models are the same in both cases except the number of output classes. The training dataset has been divided into training and validation set in 75:15 proportion with stratified sampling approach (that means both sets have the same frequencies of classes). The whole code of models is publicly available[4].

[4] https://github.com/maciejbiesek/poleval-cyberbullying

3.1. model1-svm

Using Support Vector Machines is known as a simple and efficient baseline for text classification since the 1990s (Joachims 1998). The input to the model is a tokenized dataset with stopwords removed. The pipeline is built using scikit-learn tool[5] and consists of three parts: first documents are converted to a matrix of token counts, then transformed to the TF-IDF representation and finally on these features linear SVM classifier is trained.

3.2. model2-gru

The architecture of this system is based on bidirectional GRU (Cho et al. 2014) (128 units in both directions) with dropout of value 0.5 and dense network on the top of concatenated final states from forward and backward passes. It contains 2 layers with ReLU activation function, intermediate of size 50 with 0.5 dropout and 2 or 3 neurons (depends on the task) as an output. To deal with class imbalancement weighted softmax cross entropy is used as a loss function, optimized using Adam (with default value of learning rate). It is implemented using Tensorflow library[6].

The input to the model are sequences of tokens with maximum length of 20 (short sentences are padded and longer ones trimmed[7]) mapped to the embedding matrix. In this case FastText vectors of 300 dimensions are used (Bojanowski et al. 2017).

The model was trained on the train set divided into batches of size 32 and after every epoch it was evaluated on the validation set and saved if the performance improved.

3.3. model3-flair

Contextual String Embeddings (Akbik et al. 2018) with internal states of trained character language model became state-of-the-art in sequence labeling task, eg. the system using them took the first place in Named Entity Recognition Task on PolEval 2018.

The model is implemented using Flair framework[8] and is based on bidirectional GRU network with hidden size of 128 and linear classifier. The input to it are FastText word embeddings stacked with forward and backward character-level language models trained on 1B words corpus of Polish (Borchmann et al. 2018). The assumption was that rich representation of input data would lead to high results in classification.

[5] `https://scikit-learn.org`
[6] `https://www.tensorflow.org`
[7] This value has been chosen after examination of Table 1, 95% of posts have less than 20 tokens.
[8] `https://github.com/zalandoresearch/flair`

4. Evaluation

Systems were evaluated on the test set provided by organizers of tasks. The evaluation script calculates Precision, Recall, balanced F-score and Accuracy in case of Task 6-1 and Micro-Average F-score and Macro-Average F-score in case of Task 6-2.

To compare the performance of models there were two baselines proposed: **baseline1** which is a vector with random values sampled from the set $\{0, 1\}$ in Task 6-1 and $\{0, 1, 2\}$ in Task 6-2. The other one, **baseline2** is a vector filled with zeros, as *neutral* is the most frequent class in both tasks.

Table 4 presents results of evaluation for Task 6-1 and Task 6-2. The highest value in each column is bold.

Table 4: Evaluation of models

Model	Prec	Rec	F-score	Acc
baseline1	13.65	**47.76**	21.23	52.50
baseline2	0.00	0.00	0.00	86.60
model1-svm	60.49	36.57	**45.58**	**88.30**
model2-gru	63.83	22.39	33.15	87.90
model3-flair	**81.82**	13.43	23.08	88.00

Model	Micro F-score	Macro F-score
baseline1	31.60	33.28
baseline2	86.60	30.94
model1-svm	**87.60**	**51.75**
model2-gru	78.80	49.15
model3-flair	86.80	45.05

5. Conclusions

In this paper three models were compared in the task of classifying tweets. Surprisingly, the simplest one turned out to be the best. It outperforms deep learning approaches in both tasks. Probably the reason of that is relatively small imbalanced dataset – more complicated networks can easily overfit in such case. Proposed solution for it would be acquiring more data with harmful examples.

There are more ideas of systems to investigate, e.g. it has been proved that convolutional neural networks perform remarkably well in sentence classification (Kim 2014) and even in cyberbullying detection (Ptaszynski et al. 2017). It would be also beneficial to use rich data representation (such as Flair embeddings) with more sophisticated classifier – perhaps linear one used in **model3-flair** was the reason of poor performance of this model.

The main conclusion that emerges from this research is that it is always beneficial to first check simple solutions – they are easy to create and, as it has been shown, can even win the contest.

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*, pp. 1638–1649, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Anderson M. (2018). *A Majority of Teens Have Experienced Some Form of Cyberbullying*. `https://www.pewinternet.org/2018/09/27/a-majority-of-teens-have-experienced-some-form-of-cyberbullying/`. [Online; accessed 12 May 2019].

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.

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 Science.

Cho K., van Merrienboer B., Gulcehre C., Bahdanau D., Bougares F., Schwenk H. and Bengio Y. (2014). *Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation*. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Joachims T. (1998). *Text Categorization with Support Vector Machines: Learning with Many Relevant Features*. In Nédellec C. and Rouveirol C. (eds.), *Machine Learning: ECML-98*, pp. 137–142, Berlin, Heidelberg. Springer.

Kim Y. (2014). *Convolutional Neural Networks for Sentence Classification*. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Ptaszynski M., Eronen J. K. K. and Masui F. (2017). *Learning Deep on Cyberbullying is Always Better than Brute Force*. In Rzepka R., Vallverdu J. J. and Wlodarczyk A. (eds.), *Linguistic And Cognitive Approaches To Dialog Agents (LaCATODA)*, number 1926 in CEUR Workshop Proceedings, pp. 3–10, Aachen.

Przetak: Fewer Weeds on the Web

Marcin Ciura

Abstract

This paper presents Przetak, a contribution to the cyberbullying detection task in PolEval 2019[1]. Although in the PolEval competition Przetak was used to detect cyberbullying, actually it identifies abusive and vulgar speech in Polish, blocking typical ways of circumventing the detection. Przetak is a dynamically-linked library written in Go. Under the hood, it uses logistic regression over character 5-grams of words. Thanks to FFI (Foreign Function Interface), the library can be used by programs written in many languages.

Keywords

natural language processing, text classification, cyberbullying, hate speech, offensive language

1. Introduction

In Task 6-1 of the PolEval 2019 competition, the participants were given a set of Polish-language tweets. Some of these tweets contained personal threats, ridiculing, insinuations, profane language, etc. The goal of the task was to develop a system to detect such instances of cyberbullying automatically. The author's system Przetak won the second place in this task, with F_1 score equal to 57.98%. The name of the system is a Polish word that means a riddle, i.e. a large sieve used to separate grain from chaff and seeds of weeds.

Although hate speech detection in English has been researched for several years (Ptaszyński and Masui 2018), the author is not aware of prior art for Polish.

2. The Present Approach

The author's approach is similar to that of Waseem and Hovy (2016) who detect hate speech by logistic regression over character N-grams.

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

Przetak contains three logistic regression models that recognize *toxic word forms*. The first model detects slurs, e.g. *oszołom* ('crackpot'). The other two models detect profanities. Montagu (1967) distinguishes abusive, adjurative, asseverative, ejaculatory, exclamatory, execratory, expletive, hortatory, interjectional, and objurgatory swearing. Pinker (2007) lists five categories of swearing: abusive, emphatic, dysphemistic, idiomatic, and cathartic. Przetak uses a simplified classification into abusive profanities, e.g. *ku**a* ('f***'), and asseverative profanities, e.g. *za**biście* ('f***ing awesome'). While asseverative profanities are not a good signal in hate speech detection, they can still be useful in Internet forum moderation, one of use cases the author envisions for Przetak.

The following sections outline the building of the models used in Przetak, implemented in Python[2], and features of Przetak as a library implemented in Go[3].

3. Building the Models

The word forms recognized by Przetak come from five word lists:

— 4 668 625 word forms from Polimorfologik 2.1[4], a Polish morphological dictionary

— 876 021 word forms from a corpus of 2 586 303 sentences downloaded from various sources

— 10 539 word forms of slurs

— 19 880 word forms of abusive profanities

— 230 word forms of asseverative profanities.

Replicating letters in profanities is fairly common online. Figure 1 presents the first step of model building: undoing the replication. Around 1.3% of Polish words contain legitimate double letters (*nn*, *dd*, *ii*, etc.) but to the best of the author's knowledge, the dereplication never changes a benign word form into a toxic one or vice versa.

```
def dereplicate(s):
  """Removes repeated characters from s.

  >>> dereplicate('córrreczkęę')
  'córeczkę'
  >>> dereplicate('inną')
  'iną'
  """
  return ''.join(ch for ch, _ in itertools.groupby(s))
```

Figure 1: The `dereplicate()` function

[2] https://www.python.org/
[3] https://golang.org/
[4] https://github.com/morfologik/polimorfologik

Another frequent feature of online writing is replacing Polish letters with diacritics (*ą*, *ć*, *ę*, *ł*, etc.) with basic letters (*a*, *c*, *e*, *l*, etc.). Figure 2 shows the second step of model building: a partial or full removal of diacritics by passing the `D0` dictionary to the `dediacritize()` function, and additionally simulating creative misspellings by passing the `D1` dictionary. This step may conflate benign and toxic word forms, but they can still be assigned the correct toxicity based on the number of letters changed.

```
D0 = dict(zip('ąćęłńóśźż', 'acelnoszz'))
D1 = dict(zip('ąćęłńóśźżjk', 'acelnszzyq'))
D1.update({'ó': 'ou', 'u': 'oó', 'w': 'fv'})

def dediacritize(s, d):
  """Yields copies of s preserving/removing diacritics.

  >>> list(dediacritize('córeczkę', D0))
  ['córeczkę', 'coreczkę', 'córeczke', 'coreczke']
  """
  if not s: yield ''; return
  for tail in dediacritize(s[1:], d):
    yield s[0] + tail
    for head in d.get(s[0], ''):
      yield head + tail
```

Figure 2: The `dediacritize()` function

After the removal of diacritics, 0.006% of Polish word forms contain double letters (e.g. *puśćcie*, *ideę*, *weźże*, etc.). The `get_ngrams()` function that yields the character *N*-grams (Kimbrell 1988) of a word with sentinels at both ends fixes this by calling `dereplicate()` again, as shown in Figure 3. This function constitutes the third step of model building.

```
def get_ngrams(s, n):
  """Yields the n-grams of #s# w/o repeated characters.

  >>> list(get_ngrams('córeczkę', 5))
  ['#córe', 'córec', 'órecz', 'reczk', 'eczkę', 'czkę#']
  >>> list(get_ngrams('zzeram', 5))
  ['#zera', 'zeram', 'eram#']
  """
  s = '#' + dereplicate(s) + '#'
  for i in range(max(1, len(s) - n + 1)):
    yield s[i:i+n]
```

Figure 3: The `get_ngrams()` function

The fourth step of model building consists in appending the 5-grams of word forms to array X and appending their toxicity to array y, as shown in Figure 4. This function handles also a few creative misspellings.

```
def add_5grams(s, v, X, y):
  """Appends the 5-grams of #s# to X and v to y.

  Handles frequent/creative misspellings.
  """
  X.append(' '.join(ng for ng in get_ngrams(s, 5)))
  y.append(v)
  if 'qu' in s:
    add_5grams(s.replace('qu', 'q'), v, X, y)
  if re.search('ch[uóo][jy]', s):
    add_5grams(re.sub('ch([uóo][jy])', r'h\1', s), v, X, y)
```

Figure 4: The `add_5grams()` function

Figure 5 presents the fifth step of model building: executing steps 1–4 in sequence. For non-toxic words, the `D0` dictionary is passed to `dediacritize()`. For toxic words, the `D1` dictionary is passed.

```
X, y = [], []
for word, toxicity in data:
  lw = word.lower()
  rw = dereplicate(lw)
  for dw in dediacritize(rw, (D0, D1)[toxicity]):
    add_5grams(dw, toxicity, X, y)
```

Figure 5: The fifth step of model building

The sixth and last step of model building is shown in Figure 6.

```
model = sklearn.pipeline.Pipeline([
    ('count vectorizer',
     sklearn.feature_extraction.text.CountVectorizer()),
    ('logistic regression',
     sklearn.linear_model.LogisticRegression(
         penalty='l1', C=200, tol=1e-7)),
])
model.fit(X, y)
```

Figure 6: The sixth step of model building

Using the `scikit-learn` library[5], it builds a logistic regression model that predicts the toxicity of a word form from its character 5-grams, Using L_1 regularization makes the models sparse:

[5] `https://scikit-learn.org/`

most of irrelevant N-grams are outside them. The inverse regularization coefficient C is set to 200 to make the models accurate. The tolerance is set to a value smaller than the default to keep the models small.

Logistic regression converts any real number to a probability, i.e. a number between 0 and 1. For our purposes, it is enough to look at the sign of the total score of all 5-grams of a word. If the number is positive, the probability is greater than 0.5; if the number is negative, the probability is smaller than 0.5.

4. An Example

Given a text, Przetak returns an integer, whose following bits are set if at least one word form in the text is toxic:

— 1 for slurs

— 2 for abusive profanities

— 4 for asseverative profanities.

Depending on the use case, the caller may take into account only some of these bits.

A frequent issue with word filters is the so-called Scunthorpe problem[6]. Scunthorpe is a town in England with an unfortunate name that gets blocked by simple word filters. Figure 7 illustrates the fact that the 5-grams used in Przetak differ one from another enough to let it discern benign word forms from toxic ones.

`#zako`	0.00	0.00	0.00
`zakoc`	0.00	0.00	0.00
`akoch`	0.00	0.00	0.00
`kochu`	0.00	−10.82	0.00
`och*j`	0.00	+13.53	0.00
`ch*je`	0.00	+2.59	0.00
`h*je#`	0.00	−0.43	0.00
bias	−20.03	−14.46	−18.76
`#zakochuje#`	−20.03	−9.16	−18.76
	< 0	< 0	< 0
`#ch*j`	0.00	+19.32	0.00
`ch*je`	0.00	+2.59	0.00
`h*je#`	0.00	−0.43	0.00
bias	−20.03	−14.46	−18.76
`#ch*je#`	−20.03	+7.02	−18.76
	< 0	$\mathbf{> 0}$	< 0

Figure 7: Przetak's logistic regression on a benign and an abusive profane word form

[6] `https://en.wikipedia.org/wiki/Scunthorpe_problem`

5. Features of Przetak

Przetak is open-source software with the Apache 2.0 license[7]. It is available on GitHub as Go source code[8].

Przetak is resilient to various ways of circumventing word filters:

— replicating letters

— spacing out the words

— inserting non-letters between letters

— homograph spoofing, i.e. replacing letters with similar characters.

Also, thanks to the use of character 5-grams, it handles some frequent misspellings and out-of-vocabulary words composed of morphemes with an abusive or vulgar meaning.

The author chose Go as the implementation language because of its excellent support of Unicode, easy linking with programs written in other languages, and support for the major operating systems: Windows, macOS, and Linux. The source code consists of:

— 250 lines of real code

— 1 600 lines of character replacement arrays

— 12 800 lines of logistic regression coefficients

— 250 lines of tests.

Przetak is a dynamically-linked library. The source code repository contains examples of using it in Go and the following programming languages thanks to FFI (Foreign Function Interface): C, C++, Java, Lua, Node.js, Perl 5, Python 2 and 3, R, and Ruby. Programs written in languages that do not support FFI, like PHP, can call Przetak as a web service or a subprocess.

Acknowledgments

The author thanks Jan Szumiec for sharing his collection of sentences. The author is grateful to the staff of the Paquebot Mont-Royal café in Montreal for not kicking him out in the rain while he was writing this paper.

References

Kimbrell R. E. (1988). *Searching for Text? Send an N-Gram!* „Byte”, 5, p. 297–312.

Montagu A. (1967). *The Anatomy of Swearing*. MacMillan, New York.

[7] `https://www.apache.org/licenses/LICENSE-2.0`
[8] `https://github.com/mciura/przetak/`

Pinker S. (2007). *The Stuff of Thought — Language as a Window into Human Nature*. Penguin, London.

Ptaszyński M. and Masui F. (2018). *Automatic Cyberbullying Detection: Emerging Research and Opportunities*. IGI Global Publishing.

Waseem Z. and Hovy D. (2016). *Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter*. In *Proceedings of NAACL-HLT 2016*, pp. 88–93.

Approaching Automatic Cyberbullying Detection for Polish Tweets

Krzysztof Wróbel (Department of Computational Linguistics, Jagiellonian University)

Abstract

This paper presents contribution to PolEval 2019[1] automatic cyberbullying detection task. The goal of the task is to classify tweets as harmful or normal. Firstly, the data is preprocessed. Then two classifiers adjusted to the problem are tested: Flair and fastText. Flair utilizes character-based language models, which are evaluated using perplexity. Both classifiers obtained similar scores on test data.

Keywords

natural language processing, text classification, cyberbullying detection, Polish

1. Introduction

The problem of automatic cyberbullying detection for Polish have been introduced in PolEval 2019 contest. The goal of the task is to classify user-generated content as harmful or non-harmful. The cyberbullying problem has a real impact on people lives, even suicides were committed because of it.

Ptaszyński et al. (2018) evaluated a couple of cyberbullying detection systems. The best results were obtained by a proprietary system, but the second was fastText classifier beating many commercial systems.

In this paper, two classifiers adjusted to the problem are evaluated. The results were submitted to PolEval contest.

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

2. Data

Data provided by organizers consists of Polish tweets annotated as harmful and non-harmful (normal). In Subtask 2 harmful class is divided into cyberbullying and hate-speech. In training and test data, all user mentions are anonymized and shared tweets (beginning with `RT`) are truncated (the truncated tweets ends with ellipsis Unicode character). Last characters or words of tweets may carry important information for a classifier, e.g. emoticons.

Table 1 shows the distribution of classes in the training and test data. The number of harmful tweets is about 10 times smaller than normal tweets. Distribution of classes in training and test data does not match exactly.

Table 1: Distribution of classes in training and test data for Subtasks 1 and 2

Class	Subtask 1		Subtask 2	
	Training	Testing	Training	Testing
0	9190 (91.52%)	866 (86.60%)	9190 (91.52%)	866 (86.60%)
1	851 (8.48%)	134 (13.40%)	253 (2.52%)	25 (2.50%)
2	–	–	598 (5.96%)	109 (10.90%)

In order to employ a more suitable language model, new tweets were collected for 3 days using Twitter Streaming API. In comparison to the PolEval data, new tweets are full text and are not anonymized. The corpus (referenced later as the raw Twitter corpus) consists of 1.7 millions of tweets (164 MB of raw text).

3. Approach

Firstly, the data was preprocessed:

— frequent emojis were replaced to ASCII versions, e.g. smiling face was replaced to `:)`

— beginning retweet mark (`RT`) was removed

— escaped new line (`\n`) was replaced to space

— escaped quotation mark (`\"`) was unescaped

— encoded ampersand (`\u0026`) was replaced to ampersand (`&`).

For text classification two libraries were employed: Flair (Akbik et al. 2018) with addressed imbalance and fastText (Bojanowski et al. 2017, Joulin et al. 2017) using pretrained embeddings.

3.1. Flair

Flair generates contextual embeddings for a span of text (e.g. word) using character-based language models: forward and backward. Language models are trained on raw corpora. Table 2 shows character perplexity using language models trained on National Corpus of Polish (NKJP; Pęzik 2012), KGR10 (Kocoń and Gawor 2018), the raw Twitter corpus, and Common Crawl. The perplexity was calculated on training and test data on the original form and without anonymized mentions (`@anonymized_account`), and on a separate fragment of the raw Twitter corpus. The raw Twitter corpus was left unprocessed (i.e. not processed the same as the task data).

Table 2: Character perplexity of language models trained on different corpora tested on the original PolEval data, PolEval data without user mentions, and a fragment of the raw Twitter corpus

Corpus	Original data		Without anonymized mentions		Twitter corpus
	Training	Testing	Training	Testing	
NKJP	7 806	7 840	4 903	4 789	7 474
KGR10	**6 614**	**6 568**	**4 705**	**4 549**	6 870
Twitter	7 826	7 941	4 725	4 709	**3 396**
Common Crawl	11 820	12 025	6 678	6 714	10 564

Language model trained on Twitter corpus has a significantly lower perplexity score than a model trained on KGR10, probably because the raw Twitter corpus is too small. Different conclusions on Twitter corpus can be caused by a different method of obtaining tweets by organizers (e.g. filtered by some keywords).

Flair provides also text classifier using a neural network. Word embeddings are fed to a convolutional or recurrent neural network (used in this research).

Two approaches were taken to balance the training data. The first one is oversampling and the second is the usage of weights of classes.

3.2. FastText

FastText uses static word embeddings. Two fastText word embeddings were used:

— trained on KGR10 (Kocoń 2018, Kocoń and Gawor 2018)

— trained on NKJP for 100 epochs.

FastText provides also text classifier which is a linear classifier on averaged word embeddings.

4. Evaluation

The first subtask is evaluated by organizers using F1-score for harmful class and accuracy. The second subtask is evaluated using micro-average F1-score (microF) and macro-average F1-score (macroF). The macro-average F1-score is calculated as harmonic mean of macro-average Precision and Recall. In this paper macro-average F1 is calculated as the average of F1-scores of each class.

5. Experiments and Results

Flair classifier was trained using KGR10 language model with learning rate 0.1 and annealing factor 0.5. The model has a hidden state of size 128, embeddings are projected to 32, word dropout 0.2 and bidirectional LSTM. The training was stopped after 300 epochs or if the score was not improved by 5 epochs on validation data. The fastText classifier was trained using default parameters for 5 epochs.

A baseline for Subtask 1 classifies all data as harmful (class 1) and baseline for Subtask 2 labels all data as non-harmful (class 0).

Table 3 shows scores on training data using 5-fold stratified cross-validation. The folds preserve the percentage of samples for each class. The best result was obtained using fastText classifier trained on NKJP. For Flair oversampling was the best method for class imbalance. For Subtask 2, the baseline achieved a very high score.

Table 3: Results of fastText and Flair classifiers using 5-fold stratified cross-validation on training data

Corpus	**Subtask 1**		**Subtask 2**	
	F1	**Accuracy**	**Micro-average F1**	**Macro-average F1**
fastText NKJP	50.98	93.78	92.58	53.18
fastText KGR10	40.45	92.09	90.79	54.19
Flair	41.87	91.25	90.69	49.95
Flair with weights	40.20	90.95	91.14	40.04
Flair with oversampling	43.54	91.76	91.29	53.00
Baseline	15.63	8.48	91.53	31.86

Table 4 presents results on test data compared with the best systems in PolEval contest. Flair and fastText classifiers achieve similar scores. Pretrained fastText embeddings have influence on F1 score in Subtask 1, but they do not affect micro-average F1 in Subtask 2.

As a final step to Subtask 1, optimization of output class probability with F1 as the objective was performed. 20% of training data was used as validation data for the optimization and the rest was used to train the fastText classifier. The procedure was repeated 10 times and majority voting was used to generate final scores. This result was sent to Subtask 1 named

Table 4: Results of fastText and Flair classifiers compared with baseline and the best systems in PolEval contest. Bolded results were submitted to PolEval.

Corpus	Subtask 1		Subtask 2	
	F1	Accuracy	Micro-average F1	Macro-average F1
fastText	15.89	87.30	86.70	32.19
fastText NKJP	31.64	87.90	**86.80**	**44.04**
fastText KGR10	33.17	86.70	85.10	39.99
Flair	32.10	87.32	86.56	42.06
Flair with weights	34.03	87.26	86.22	33.00
Flair with oversampling	32.48	87.22	86.46	41.79
fastText NKJP optimized	**41.35**	**87.80**	–	–
Baseline	23.63	13.40	86.60	30.94
Best PolEval system	58.58	90.10	87.60	–

`fasttext`. The result obtained using fastText model trained on NKJP was sent to Subtask 2 named `fasttext`.

6. Conclusions

Both approaches did not achieve comparable results with the best systems in the PolEval contest. The cyberbullying detection problem is very complex as the results for Subtask 2 shows in comparison to the simple baseline.

Future works can focus on transfer learning from similar tasks, e.g. sentiment analysis. The larger raw dataset of tweets would be helpful to train language model. For English, Godin et al. (2015) shared word embeddings trained on 400 million tweets. Automatic labeling can be used to obtain more training data, e.g. by matching tweets with vulgarisms in the vocative case. Training data can be augmented by machine translation into another language and back to Polish.

Acknowledgments

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.

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.

Godin F., Vandersmissen B., De Neve W. and Van de Walle R. (2015). *Multimedia Lab @ ACL WNUT NER Shared Task: Named Entity Recognition for Twitter Microposts using Distributed Word Representations*. In *Proceedings of the Workshop on Noisy User-generated Text*, pp. 146–153.

Joulin A., Grave E., Bojanowski P. and Mikolov T. (2017). *Bag of Tricks for Efficient Text Classification*. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pp. 427–431. Association for Computational Linguistics.

Kocoń J. and Gawor M. (2018). *Evaluating KGR10 Polish Word Embeddings in the Recognition of Temporal Expressions Using BiLSTM-CRF*. „Schedae Informaticae", 2018(27).

Kocoń J. (2018). *KGR10 FastText polish word embeddings*. CLARIN-PL digital repository.

Pęzik P. (2012). *Wyszukiwarka PELCRA dla danych NKJP*. In Przepiórkowski A., Bańko M., Górski R. and Lewandowska-Tomaszczyk B. (eds.), *Narodowy Korpus Języka Polskiego*, pp. 253–279. Wydawnictwo Naukowe PWN.

Ptaszyński M., Leliwa G., Piech M. and Smywiński-Pohl A. (2018). *Cyberbullying Detection – Technical Report 2/2018, Department of Computer Science AGH, University of Science and Technology*. „CoRR", abs/1808.00926.

Exploiting Unsupervised Pre-training and Automated Feature Engineering for Low-Resource Hate Speech Detection in Polish

Renard Korzeniowski, Rafał Rolczyński, Przemysław Sadownik, Tomasz Korbak, Marcin Możejko (Sigmoidal)

Abstract

This paper presents our contribution to PolEval 2019 Task 6: Hate speech and bullying detection. We describe three parallel approaches that we followed: fine-tuning a pre-trained ULMFiT model to our classification task, fine-tuning a pre-trained BERT model to our classification task, and using the TPOT library to find the optimal pipeline. We present results achieved by these three tools and review their advantages and disadvantages in terms of user experience. Our team placed second in subtask 2 with a shallow model found by TPOT: a logistic regression classifier with non-trivial feature engineering.

Keywords

natural language processing, transfer learning, autoML, BERT, FastAI, TPOT

1. Introduction

This paper presents our contribution to PolEval 2019 Task 6: Hate speech and bullying detection[1]. In the following three sections we describe three parallel approaches that we followed: fine-tuning a pre-trained ULMFiT model to our classification task, fine-tuning a pre-trained BERT model to our classification task, and using a TPOT solution to find the optimal pipeline. There instantiate two important trends in modern machine learning: automated machine learning (autoML) and transfer learning. AutoML and transfer learning are of particular importance for industry practitioners who struggle with data scarcity and development time constraints. We describe results achieved by these three tools and them from a machine learning engineer point of view, highlighting advantages and disadvantages in terms of user experience.

[1] `http://poleval.pl/tasks/task6`

Our team placed second with a pipeline consisting of count-vectorizing the documents, recursive feature elimination guided by an extra-tree classifier with Gini criterion and dual logistic regression with L2 regularization. Our official results were micro-average F1 score 87.10% and macro-average F1 score 46.45%.

2. TPOT

2.1. Introduction

Tree-based Pipeline Optimization Tool (Olson et al. 2016a,b) is an autoML solution, which uses evolutionary algorithms to design tree-shaped machine learning pipelines based on operators defined in the `scikit-learn` library (Pedregosa et al. 2011). Before passing data to `TPOT`, we transformed the sentences using scikit-learn's `CountVectorizer`. We ran the `TPOT` with sparse configuration.

2.2. First Subtask: Binary Classification

For the first task we left `TPOT` for about 9 hours. We discovered that the time processed is less important than what parameters were applied. We only tried accuracy for the fitness score. The results achieved are presented in Table 1.

Table 1: Metrics for `TPOT` in the first subtask

Metric	**Value**
Precision	30.65%
Recall	56.72%
Balanced F-score	39.79%
Accuracy	77.00%

The best pipeline consisted of count-vectorizing the documents, recursive feature elimination guided by an extra-tree classifier with Gini criterion and dual logistic regression with l2 regularization.

To improve balanced F-score, we manually lowered the decision threshold to 0.7%.

2.3. Second Subtask: Multiclass Classification

In the second task, after about 17 minutes of computation on a multi-threaded machine, our solution achieved results presented in Table 2.

After the competition, we verified that it could achieve these and higher scores reliably. As a fitness function, we tried `f1_micro, f1_macro` and accuracy scores, which turned out to have a little difference in the outcome.

Table 2: Metrics for TPOT in the second subtask

Metric	Value
Micro-Average F-score	87.10%
Macro-Average F-score	46.45%

So far we have noticed, that the method consistently chose either an SVM or a logistic regression with an optional pre-processing step. After several hours, the best solution we managed to achieve consisted of count-vectorization of the documents, selection of the best 6% features, one-hot encoding and logistic regression with the L2 penalty and 0.05 regularization strength as the final classification model. The results are presented in Table 3.

Table 3: Our post-contest experiments

Metric	Value
Micro-Average F-score	87.60%
Macro-Average F-score	50.20%

2.4. Transfer

Training best pipeline found in the second task on binary labels resulted in metrics displayed in Table 4.

Table 4: The final model

Metric	Value
Precision	32.58%
Recall	64.18%
Balanced F-score	43.22%
Accuracy	77.40%

2.5. Summary

TPOT seems to find solutions reasonably fast for this kind of tasks. We assume that since all other solutions performed comparably well in the second task, we could achieve the level of the irreducible error for it. We also observed that it might provide significant improvement when ensembled with other methods including neural-network based models described in further sections.

2.6. Ease of Use

Out of all the solutions we have tried, in our opinion, TPOT was the easiest one to use. However one should note that with default values for the evolutionary algorithm (with 100 generations, population size of 100 and maximum evaluation time of single individual of 5 minutes), the overall optimization process can be quite long.

3. ULMFiT

3.1. Introduction

The main ULMFiT (Howard and Ruder 2018) schema consists of:

- training language model on a huge dataset,
- fine-tuning language model to a smaller task-specific dataset,
- using language model trained on classification data to improve its understanding of input text during classification.

This idea makes extensive usage of two major Machine Learning concepts, namely: transfer learning which has proved successful in computer vision (Sharif Razavian et al. 2014) and semi-supervised learning (Peters et al. 2017).

The aim of this section is to determine how fitting a pre-trained language model on small task-specific dataset influences the performance of the classifier build on top of the pre-trained model and using it as an encoder.

3.2. Architecture

The model architecture used for experiments was AWD_LSTM (Merity et al. 2017). This model architecture has a strong, built-in regularization in the form of smartly applied dropout. Because of that, we have found it as a good candidate for transfer learning base that involved fitting pre-trained model on a small dataset.

3.3. Training Procedure

In the following subsection, we describe the procedure used to preprocess the data and train our ULMFiT solution. Two datasets were needed to train language model: a huge unlabelled corpus to teach a Polish language model and a smaller one for classification fine-tuning. In our case, the smaller dataset was just the training set provided by the competition organizers and as the huge language corpus we have used the last year's PolEval (Ogrodniczuk and Kobyliński 2018) dataset for language modeling task[2].

[2] http://2018.poleval.pl/task3/task3_train.txt.gz

Tokenization and general preprocessing of our corpora was performed using a popular NLP tool SpaCy (Honnibal and Montani 2019).

We have used one fit cycle policy (Smith 2018, Smith and Topin 2017, Smith 2015) to train both language model and classifier since it increased the test set accuracy and lowered the time of training.

3.4. Results

Metrics used in the competition depended on the task. Our ULMFiT solution was used only in the first task of binary classification, and we trained "Without fine-tuning" solution. Second result "With fine-tuning" in Table 5 below is the submission of winning team n-waves.

Table 5: Binary classification

Fine-tuning	Precision	Recall	F1	Accuracy
With	66.67	52.24	58.58	90.10
Without	52.71	50.75	51.71	87.30

As we can see fine-tuning had a significant impact on the final performance of the model classifier.

3.5. Accessibility

We used FastAI implementation of ULMFiT. It was user-friendly and easy to use. The only problems we encountered were occasional unexpected errors connected to memory management when training a language model on a huge unlabeled corpus.

4. BERT

One of the latest milestones in NLP development was the release of BERT (Bidirectional Encoder Representations from Transformers Devlin et al. 2018). BERT is an unsupervised, deeply bidirectional system for language model pre-training. It has become a state of the art method for many NLP tasks. The models, which were pre-trained in an unsupervised manner on massive plain text corpus, are available for download and free reuse. Thanks to that BERT might serve as a provider of bidirectional contextual words representations which can be easily reused in any language-processing pipeline.

4.1. Binary Classifier

The most straight-forward way to use BERT representations is to apply them for the classification task. In order to do that, we have fine-tuned the classifier with minimal changes applied to the BERT model architecture during the training phase (the process is performed in a manner similar to Semi-supervised Sequence Learning and ULMFiT fine-tuning process). In our experiments we have used the BERT-Base Multilingual Cased model as a classification base. This pre-trained model supports 104 languages (including Polish) and has over 110M parameters.[3] It consists of a trained Transformer Encoder stack with 12 layers (the size of hidden state is 768) and Multi-Head Attention (12-heads) (Vaswani et al. 2017, Devlin et al. 2018). At the top of the base model, we have added the classification softmax layer.[4] The hyper-parameters of the fine-tuning process are presented in Table 6 (the selected model).

Table 6: Hyper-parameters used in the fine-tuning process

Batch size	Learning rate	Epochs	Warm-up	Max sequence
32	$2 \cdot 10^{-5}$	3	0.1	128

4.2. Results

The results of our best model are presented in Table 7. In our experiment we have used BERT *mean-pooled output* of the last hidden layer which assigns a single vector to an entire sequence. In the future, we plan to experiment with several different pooling strategies, e.g. six choices examined by Devlin et al. (2018).

Table 7: BERT: Binary Classification

Metric	Value
Precision	65.12%
Recall	42.31%
Balanced F-score	**51.32%**
Accuracy	93.23%

BERT is a powerful component which can be used effectively in different types of tasks. Unfortunately, the largest version of the model, which currently is reported to achieve the state of the art results is ridiculously large (340M parameters) and it is unavailable for the multilingual use case. It is also worth to point out that it is currently impossible to reproduce most of the BERT results using GPU machine due to the out-of-memory issues what could be a severe limitation in everyday applications.

[3] The model is case-sensitive and performs much better than the uncased one.
[4] Add dropout layer with ratio 0.1 helps to prevent high overfitting.

5. Conclusions

We arrived at a slightly surprising result that it was a shallow model produced by TPOT – the easiest to use library we tried – that earned us the second place in subtask two. Even if the reliability of these results can be questioned, it still proves a strong case in favor of proper exploitation of the powers of shallow models when training data are limited.

References

Devlin J., Chang M., Lee K. and Toutanova K. (2018). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. „CoRR", abs/1810.04805.

Honnibal M. and Montani I. (2019). *spaCy 2: Natural Language Understanding with Bloom Embeddings, Convolutional Neural Networks and Incremental Parsing*. To appear.

Howard J. and Ruder S. (2018). *Universal Language Model Fine-tuning for Text Classification*. „CoRR", abs/1801.06146.

Merity S., Shirish Keskar N. and Socher R. (2017). *Regularizing and Optimizing LSTM Language Models*. „CoRR", abs/1708.02182.

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

Olson R. S., Urbanowicz R. J., Andrews P. C., Lavender N. A., Kidd L. C. and Moore J. H. (2016a). *Automating Biomedical Data Science Through Tree-based Pipeline Optimization*. In Squillero G. and Burelli P. (eds.), *Proceedings of the 19th European Conference on Applications of Evolutionary Computation (EvoApplications 2016): Part I*, pp. 123–137. Springer International Publishing.

Olson R. S., Bartley N., Urbanowicz R. J. and Moore J. H. (2016b). *Evaluation of a Tree-based Pipeline Optimization Tool for Automating Data Science*. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, GECCO '16, pp. 485–492, New York, NY, USA. ACM.

Pedregosa F., Varoquaux G., Gramfort A., Michel V., Thirion B., Grisel O., Blondel M., Prettenhofer P., Weiss R., Dubourg V., Vanderplas J., Passos A., Cournapeau D., Brucher M., Perrot M. and Duchesnay E. (2011). *Scikit-learn: Machine Learning in Python*. „Journal of Machine Learning Research", 12, p. 2825–2830.

Peters M. E., Ammar W., Bhagavatula C. and Power R. (2017). *Semi-Supervised Sequence Tagging with Bidirectional Language Models*. „CoRR", abs/1705.00108.

Sharif Razavian A., Azizpour H., Sullivan J. and Carlsson S. (2014). *CNN Features Off-the-shelf: An Astounding Baseline for Recognition*. „CoRR", abs/1403.6382.

Smith L. N. (2015). *Cyclical Learning Rates for Training Neural Networks*. „CoRR", abs/1506.01186.

Smith L. N. (2018). *A Disciplined Approach to Neural Network Hyper-parameters: Part 1 – Learning Rate, Batch Size, Momentum, and Weight Decay*. „CoRR", abs/1803.09820.

Smith L. N. and Topin N. (2017). *Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates*. „CoRR", abs/1708.07120.

Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser Ł. and Polosukhin I. (2017). *Attention Is All You Need*. In *Advances in Neural Information Processing Systems*, pp. 5998–6008.

Universal Language Model Fine-Tuning for Polish Hate Speech Detection

Piotr Czapla (n-waves), **Sylvain Gugger** (fast.ai), **Jeremy Howard** (fast.ai), **Marcin Kardas** (n-waves)

Abstract

Transfer learning in NLP allows for using large amounts of unlabelled text in unsupervised manner to dramatically reduce data necessary for a target task. However, the high performance of model on the source task does not indicate whether the final model will perform well on the target task. Our experiments with Universal Language for Fine-tuning (Howard and Ruder 2018) architecture run on PolEval 2019 Harmful Speech Detection task show that initial weights of language model play an important role in model performance on the target task. Interestingly, the language model's perplexity was not affected by the initial weights and in both studied cases the models performed equally well on the source task even though the performance differ significantly for the target task. We propose a simple mechanism to test if the sampled initial weights are well suited for the target task.

Finally, we present our solution for Harmful Speech Detection that achieves state-of-the-art performance and took first place in Task 6.1 of the PolEval'19 competition. Our model and source code are publicly available.[1]

Keywords

natural language processing, document classification, offensive speech detection

1. Introduction

Offensive speech is a growing problem on the Internet, amplified by the use of social media. According to Wirtualne Media (2018) in February 2018 there were 4.61 million active Polish Twitter users, which constituted 16.51% of all Polish Internet users. 4.52% were under 15 years old. A popular approach of automatic detection of offensive speech is to use a curated

[1] `https://n-waves/ulmfit-multilingual`

list of forbidden words. The method often is ineffective at detecting instances of direct insults, cyber-bullying, or hate speech.

Recent work that uses language modeling as a source for transfer learning to classification tasks makes it possible to achieve higher performance than previously known transfer learning techniques. We present an extension to the Howard and Ruder (2018) ULMFiT architecture adapted to the morphological rich languages using subword tokenization (Kudo 2018), that let us win the first place on Task 6.1 of PolEval 2019 competition with an F1 score of 58.6%. The result were further improved during ablation studies and our best performing model achieves 62% F1 score.

We show how selection of the pretraining dataset is key to the good performance. Our ablation studies suggest that perplexity of the language model does not provide a strong indication of performance on downstream tasks. We show evidence that fine-tuning is ineffective to combat bad luck during initialization of language model weights, and the difference in performance between two initialization does not change when the pretraining dataset is changed. We propose an alternative way to quickly measure the applicability of the drawn weights on the downstream task. The method requires further testing. Our results align with the recent findings of Frankle and Carbin (2018) that highlight the importance of the weights drawn during initialization.

2. Related Work

2.1. Pretrained Language Models

Pretrained language models based on an LSTM (Howard and Ruder 2018) and a Transformer (Devlin et al. 2018, Radford et al. 2019) have been proposed. Howard and Ruder (2018) used an English Wikipedia as a pretraining corpus to achieve state-of-the-art accuracy on several sentiment analysis datasets. Recent work by Peters et al. (2018) suggests that—all else being equal—an LSTM outperforms the Transformer in terms of downstream performance. For this reason, we use LSTM as our language model.

2.2. The Importance of Initialization

The importance of the initial connections and the numbers returned by the random generator were mentioned previously by Frankle and Carbin (2018), Zhou et al. (2019). Zhang et al. (2019) also show that the upper layers of neural networks do not change much from their initial random weights. All of these findings inclined us to pretrain multiple language models. Our study confirms the importance of luck during initialization of the weights. We show that two sets of weights can have similar perplexity, but one will perform significantly better on the downstream classification task. This relation holds even when the underlying text corpus, tokenization and the order of training examples are changed.

Tanti et al. (2019) experimented with transfer learning for image caption generation. Similar to our findings, they noticed that the best language models (in terms of perplexity) do not result in the best caption generators after transfer learning.

2.3. Subword Tokenization for Polish

Due to rich morphology of Polish language word-based models require much larger vocabularies and training data compared to English. This is why it is more common for such languages to use a subword tokenization. Czapla et al. (2018) used ULMFiT with subword tokenization for Polish language modelling achieving state-of-the-art perplexity on PolEval'18 LM corpus. The model with vocabulary consisting of 25K subword tokens was able to generalize conjugation and declension forms for words in new contexts that were not present in training corpus.

3. Experiments

Our solution uses Universal Language Model for Fine-tuning ULMFiT (Howard and Ruder 2018) with Sentence Piece tokenization, as in (Czapla et al. 2018). We use ULMFiT implementation from fast.ai library (Howard et al. 2019). It was pretrained using the Polish language part of reddit.com. We use weighted binary cross entropy as a loss function to handle class imbalance, and early stopping to minimize overfitting. In ablation studies we show that all of these decisions except for early stopping were critical to achieving good performance on the test set.

3.1. Weights of Language Model

The specific instance of weights has a significant impact on the performance of the downstream task; the relation holds even when other aspects of the training varies. We noticed this when training 4 ULMFiT models with weights sampled from random generator initialized with seed 0 and 1 for the Wikipedia and reddit pretraining corpuses. These pretrained models were then used to train 298 classification models that differed from each other in weights for classification heads, tokenization (different SentencePiece models), the number of the fine-tuning epochs (0, 6 and 20 epochs) on the PolEval dataset and the order of training examples. In every subset of the experiments, the seed 0 under-performed on the test set compared to the seed 1. The Table 1 and the histogram in Figure 1 show statistics across all classification models with respect to the initial seed used to initialize language model weights. This observation aligns with the recent work describing the importance of the model initialization (Frankle and Carbin 2018, Zhang et al. 2019).

The difference in the performance can be observed even when the language model was pretrained only for one epoch (instead of 10), see histogram in Figure 2. This suggests a quick way to search for the optimal weights of a language model for a particular task. Our experiments were done only on two sets of weights, and the validation set used in early stopping had training set distribution that was significantly different from test distribution

Table 1: Table showing how seed 0 consistently under-performs compared to seed 0

	10 epochs of training on reddit & wiki			1 epoch of training on wiki		
	Seed 1	**Seed 0**	**Diff**	**Seed 1**	**Seed 0**	**Diff**
Count	151	147		10	10	
Mean	0.555511	0.539647	0.015865	0.525926	0.469891	0.056035
Std	0.028428	0.033603	-0.005174	0.024596	0.038780	-0.014185
Min	0.488479	0.451613	0.036866	0.495798	0.394231	0.101568
25%	0.536181	0.515420	0.020761	0.504658	0.443662	0.060996
50%	0.558333	0.541485	0.016849	0.526767	0.478060	0.048707
75%	0.576201	0.563492	0.012709	0.539330	0.490383	0.048947
Max	0.622222	0.614232	0.007990	0.564885	0.523809	0.041076

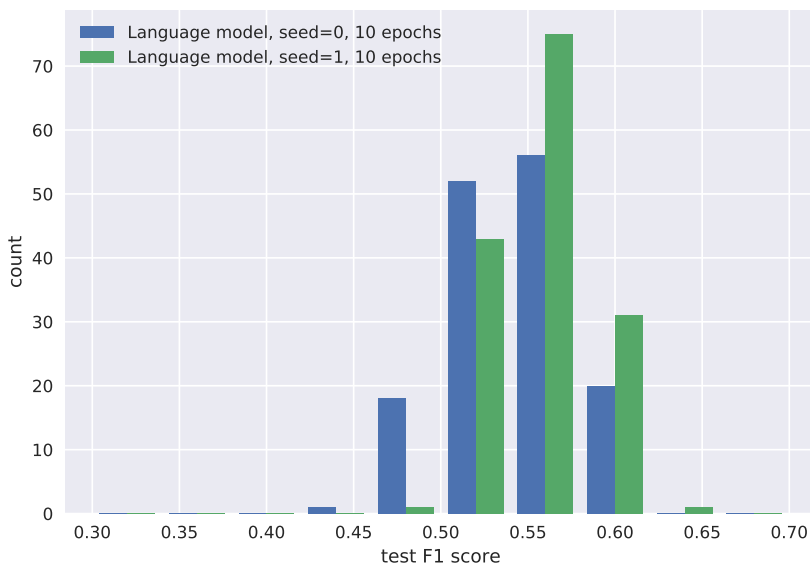

Figure 1: Comparison of distribution of F1 score on test set with seed=1 (green) and seed=0 (blue) for all experiments

which makes this results inconclusive but promising. We hypothesize that if this phenomenon is consistent, it may explain why larger models such as BERT (Devlin et al. 2018) underperform on classification tasks compared to (Howard and Ruder 2018), as such models are only fine-tuned for each classification task without new random initialization and pretraining which might be important for specific tasks.

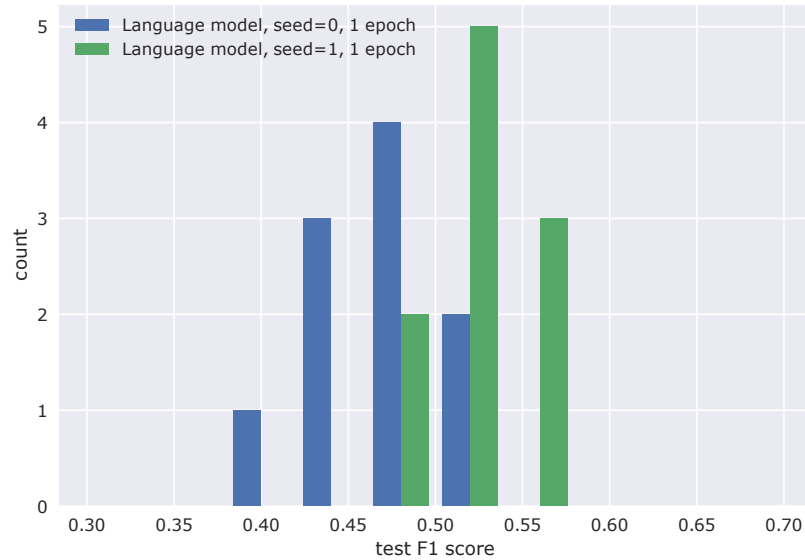

Figure 2: Comparison of distribution of F1 score on test set of classifiers based on two language models, initialized with seed=1 (green) and seed=0 (blue), that were pretrained for 1 epoch

3.2. Datasets

Harmful speech dataset

The dataset consists of 10K tweets in the training set and 1K tweets in the test set, all labelled either as harmful or non-harmful. The training dataset was used for unsupervised fine-tuning of language models. The test set had 13.4% of harmful tweets which is more than the training set and less retweets compared to the training set.

Reddit comments

We used Google BigQuery and a public dataset[2] of comments from Reddit, a social media platform, to extract comments from all subreddits marked as Polish. According to OpenNLP language detector, 67% of obtained comments use Polish and 23% use English. The reddit dataset is preprocessed with the default fastai.text (v1.0.51) transformations.

Wikipedia

The wiki dataset was downloaded from the Mediawiki dumps. It was pre-tokenized using Moses Tokenizer for consistency with WikiText-103 (Merity et al. 2016) and transformed using fastai.text (v1.0.51) transformations (see Howard et al. 2019).

[2] `https://bigquery.cloud.google.com/table/fh-bigquery:reddit_comments.2015_05`

3.3. Architecture

We use Universal Language Model for Fine-Tuning (Howard and Ruder 2018) with hyperparameters as presented in Table 2.

Table 2: Details of our submission

Language model	vocabulary size	25 K
	RNN type	LSTM
	recurrent layers	4
	embeddings dimension	400
	hidden state dimension	1150
	training time	12 epochs
	peak learning rate	0.01
	batch size	160
	BPTT	70
	data set	reddit comments
Fine-tuning	training time	6 epochs
	dropout	no
	peak learning rate	0.001
	batch size	128
Classifier	training time	8 epochs
	loss	weighted cross entropy
	dropout	0.1
	linear layers	2
	batch size	320
Results	precision	66.67%
	recall	52.24%
	F1 score	58.58%
	accuracy	90.10%

Tokenization and preprocessing

We used sentence piece unigram model (Kudo 2018) for tokenization, following architecture described by Czapla et al. (2018). The unigram model was trained on the language model pretraining corpus with 25K subword tokens limit, including 100% characters in the corpus alphabet. We do not use subword regularization during training or inference. The goal of preprocessing step was mainly to normalize texts between language model training corpus and tweets, as well as to remove parts that we considered noise (links, user names, numbers). We also replaced emojis and some emoticons with their descriptions taken from The Unicode Consortium and Wikipedia's list of emoticons. We removed duplicated tweets in an attempt to make the training and validation sets independent.

Pretraining and fine-tuning

Our models were pretrained for 10–12 epochs on our reddit dataset. The training is relatively quick and takes only 4 hours to complete on a single GPU, which allowed us to experiment with different modifications to the architecture. The sentence piece tokenization model is trained on the first dataset and it is left unchanged during the fine-tuning and classification. It is one of the reasons why we used reddit instead of Wikipedia. The corpus was close enough to the PolEval dataset that the fine-tuning step was not necessary, and both models with and without fine-tuning performed well. On the other hand, language models trained on Wikipedia during ablation studies performed worse without fine-tuning. See Table 3 for more details.

Table 3: Performance of models with and without fine-tuning

Dataset	fine-tuning	mean	std	max	75%
wiki	no	0.522515	0.026661	0.582781	0.541998
wiki	yes	0.536890	0.030744	0.608392	0.558897
reddit	yes	0.561675	0.027365	0.622222	0.581680
reddit	no	0.573931	0.016832	0.603390	0.581451

Classifier

As shown in Table 4 the datasets are highly unbalanced. To mitigate the fact we used weighted binary cross entropy as a loss function:

$$L(y, \hat{y}) = -\frac{1}{m}\sum_{i=1}^{m}(30 y_i \ln \hat{y}_i + 0.5(1-y_i)\ln(1-\hat{y}_i)),$$

where m is a mini-batch size, y_i is a true label of the i-th training example and $\hat{y}_i$ is model's prediction.

Table 4: Summary of PolEval 2019 Harmful Speech datasets; in deduplication tweet and retweet are considered the same

Dataset	Tweets	Harmful	Tokens/tweet
Train	10 041	8.48%	12.40
Train (dedup.)	9 400	8.05%	12.20
Test	1 000	13.40%	12.20
Test (dedup.)	946	12.79%	12.00

3.4. Submission

We selected our model for submission by looking at the F1 score on a validation set of 10% of the training set. The model was one of the first that we trained during competition. The later models were exploring a number of alternative methods to improve our accuracy, including use of an English hate speech corpus to train multilingual models, such as Laser by Artetxe and Schwenk (2018). The attempts where not successful. During ablation studies we noticed that ULMFiT has high variability in performance, depending on the weight initialization of both the classifier and the language model. To draw meaningful conclusions we trained around 500 classifiers. Some of them had much better performance. Unfortunately we noticed that the F1 performance on our validation set is slightly negatively correlated with the performance on the test set. We performed a number of experiments in order to align the validation set with test set. The only successful attempt that gave us a positive correlation was using half of the test set as the validation set. Unfortunately, this makes the selection of further models impossible without risking over-fitting to the test set. As shown in Table 4 the training and test sets have significantly different fraction of tweets labelled as harmful. It could be simply a result of increased hate speech rate during the time the test data was acquired. However, the difference in performance between validation and test sets in our experiments suggest that there might be a mismatch between distributions of labels, e.g., due to different sensitivity of annotators annotating each dataset.

4. Ablation Studies

Our architecture have high variance of results between runs, even with all hyper parameters fixed. In order to mitigate the issue during our experiments we forced all executions to be deterministic. We fix seed values at 4 stages of our pipeline:

— at the beginning of pretraining, before language model weights are sampled

— at the beginning of fine-tuning, to fix the order in which tweets are shuffled

— at the beginning of classifier initialization, before classifier weights are sampled

— at the beginning of classifier training, to fix the order in which tweets are shuffled.

For each experiment we used at least two pretrained language models, and trained 10 classification models for each model. Our results of the ablation studies are presented below in Table 5. We found that increasing dropout does not improve the performance of the classifiers. The weighted cross entropy was crucial to achieve good results. Without weights the best results are worse than the average result trained with weighted cross-entropy. Early stopping was not necessary for the language model with seed 1 but was crucial for the language model with seed 0. Table 3 shows the summary of the experiment we executed in order to see if the fine-tuning was necessary to achieve good performance on the classification task. We fine-tuned the all 4 language models trained on reddit and wikipedia for 0 epochs (i.e. no finetune) 6 and 20. The finetuning was not necessary for reddit to achieve good performance but was crucial for language models pretrained on wikipedia.

Table 5: Summary of ablation studies

Dataset	exp_type	lmseed	mean	std	max	75%
reddit	dropmul = 0.5	0	0.519352	0.030414	0.589552	0.533757
reddit	dropmul = 0.5	1	0.538298	0.030352	0.602151	0.557069
wiki	1 epoch pretraining	0	0.469891	0.038780	0.523809	0.490383
wiki	1 epoch pretraining	1	0.525926	0.024596	0.564885	0.539330
wiki	cross entropy w/o weights	0	0.433285	0.062494	0.521739	0.487437
wiki	cross entropy w/o weights	1	0.451950	0.050503	0.539535	0.490566
wiki	w/o early stopping	0	0.516319	0.033725	0.564315	0.546150
wiki	w/o early stopping	1	0.564405	0.019395	0.608392	0.574534
wiki	our model	0	0.523124	0.027906	0.570470	0.540592
wiki	our model	1	0.550656	0.027349	0.608392	0.573604
reddit	our model	0	0.553660	0.029906	0.614232	0.575757
reddit	our model	1	0.560869	0.028504	**0.622222**	**0.580522**

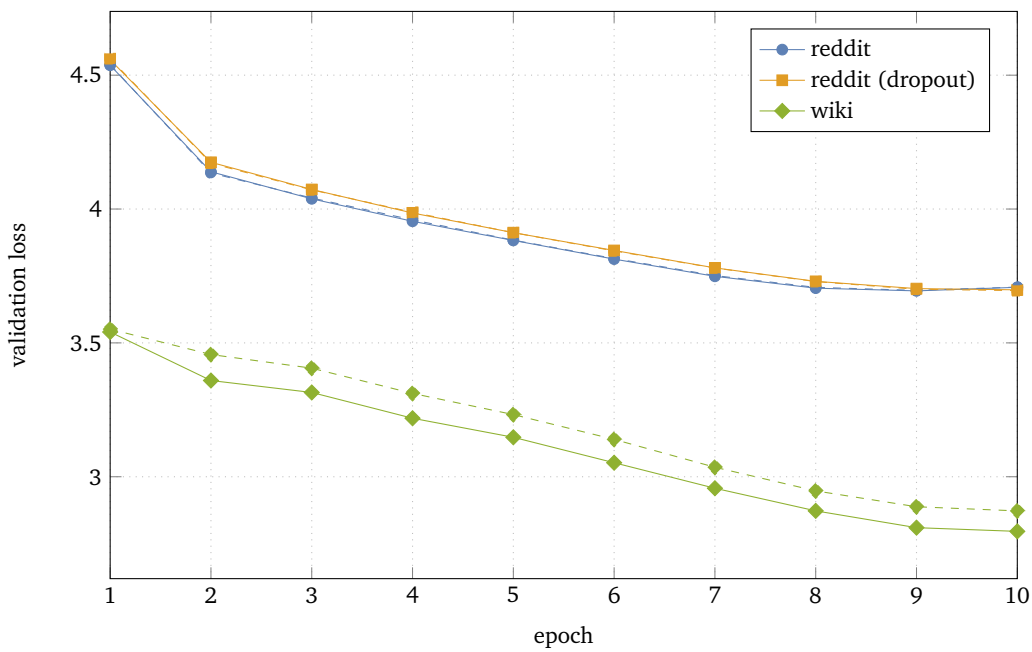

Figure 3: Validation loss of language models. Each setting was trained twice with seed values: 0 (solid lines) and 1 (dashed lines). The models with seed value 1 performed better than the models with seed value 0. The models pretrained on Reddit were performing better than models pretrained on Wikipedia.

We further explored the difference between language model and different weight initialization, and noticed a possible reverse correlation between perplexity and the performance on the downstream task (see 3). However, 2 random initializations is not enough to draw conclusive results.

5. Final Remarks

In (Czapla et al. 2018) we showed that Universal Language Model for Fine-tuning complemented with subword tokenization achieves state-of-the-art perplexity in Polish language modelling. In this paper we presented experimental evidence that ULMFiT pretrained on Polish corpus can be successfully used for Polish documents classification.

It remained an open question whether high performance of ULMFiT on the language modelling task will translate to high performance on downstream tasks. Our experiments present, in accordance with findings from Tanti et al. (2019), evidence that this may not be the case. Therefore, to evaluate a model one is required to go through the whole iteration from pretraining language model through fine-tuning to training the model on the downstream task. We showed an alternative way of measuring the performance of sampled language model weights. The work is inconclusive but promising and should be further explored.

References

Artetxe M. and Schwenk H. (2018). *Massively Multilingual Sentence Embeddings for Zero-Shot Cross-Lingual Transfer and Beyond*. „CoRR", abs/1812.10464.

Czapla P., Howard J. and Kardas M. (2018). *Universal Language Model Fine-Tuning with Subword Tokenization for Polish*. In Ogrodniczuk M. and Łukasz Kobyliński (eds.), *Proceedings of PolEval 2018 Workshop*.

Devlin J., Chang M.-W., Lee K. and Toutanova K. (2018). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. „CoRR", abs/1810.04805.

Frankle J. and Carbin M. (2018). *The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks*. „CoRR", abs/1803.03635.

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

Howard J., Gugger S. et al. (2019). *fastai*. `https://github.com/fastai/fastai`.

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

Merity S., Xiong C., Bradbury J. and Socher R. (2016). *Pointer Sentinel Mixture Models*. „CoRR", abs/1609.07843.

Peters M., Neumann M., Zettlemoyer L. and Yih W.-t. (2018). *Dissecting Contextual Word Embeddings: Architecture and Representation*. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*, pp. 1499–1509, Brussels, Belgium. Association for Computational Linguistics.

Radford A., Wu J., Child R., Luan D., Amodei D. and Sutskever I. (2019). *Language Models are Unsupervised Multitask Learners*.

Tanti M., Gatt A. and Camilleri K. P. (2019). *Transfer Learning from Language Models to Image Caption Generators: Better Models May Not Transfer Better*. „CoRR", abs/1901.01216.

Wirtualne Media (2018). *Wśród polskich użytkowników Twittera przeważają mężczyźni, osoby z dużych miast i ze średnim lub wyższym wykształceniem (analiza)*.

Zhang C., Bengio S. and Singer Y. (2019). *Are All Layers Created Equal?* „CoRR", abs/1902.01996.

Zhou H., Lan J., Liu R. and Yosinski J. (2019). *Deconstructing Lottery Tickets: Zeros, Signs, and the Supermask*. „CoRR", abs/1905.01067v1.

A Simple Neural Network for Cyberbullying Detection

Katarzyna Krasnowska-Kieraś, Alina Wróblewska
(Institute of Computer Science, Polish Academy of Sciences)

1. Introduction

This abstract briefly presents a neural network solution that competed in Task 6 (Automatic cyberbullying detection) of PolEval 2019 evaluation campaign. The task at hand consisted in detecting harmful tweets and was divided into two subtasks:

1. Task 6-1: **Harmful vs non-harmful**. The goal was to binarily discriminate between tweets carrying a harmful content (cyberbullying, hate speech etc.) and non-harmful ones.
2. Task 6-2: **Type of harmfulness**. The goal was to further distinguish between two possible kinds of harmfulness contained in the tweets by means of 3-way classification (non-harmful, cyberbullying or hate-speech).

The source code and model used for the described PolEval 2019 submission is available at `http://mozart.ipipan.waw.pl/~kkrasnowska/PolEval2019/`.

2. Network Architecture

The models used for both subtasks were multilayer dense neural networks with several input vectors, encoding different features of each classified tweet (see below for feature description). Every input vector was first fed to a sub-network consisting of 2 consecutive dense layers of size 128. The role of the sub-networks was to enable learning any useful feature extraction from the input vectors. The outputs of the sub-networks were then concatenated and passed into 4 stacked dense layers of size 128. The top layer of the network was also a dense one, with a *softmax* activation function. It therefore predicted a probability distribution over the tasks' target classes, with its *argmax* being the final classification. The number and size of the dense layers were selected based upon cross-validation on the training dataset. All the

layers were trained jointly by optimising with respect to the cross-entropy loss function. The network was implemented using Keras[1] Python library with TensorFlow[2] backend.

3. Input Features

The following feature vectors were created for each tweet:

— **LASER embedding.** A vector of length 1024 generated using the pre-trained LASER[3] model (Artetxe and Schwenk 2018).

— **Morfeusz qualifiers.** A vector counting the number of segments in the tweet that received at least one "bad word" stylistic qualifier when analysed with Morfeusz (Woliński 2014). The vector consists of four numbers, representing the counts for *wulg.* (vulgar), *pot.* (colloquial), *pogard.* (contemptuous) and *lekcew.* (depreciating) qualifiers.

— **Vulgar/offensive words.** Vectors constructed using vocabularies of vulgar/offensive word forms. The coordinates of the vector correspond to particular entries in the vocabulary. Their values represent the number of occurrences of each form in the tweet. Two vocabularies of lengths 711 and 401 were used, resulting in two feature vectors of the same respective sizes, fed to separate sub-networks.

— **Character n-grams.** A vector constructed similarly to the previous ones, but using a vocabulary of character n-grams. The n-grams were extracted from training data based on a ranking giving preference to n-grams that (1) occur frequently in the tweets labeled as harmful (2) have a high ratio of frequency in the harmful tweets to frequency in the non-harmful ones. 60 n-grams for $n \in \{6, 7, 8, 9, 10\}$ were selected, resulting in a vocabulary (and feature vector) of size 300. The idea behind this feature vector was to account for spelling variations and derivational phenomena by finding matching subwords.

4. Training Data Expansion

The training data provided for both tasks was highly imbalanced in terms of target class: 91.5% of the training data was labeled as non-harmful. Therefore, a classifier labeling everything as the majority (negative) class constitutes a rather high baseline in terms of accuracy, at the same time performing extremely poorly in terms of precision, recall and F1 scores.[4] In order to mitigate the effect of data imbalance, the dataset was extended with artificially generated training instances of harmful tweets. Two types of generated data were used:

[1] `https://keras.io/`
[2] `https://www.tensorflow.org/`
[3] `https://github.com/facebookresearch/LASER`
[4] With recall score trivially equal 0, and precision undefined due to division by zero.

— **Chunk shuffling.** The tweets belonging to a class that needed more examples were simply chunked (using full stop as delimiter). New 'tweets', not present in the original dataset, were then generated by pairing random chunks, doubling the number of instances for the given class.

— **Translation.** Each original tweet from the expanded class was also used to create translation-based examples. The translations were performed with Google Translate. Two translation routes were employed, yielding two new instances: (1) Polish → Russian → Polish, (2) Polish → English → German → Polish. The aim of this procedure was to create paraphrased (or slightly distorted) versions of the original tweets.

5. Results

In the final evaluation, the systems for each subtask obtained the following results:

— **Task 6-1:** accuracy 86.90% (11th out of 14 competing systems; 3.6% below the best score), precision 51.90% (9th; 36.6% below best), recall 30.60%, (12th; 47.6% below best), F1 38.50% (10th; 43.41% below best).

— **Task 6-2:** micro-average F1 83.70% (5th out of 8 competing systems; 4.5% below the best score), macro-average F1 49.47% (2nd; 4.4% below best).

Acknowledgments

The presented work was funded by SONATA 8 grant no 2014/15/D/HS2/03486 from the Polish National Science Centre.

References

Artetxe M. and Schwenk H. (2018). *Massively Multilingual Sentence Embeddings for Zero-Shot Cross-Lingual Transfer and Beyond*. „CoRR", abs/1812.10464.

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 Ninth International Conference on Language Resources and Evaluation, LREC 2014*, pp. 1106–1111, Reykjavík, Iceland. ELRA.