Proceedings of the PolEval 2021 Workshop

Maciej Ogrodniczuk, Łukasz Kobyliński (eds.)

Institute of Computer Science, Polish Academy of Sciences
Warszawa, 2021

ISBN 978-83-63159-31-3

© Copyright by Institute of Computer Science, Polish Academy of Sciences
Jana Kazimierza 5
01-248 Warszawa

The publication is available under Creative Commons Attribution 4.0 International (CC BY 4.0) license. The licence text is available at https://creativecommons.org/licenses/by/4.0/deed.en.

Warszawa 2021

Contents

PolEval 2021: The Fifth Anniversary
Maciej Ogrodniczuk, Łukasz Kobyliński 5

Gonito Platform as the Evaluation Engine in PolEval 2021
Filip Graliński 11

PolEval 2021 Task 1: Punctuation Restoration from Read Text
Agnieszka Mikołajczyk, Adam Wawrzyński, Piotr Pęzik, Adam Kaczmarek, Michał Adamczyk, Wojciech Janowski 21

Punctuation Restoration with Transformers
Krzysztof Wróbel, Dmytro Zhylko 33

Comparison of Translation and Classification Approaches for Punctuation Recovery
Norbert Ropiak, Michał Pogoda, Jarema Radom, Karol Gawron, Michał Swędrowski, Bartłomiej Bojanowski 39

Punctuation Restoration with Ensemble of Neural Network Classifier and Pre-trained Transformers
Michał Marcińczuk 47

Punctuation Restoration from Read Text with Transformer-based Tagger
Tomasz Ziętkiewicz 55

PolEval 2021 Task 2: Evaluation of Translation Quality Assessment Metrics
Krzysztof Wołk, Maciej Szymkowski 61

Simple Recipes for Assessing Translation Quality
Dariusz Kłeczek 67

Approaching English-Polish Machine Translation Quality Assessment with Neural-based Methods
Artur Nowakowski 73

Transformer as Machine Translation Evaluation Metrics
Krzysztof Wróbel 79

PolEval 2021 Task 3: Post-correction of OCR Results
Łukasz Kobyliński, Witold Kieraś, Szymon Rynkun 85

Post-correction of OCR Results Using Pre-trained Language Model
Mateusz Piotrowski 93

OCR Correction with Encoder-Decoder Transformer
Krzysztof Wróbel 97

Simple Yet Effective Method of Post-correcting OCR Errors
Paweł Dyda 103

OCR Post-correction with Heuristics
Michał Marcińczuk 117

PolEval 2021 Task 4: Question Answering Challenge
Maciej Ogrodniczuk, Piotr Przybyła 123

Search-Augmented Question Answering System Using Multilingual Transformer Model
Mateusz Piotrowski 137

Answering Polish Trivia Questions with the Help of Dense Passage Retriever
Aleksander Smywiński-Pohl, Dmytro Zhylko, Krzysztof Wróbel, Magdalena Król 141

Retrieve and Refine System for Polish Question Answering
Piotr Rybak 151

Simple Recipes for Question Answering
Dariusz Kłeczek 159

PolEval 2021: The Fifth Anniversary

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

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

In 2021 we celebrated its fifth edition and got a great gift: a new evaluation platform. Previously, each participant could submit only 3 solutions to a particular task, which where then evaluated by task organizers using custom-made scripts. This approach has obviously limited the interaction between the task organizers, the participants and the general audience. This year, the evaluation platform significantly improved the process for all participants who could track their results on the leaderboard and receive immediate feedback (see Figure 1). We believe that it encouraged them to tweak and resubmit their solutions, thus improving the results of all tasks. It also reduced the work on task organizers side and allowed the audience to observe a live contest, instead of a static results table, published after the competition ended.

We further encouraged interaction by introducing a Discord server intended for discussions about PolEval in real time. This way the participants had a direct channel of communication with PolEval organizers, organizers of particular tasks and could also exchange thoughts and ideas between themselves. An automated message sent to the server after a result beating other submissions has been sent in was also a great nuance engaging the participants.

In 2021 the systems competed in the following tasks:

— Task 1: Punctuation restoration from read text

— Task 2: Evaluation of translation quality assessment metrics (in two variants)

— Task 3: Post-correction of OCR results

— Task 4: Question answering challenge

We can immodestly admit that PolEval 2021 has been a great success, breaking all previous records. We received over 200 submissions from 21 teams, almost twice as many as in all editions to date (see Figures 2 and 3).

Post-correction of OCR results

Overview | Leaderboard | Submissions | Dataset | Submit Solution

Contest only	Submitter	Affiliation	Best Submission Description	Task Version	Entries	test-A WER score	test-B WER score
1	Mateusz Piotrowski	None	XXLv1	2.0.1	5	3.725	3.744
2	Krzysztof Wróbel	eNeLPol UJ AGH	ED 3 pl	2.0.1	4		4.302
3	Paweł Dyda	Adam Mickiewicz University & Applica.ai	uml	2.0.1	4	5.338	5.328
4	Zbigniew Bronk	niezależny	v.1	2.0.1	1		7.208
5	Michał Marcińczuk	Samurai Labs, Wrocław University of Science and Technology	Heuristics	2.0.1	10	8.063	8.217

Figure 1: PolEval 2021 evaluation platform: the leaderboard

This volume consists of proceedings of the online workshop session organized during the AI & NLP Day conference[1] on October 25, 2021, presenting the results of the 2021 edition of the shared task.[2] It starts with the description of our new evaluation platform by its author, Filip Graliński. The main part, as previously, contains four sections dedicated to evaluation tasks. Each section starts with a paper by task organizers, describing the task, its dataset and evaluation procedures, and summarising the submissions and the results. Then selected papers by authors of the solutions are presented.

[1] https://2021.nlpday.pl
[2] http://2021.poleval.pl

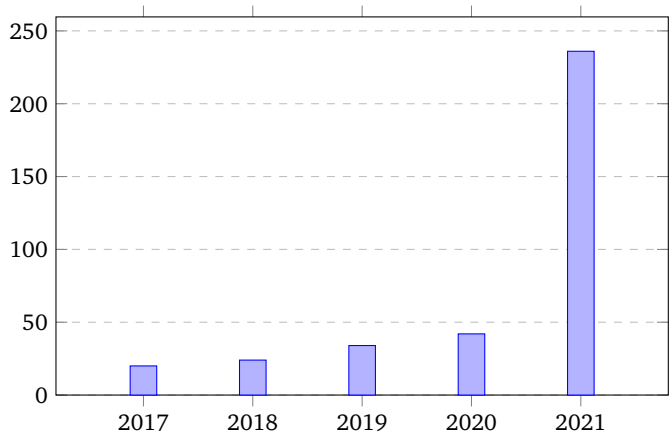

Figure 2: Number of PolEval submissions in 2017–2021

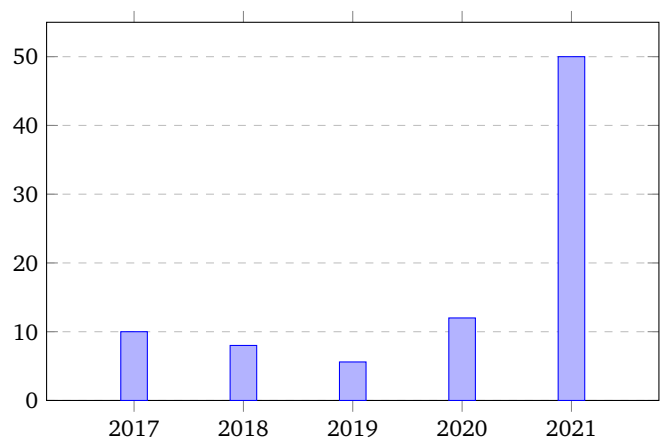

Figure 3: Average number of PolEval submissions per task in 2017–2021

This year two first tasks, *Punctuation restoration from read text* and *Evaluation of translation quality assessment metrics* (in two variants) attracted the most attention, gathering 77 and 80 submissions respectively. *Post-correction of OCR results* had 28 submissions and *Question answering challenge* – 51 submissions in total.

Thank you for being with us again in this difficult COVID time. Please feel free to share your ideas for improving this competition or join in to organize 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 / Sages)

Evaluation platform

Filip Graliński (Adam Mickiewicz University in Poznań)
Agnieszka Olech (Institute of Computer Science, Polish Academy of Sciences)
Karol Saputa (Institute of Computer Science, Polish Academy of Sciences)
Michał Modzelewski (Institute of Computer Science, Polish Academy of Sciences)

Task 1: Punctuation restoration from read text

Agnieszka Mikołajczyk (Voicelab)
Piotr Pęzik (Voicelab / University of Łódź)
Adam Wawrzyński (Voicelab)
Adam Kaczmarek (Voicelab)
Wojciech Janowski (Voicelab)
Michał Adamczyk (Voicelab / University of Łódź)

Task 2: Evaluation of translation quality assessment metrics

Krzysztof Wołk (Polish-Japanese Academy of Information Technology)
Maciej Szymkowski (Białystok University of Technology)

Task 3: Post-correction of OCR results

Szymon Rynkun (University of Warsaw)
Łukasz Kobyliński (Institute of Computer Science, Polish Academy of Sciences / Sages)
Witold Kieraś (Institute of Computer Science, Polish Academy of Sciences)

Task 4: Question answering challenge

Maciej Ogrodniczuk (Institute of Computer Science, Polish Academy of Sciences)
Piotr Przybyła (Institute of Computer Science, Polish Academy of Sciences)

Gonito Platform as the Evaluation Engine in PolEval 2021

Filip Graliński (Faculty of Mathematics and Computer Science, Adam Mickiewicz University / Applica.ai)

Abstract

In this paper, I present the Gonito evaluation platform along with its companion command-line tool, GEval, as used in the PolEval 2021 tasks. The advantages of using Gonito are discussed, along with the features introduced specifically for PolEval.

Keywords

evaluation platform, data formats

1. Introduction

Gonito (Graliński et al. 2016)[1] is an open-source platform for evaluating solutions to machine learning challenges. It is based on GEval[2], a command-line evaluation tool (Graliński et al. 2019). Both tools are written in Haskell, GEval is also exposed as a Haskell library. Over 30 metrics are already implemented in GEval (mostly, but not limited to, metrics used in NLP); together with a flexible system of preprocessing flags it covers a wide range of possible tasks.

In the 2021 edition of the PolEval campaign, Gonito was used for the first time[3] as an evaluation engine for all the five tasks. In this paper, I am going to discuss the rationale behind using Gonito as the evaluation platform in PolEval, present advantages (and disadvantages) of Gonito in this context, showcase new features introduced to Gonito for the needs of PolEval and, finally, demonstrate practical applicability of GEval in the context of PolEval challenges.

[1] `https://gitlab/filipg/gonito`

[2] `https://gitlab/filipg/geval`

[3] Some initial experiments were done during the 2020 edition. See also (Borchmann et al. 2018) for the discussion of using GEval internally by one of the teams submitting a solution to a task in PolEval 2019.

2. Comparison to the previous editions

In the previous editions of PolEval, evaluations were carried out by the authors of the challenges: for each data set, separate evaluation scripts were prepared (in some cases, *ad hoc* scripts prepared specifically for PolEval), the submissions were sent to the organizers and the scripts were run manually against the hidden gold standards. Usually such evaluation scripts would just print the evaluation metric, and even if some of them were equipped with other features, it was not a consistent experience across various challenges (and PolEval editions). Also, the data was in various formats, depending on what was chosen by the authors of a particular task.

By contrast, with Gonito solutions are submitted by participants, *evaluated automatically* (against both the disclosed development sets and hidden test sets) and reflected on a *leaderboard updated continuously*. All tasks share *the same directory scheme* and *similar data format* so that a PolEval platform can provide a unified experience (a PolEval participant learns, more or less, only *once* what to expect from a task).

3. Data format

Gonito (or rather GEval) expects data under evaluation to be formatted in the simplest possible way, i.e. as plain-text files. Input data, expected data and the actual output to be evaluated are assumed to be TSV (tab-separated values) files with the same number of lines, each line corresponding to a single item being considered in evaluation (a sentence or a longer text along with possible metadata). In many cases, e.g. for the output and gold standard for the OCR correction task, TSV files contain just one column, so actually they are simply plain text files: in each line, a corrected or, respectively, expected paragraph is given.

Using plain-text formats goes against the prevalent practice of using JSON or similar formats. This is by design and is motivated by the following advantages of the plain-text format:

— it is easily accessible for humans and future-proof;

— in some type of NLP tasks, it is already popular, e.g. in machine translation, where the evaluation is usually done on text files;

— minimalist and lightweight (repeated, boiler-plate parts are avoided);

— TSV files are compatible with GNU core utilities (in particular, text utilities), with which it is trivial to count the number of items in a test set (`wc`), sort them or draw a random sample (`shuf`), select a column (`cut`), etc. without resorting to any programming language;

— it is easy to create a general algorithm for generating simple baselines (null-model baselines, majority class, etc.);

— representing a challenge in terms of plain texts opens up possibilities for automatically generating advanced baselines using the state-of-art Transformer models without any changes in the model architecture, i.e. treating the task in question as a generation task

(GPT 2/3 models (Radford et al. 2018, Brown et al. 2020)), a text-to-text transformation challenge (cf. T5 model (Raffel et al. 2020)) or a machine translation task where the input is treated as the source text and the (expected) output as the target (Ott et al. 2019).

For instance, the PolEval 2021 punctuation restoration task[4] was expressed by its original authors as a JSON in the following format:

```
{
    "title": "wikitalks00786",
    "words": [
        {
            "word": "piszesz",
            "punctuation": ",",
            "space_after": true
        },
        {
            "word": "że",
            "punctuation": "",
            "space_after": true
        },
        {
            "word": "wyszła",
            "punctuation": "",
            "space_after": true
        },
```

Nevertheless, the problem can simply be stated as transforming the plain text without punctuation marks into the text with them and that's the way it was presented in the final PolEval challenge. To be precise, the input is a two-column TSV file, where the first field is the text ID (so that it could be matched with an audio recording) and the second the text to be fixed. The output is simply the plain text with the punctuation marks, see Table 1.

Gonito format has some disadvantages:

— reading it is usually trivial (simple parsers based on regular expressions are usually enough), but still reading a JSON directly into data structures of a given programming language might be easier;

— as item *must* be expressed as a single line, for multi-line texts some escaping scheme is needed, for instance for the OCR correction task[5] new lines were encoded as `\n`;

— schemas such as JSON Schema cannot be used.

[4] `https://beta.poleval.pl/challenge/punctuation-restoration`
[5] `https://beta.poleval.pl/gonito/challenge/ocr-correction`

Table 1: Sample input and expected output for the punctuation restoration task in the Gonito format

in.tsv	wikitalks00786	piszesz że wyszła cała polska wersja beyblade v force tymczasem można znaleźć tylko 7 odcinków tej serii usunąłem sekcję " fabuła " gdyż zawierała liczne błędy językowe i stylistyczne i była bez źródeł czy przypadkiem nie powinniśmy zrobić serii beyblade oddzielnie no wiecie beyblade beyblade metal fury beyblade v force
	wikitalks00411	lewicowa alternatywa i stowarzyszenie " wolnosc rownosc solidarność " właściwie już nie istnieją może warto wykreślić je z listy obecnie działających organizacji warto za to napisać kilka zdań o " warszawskiej trojce " trzech anarchistach aresztowanych kilka lat temu podczas próby podpalenia radiowozów w warszawie i powiązać ze " stop bzdurom " czy to ci sami
expected.tsv		piszesz, że wyszła cała polska wersja beyblade v- force. tymczasem można znaleźć tylko 7 odcinków tej serii. usunąłem sekcję " fabuła ", gdyż zawierała liczne błędy językowe i stylistyczne i była bez źródeł. czy przypadkiem nie powinniśmy zrobić serii beyblade oddzielnie? no wiecie: beyblade, beyblade: metal fury, beyblade v- force? lewicowa alternatywa i stowarzyszenie " wolnosc- rownosc- solidarność " właściwie już nie istnieją, może warto wykreślić je z listy obecnie działających organizacji? warto za to napisać kilka zdań o " warszawskiej trojce ", trzech anarchistach aresztowanych kilka lat temu podczas próby podpalenia radiowozów w warszawie i powiązać ze " stop bzdurom ". czy to ci sami?

4. Enhancements to Gonito related to PolEval

4.1. Application architecture

Gonito is a monolithic web application based on the Yesod framework, originally written without backend-frontend separation. The main Gonito instance (`https://gonito.net`) works this way (another example is the instance[6] for the Kleister information extraction challenge (Stanisławek et al. 2021)).

[6] `https://kleister.info`

For PolEval 2021, the organizers decided to create a separate front-end to be written using the Django web framework. Hence, Gonito features has been exposed as a REST API and now, as a by-product, are available also for other instances (see, for instance, the `https://gonito.net/api/list-challenges` end-point for listing all challenges[7]).

As can be seen, some changes needed to be done in the GEval evaluation engine for specific needs of the PolEval tasks. Still, new evaluation metrics, preprocessing flags and matching specifications form a modular, re-usable system, where various types of needs can be covered. What is more, when a new evaluation scheme is created it can be combined, out of the box, with powerful GEval features such as line-by-line mode, listing the worst features (Graliński et al. 2019) or bootstrap sampling.

4.2. Evaluation metrics

As no external evaluation scripts can be used with Gonito, the evaluation metrics needed to be realized with the GEval evaluation engine alone.

The simplest case were the blind and non-blind machine-translation quality estimation tasks, as the Pearson correlation coefficient had already been implemented in GEval.

For the OCR correction task, the word-error-rate (WER) evaluation metric had also already been implemented in GEval, however, some pre-processing needed to be done to disregard whitespace characters. Fortunately, GEval comes with powerful *pre-processing flags*, with which output texts can be modified (e.g. with regular expressions) for specific needs. In the case the OCR correction task the `s` substitution flag was used: `WER:s<(\\n|\ )+><\ >` (use WER, but first replace all sequences of spaces and end-of-lines with spaces).

The most complicated metric was the one for the punctuation restoration task, which was F-score weighted across specific punctuation marks. It was not difficult *per se*, but rather in the context in the simple plain-text output assumed (as discussed in Section 3). The solution was to re-purpose the BIO-F1 evaluation metric, but first it needed to be re-implemented in GEval to account for a weighted variant. It might seem surprising as the format on the task is clearly different from the BIO format popular for NER tasks (Ramshaw and Marcus 1999). Nonetheless, the plain text from the task can be converted into the BIO format using pre-preprocessing flags, basically a chain of flags such as `s<\S+!+><B-XCL>` were used to convert words to B tags (as it is only the punctuation marks that matters).

Finally, for the question answering task[8], simple Accuracy was used, however the matching scheme is rather complicated there (fuzzy matching is acceptable, but numbers are treated in a different way). Fortunately, GEval is equipped with a general notion of a *matching specification*. Specifically for the purposes of the task two new matching specifications were implemented (`HardenLenient` and `ExtractNumber`) and chained.

[7] All end-points are described in Swagger at `https://gonito.net/static/swagger-ui/index.html`
[8] `https://beta.poleval.pl/challenge/question-answering`

4.3. Other enhancements

As PolEval tasks are meant to be a long-term endeavor, new submissions can still be done at the platform. Nevertheless, there is a need to clearly separate submissions done before the deadline of competition and later ones. For these reasons, a notion of *challenge phases* were introduced into Gonito. Now, the PolEval challenges are in the `after-poleval-2021` phase and new submissions are distinguished from in-competition solution with a different color.

5. Sample submission

Here I present a sample submission to the punctuation restoration task to showcase some deeper ideas behind the way Gonito handles data and some capabilities of GEval.

Gonito is fundamentally based on git, as both challenges and submissions are uploaded to the platform as git repositories with common history. Even though the PolEval front-end provided an option for manual upload of output files via a web form, it is still recommended providing a URL to a proper git repository. Hence, here I will present the way to submit the solutions using git.

The challenge itself can be downloaded as a git repository:[9]

```
git clone --single-branch \
  https://github.com/poleval/2021-punctuation-restoration \
  punctuation-restoration
```

Now let us create a simple Python script which inserts a comma before some specific words (as a crude baseline solution to the punctuation restoration problem):

```
import re
import sys

for line in sys.stdin:
    print(re.sub(r'(\S+) (że|aby|jak[iaą]|któr[zaey])',
          r'\1, \2', line.rstrip()))
```

Now run the script for the train set and all the test sets:

```
for T in test-[ABCD] train
do
   cut -f 2 < $T/in.tsv | python3 solution.py > $T/out.tsv
done
```

[9]The repository is cloned to a directory without the prefix `2021-` due to inconsistencies in the challenge names.

The output can be easily evaluated locally using the GEval evaluation tool (provided that the expected files are disclosed). The tool can be downloaded as a static binary:

```
wget https://gonito.net/get/bin/geval
chmod u+x geval
```

and then run, for instance, for the train set:

```
./geval -t test-A
```

```
Weighted-F1 14.75
Hyphens-F1 0.00
Comma-F1 36.53
Ellipsis-F1 100.00
Fullstop-F1 0.00
QMark-F1 0.00
Colon-F1 0.00
Excl-F1 0.00
```

As can be seen, not only is the final metric (Weighted-F1) shown, but also scores for each punctuation marks are presented.[10]

In order to submit this solution to PolEval, we need to push the output files to some git repository, also it is recommended to supply the source codes (for reproducibility) and a `gonito.yaml` metadata file, e.g.:

```
description: simple, hand-crafted solution
tags:
  - baseline
  - rule-based
```

```
git remote set-url origin \
   git@github.com:filipggg/2021-punctuation-restoration.git
git config remote.origin.fetch "+refs/heads/*:refs/remotes/origin/*"
git checkout -b simple-solution
git add test-[ABCD]/out.tsv gonito.yaml solution.py
git commit -m 'simple solution'
git push -u origin simple-solution
```

An important point is that a git repository must be accessible by the PolEval platform (with this condition met, any git server is acceptable, whether a popular git platform such as GitHub or GitLab, or a private server). This could be achieved by simply making the repository public, configuring access via the PolEval public key or inviting the poleval-gonito user (on GitHub).

[10] No ellipsis happened to be in the test-A set, hence the 100% result.

The submission itself can be done using the PolEval front-end, another option is to use GEval tool:

```
./geval --submit --token TOKEN \
     --gonito-host https://beta.poleval.pl/gonito
```

where `TOKEN` can be obtained by querying a Gonito end-point[11] while being signed-in.

6. Submission visibility

The results are immediately shown at the public leaderboard (and on a list of all submissions) after a user submitted a solution (see Figure 1).

Submitter	When	Description	Rank	test-A Weighted-F1 score	test-B Weighted-F1 score	test-C Weighted-F1 score	test-D Weighted-F1 score	Tags
Filip Graliński	09/10/2021	simple, hand-crafted solution	30	14.75	9.90	9.67	9.45	baseline rule-based

Figure 1: Evaluation results for a sample submission as shown by the PolEval platform.

Obviously, the results are guaranteed to be the same as the ones obtained locally with GEval (as GEval is used in Gonito for the actual evaluation).

Submitters can *open* their submissions. The repository content is pushed to a publicly accessible content then.

Submissions can be referred to using git SHA1 hashes, such references can be cited in papers, like this: 9.45 {53ed07}, with a reference given in curly brackets. Such a repository may be also accessed by going to `https://beta.poleval.pl/gonito/q` and entering the code there. Note that such submissions might be found even if the PolEval platform cease to exist (e.g. on a git server).

7. Conclusions

Using Gonito as the evaluation back-end for PolEval 2021 brought entirely new possibilities for the competition. It is planned to provide the data sets from the previous editions in the Gonito format.

[11] `https://beta.poleval.pl/api/my-evaluation-trigger-token`

References

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, Warszawa. Institute of Computer Science, Polish Academy of Science.

Brown T. B., Mann B., Ryder N., Subbiah M., Kaplan J., Dhariwal P., Neelakantan A., Shyam P., Sastry G., Askell A., Agarwal S., Herbert-Voss A., Krueger G., Henighan T., Child R., Ramesh A., Ziegler D. M., Wu J., Winter C., Hesse C., Chen M., Sigler E., Litwin M., Gray S., Chess B., Clark J., Berner C., McCandlish S., Radford A., Sutskever I. and Amodei D. (2020). *Language Models are Few-Shot Learners*. In Larochelle H., Ranzato M., Hadsell R., Balcan M. and Lin H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems (NeurIPS 2020)*, vol. 33, pp. 1877–1901. Curran Associates, Inc.

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

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

Ott M., Edunov S., Baevski A., Fan A., Gross S., Ng N., Grangier D. and Auli M. (2019). *fairseq: A Fast, Extensible Toolkit for Sequence Modeling*. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pp. 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Radford A., Wu J., Child R., Luan D., Amodei D. and Sutskever I. (2018). *Language Models are Unsupervised Multitask Learners*. Technical report, Open AI. `https://d4mucfpksywv.cloudfront.net/better-language-models/language_models_are_unsupervised_multitask_learners.pdf`.

Raffel C., Shazeer N., Roberts A., Lee K., Narang S., Matena M., Zhou Y., Li W. and Liu P. J. (2020). *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*. „Journal of Machine Learning Research", 21(140), p. 1–67.

Ramshaw L. A. and Marcus M. P. (1999). *Text Chunking Using Transformation-Based Learning*. In Armstrong S., Church K., Isabelle P., Manzi S., Tzoukermann E. and Yarowsky D. (eds.), *Natural Language Processing Using Very Large Corpora*, pp. 157–176. Springer Netherlands, Dordrecht.

Stanisławek T., Graliński F., Wróblewska A., Lipiński D., Kaliska A., Rosalska P., Topolski B. and Biecek P. (2021). *Kleister: Key Information Extraction Datasets Involving Long Documents with Complex Layouts*. In Lladós J., Lopresti D. and Uchida S. (eds.), *Document Analysis and Recognition – ICDAR 2021*, pp. 564–579, Cham. Springer International Publishing.

PolEval 2021 Task 1: Punctuation Restoration from Read Text

Agnieszka Mikołajczyk, Adam Wawrzyński
(VoiceLab / Gdańsk University of Technology)

Piotr Pęzik, Michał Adamczyk (VoiceLab / University of Łódź)

Adam Kaczmarek, Wojciech Janowski (VoiceLab)

Abstract

This paper presents our contribution to PolEval 2021[1] Task 1: Punctuation Restoration from Read Text. The purpose of this task is to restore original punctuation in automatically recognized speech. The challenge introduces a new text and audio corpus WikiPunct, which contains over 39 hours of speech with time-aligned transcripts and around 38,000 texts of varied length. The data was collected from two distinct sources: WikiNews and WikiTalks. The corpus was automatically cleaned and filtered; the recorded part of the data set was also manually edited. Speech transcripts were force-aligned with edited texts to emulate the ideal ASR output. Participants were allowed to use both text and audio files, but the final evaluation was conducted on speech transcripts. The problem was approached as a task in token classification. Most of the submissions employed transformer-based models, and the winning solution used the HerBERT model.

Keywords

natural language processing, punctuation restoration, speech corpora, ASR, transformers

1. Introduction

Speech transcripts generated by Automatic Speech Recognition (ASR) systems typically do not contain any punctuation or capitalization. In longer stretches of automatically recognized speech, the lack of punctuation affects the clarity of the output text (Yi et al. 2020). The purpose of punctuation restoration (PR) and capitalization restoration (CR) as distinct natural

[1] http://poleval.pl

language processing (NLP) tasks is to improve the legibility of ASR-generated text and possibly other types of texts lacking punctuation. Aside from their intrinsic value, PR and CR may positively affect the performance of other techniques from the field of NLP such as Named Entity Recognition (NER), part-of-speech (POS) and semantic parsing or spoken dialog segmentation (Nguyen et al. 2020, Hlubík et al. 2020).

Nevertheless, it is difficult to systematically evaluate PR on transcripts of conversational language. This is largely because punctuation rules can sometimes be ambiguous even for texts which were originally written with punctuation. The very nature of naturally-occurring spoken language renders the task of identifying transparent phrase and sentence boundaries problematic (Sirts and Peekman 2020, Wang 2020). A set of annotation guidelines tested for rater agreement is necessary to properly evaluate punctuation added to transcripts of spoken texts. Given these requirements and limitations, a PR task based on a redistributable corpus of read text was suggested in the first attempt evaluating PR systems. Figure 1 gives an overview of the punctuation restoration task described in this article.

A corpus of 38,000 texts was collected from two distinct sources: WikiNews and WikiTalks, which, on the one hand, have the advantage of being redistributable. On the other hand, punctuation found in texts from these sources should be approached with some reservation when used for evaluation as they may contain user-induced errors and bias. The texts were read out by 121 different speakers and forced-aligned with recordings. The force-aligned 'transcripts' can thus be treated as an approximation of the ideal ASR transcript of the recordings.

As shown in Figure 1 the goal of the task was to come up with a solution for restoring punctuation in the provided test set. It consists of time-aligned ASR transcripts of read texts from the two sources. Participants were encouraged to use both text-based and speech-derived features to identify punctuation symbols such as the multimodal framework proposed by Sunkara et al. (2020). The train set is accompanied by reference text corpora of WikiNews and WikiTalks, which can be used for training and fine-tuning punctuation models.

2. Data

This section contains the description of the WikiPunct data set prepared for the task of punctuation restoration. The data set contains both conversational and news-oriented components. The description of the two sets of data is followed by a more in-depth account of the data preparation, formatting, organization and descriptive statistics.

2.1. Sources of data

WikiPunct is a crowd-sourced data set of Polish Wikipedia pages read out by Polish speakers. The data set comprises of text and audio files. It was split into two parts, according to the initial data source: (1) the conversational part named WikiTalks and (2) WikiNews, the news-oriented component.

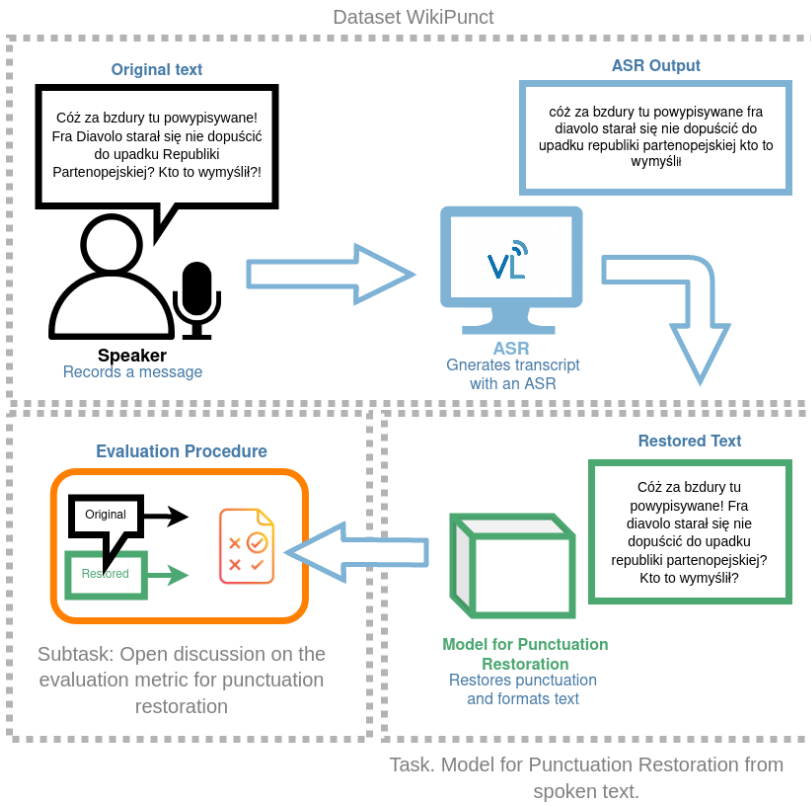

Figure 1: The overview of the punctuation restoration task

WikiTalks is a subset of data scraped from Polish Wikipedia Talk pages. Talk pages, also known as discussion pages, form the administrative side of Wikipedia. They contain ongoing discussions about the contents of articles alongside editorial details pertaining to Wikipedia articles. Talk pages were scrapped from the web using a preexisting list of article titles distributed as a part of the Wikipedia dump archives. Users participating in conversations on Talk pages interact with each other by writing and replying to other people's comments, creating threads, etc. in order to manage and discuss the contents of Wikipedia articles. Given the interactive nature of forum-like conversation, vocabulary and punctuation mistakes should be anticipated. Overall this part of the data set forms the conversational component of WikiPunct.

WikiNews is a free-content news wiki with news written for a global audience; a project of the Wikimedia Foundation. This website works through collaborative journalism where anyone is welcome to make a contribution. The content of news is managed in the same fashion as Wikipedia articles. Decisions about the content of individual news are negotiable and established collectively. The data for the news-oriented component was extracted from Wikinews dump archive. This part of the data set is more passive: it offers text, but it does not reflect the conversational nature of collaborative editing happening behind the scenes

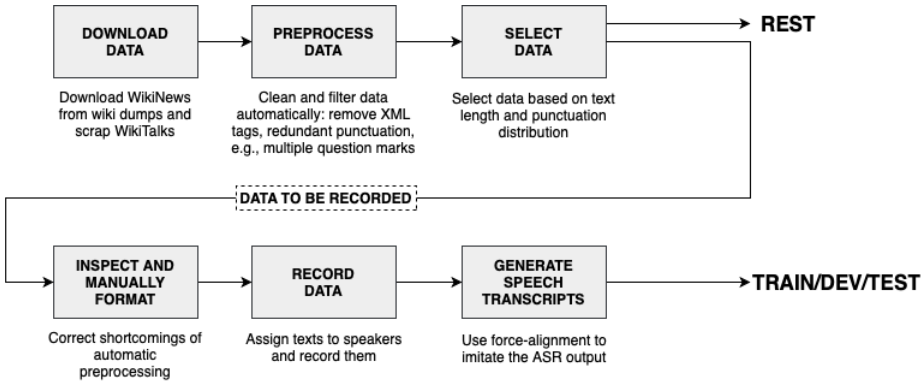

Figure 2: The flow of the data preparation procedure

on editorial pages. With the overall quality of text being high, the collaborative nature of Wikipedia initiatives may bring about some vocabulary and punctuation errors introduced by some of the editors.

2.2. Data preparation

Text data was first either downloaded as a part of wiki dumps or scrapped directly from Wikipedia websites. The unstructured text was automatically cleaned and filtered in order to remove components that would not add much value to the text itself, for example, leftover XML tags. The complex formatting of Wikipedia Talk pages meant that each text had to be passed through a series of regular expressions to eliminate redundant content.

As shown in Figure 2, the next step was to select texts suitable to be rendered as recordings. Each text was assessed in terms of the distribution of different types of punctuation marks, its general quality and length. The working assumption was that the number of running words in a text should fall between 150 and 300 words for it to be included in the set. The number of running words can be proportional to the number of punctuation marks for a given text. By imposing the length restriction, overt outliers are excluded from the set and the text length is controlled. Additionally, a WikiTalks text should contain at least one question mark. Since the final dump from August 1, 2021 used to source data for the test set contained fewer text than anticipated, texts from the previous dump sourced on March 1, 2021 were used with the word limit threshold lowered to 50 and at least one question mark.

The resulting subset of texts was manually edited by native speakers of Polish. Eventually, all text were recorded as they were read out by the native speakers of Polish. Recordings were then force-aligned with the corrected texts to imitate the ideal ASR output.

Over a hundred people were involved in the production of recordings. In total, the length of the audio material amounts to over 39 hours; this includes the test set. The ratio of speakers with masculine and feminine voice features was balanced and should not interfere with the results of individual submissions. There were 58 Polish speakers with masculine voice features

accounting for 18.7 hours of speech and 63 Polish speakers with feminine voice features accounting for additional 20.4 hours of speech.

The forced-aligned texts may contain some minor mistakes resulting from reading errors on the side of the speakers, i.e., skipped portions of text, added words missing from the original. Also some alignment errors might appear owing to the configuration of the alignment program. The configuration was set to target the Polish language, so it may occur that foreign-sounding words such as proper nouns are assigned the duration equal to zero. In other words, start and end timestamps show the same value.

This procedure yielded two data components that were given to the competitors participating in the task. The smaller subset of manually-corrected texts time aligned with recordings formed the *train/dev/test* set split. The large body of automatically cleaned and filtered text was given as the *rest* set.

As for the exact dating of Wikipedia dumps used to source the data for the task, the training data was obtained from two dumps: WikiNews was sourced on March 1, 2021 and WikiTalks on March 10, 2021. The final test data for WikiNews was collated from the dump dated to August 1, 2021 and for WikiTalks this was March 10, 2021.

The data split into training and test sets looks as follows. The training set contains 1,274 recordings and 200 recordings were placed in the test set. As for the balance between the conversational and news components, 80% of all audio files were randomly selected from WikiNews. This component shows much richer punctuation in terms of the raw count of punctuation marks but also in terms of the variance between different types of punctuation marks. The remaining 20% of recordings were selected from WikiTalks.

The final data input is provided as a tab separated value file with two columns. The first column contains text identifiers. It can be utilized when handling forced-aligned transcriptions and WAV files. The second column stores the input text, all lower cased and stripped of all punctuation marks. To maintain consistency, the submitted output should have the same number of lines as the input file with each line containing text with predicted punctuation marks.

2.3. Descriptive statistics

Table 1 shows statistics for the entire WikiPunct and for the four subsets of data separated for the purpose of training, validating and testing models. Each subset is summarized in terms of the count of punctuation marks falling into one of ten classes with the first eight being predicted. The number of instances for each class is balanced across all four subsets of data. The *rest* subset is the largest because it contains all texts that do not have corresponding recordings.

Table 2 gives an overview of WikiTalks. Parenthesized values correspond to the recorded portion of the subset; the regular ones refer to the subset as a whole. The medians for the portions with and without recordings are similar with small exceptions, for example, there are no instances of ellipsis in the recorded set. The mean and max are more divergent. This might result from the fact that the texts in the recorded set have between 50 and 300 words.

Table 1: Raw statistics for the subsets of WikiPunct

	symbol	train	rest	dev	test	sum	predicted
full stop	.	10 372	370 538	2 573	2 856	386 339	yes
comma	,	10 069	331 404	2 498	2 542	346 513	yes
question mark	?	768	24 262	149	449	256 513	yes
exclamation mark	!	114	8 328	23	44	8 509	yes
hyphen	-	2 422	72 598	621	567	76 208	yes
colon	:	897	42 597	323	433	44 250	yes
ellipsis	...	0	8 053	0	0	8 053	yes
semicolon	;	90	4 049	11	63	4 213	yes
quote	"	3 363	106 511	720	957	111 551	no
words		163 471	5 199 670	40 842	45 461	5 449 444	no

The mean and max observed for the whole subset is affected by the presence of extra long texts. For instance, the maximal number of full stops in a single text amounts to 1,130 in the complete set; for the recorded subset, there are no more than 25 full stops in a single text.

Table 2: Statistics for WikiTalks

	mean		median		max		sum	
full stop	25.96	(12.72)	11	(12)	1130	(25)	225 307	(3 587)
comma	23.81	(12.98)	9	(12)	1283	(64)	203 703	(3 660)
question mark	4.58	(4.61)	2	(4)	140	(12)	24 418	(1 292)
exclamation mark	3.32	(1.62)	2	(1)	112	(7)	7 565	(141)
hyphen	6.69	(3.37)	3	(3)	305	(12)	42 650	(778)
colon	5.28	(2.56)	2	(2)	408	(15)	32 003	(558)
ellipsis	2.81	(0.00)	2	(0)	60	(0)	7 692	(0)
semicolon	2.30	(1.69)	1	(1)	50	(17)	3 055	(59)
quote	12.38	(7.38)	6	(6)	346	(74)	75 747	(1 749)
words	365.02	(225.37)	145	(225)	17 633	(312)	3 179 728	(63 555)

The overview of WikiNews is given in Table 3. Similarly to Table 2, parenthesized values describe the recorded subset of data and regular ones refer to the complete set. WikiNews is particularly important because it accounts for 80% of all recorded texts. The variation in median between the parts with and without recordings indicates that the set of recorded texts is fairly representative of the whole WikiNews. Full stops and commas are over-represented by a slight margin in the recorded set. Ellipses never made it to recorded set. As for mean and max, differences are not as apparent as in the case of WikiTalks. This might owe to the fact that news is generally a shorter and edited text type. Maximal values are higher for the complete set than they are for the recorded part; this can be also explained in terms of long text being excluded from the recorded sample. The mean for all punctuation mark classes is fairly similar for both the whole subset and the recorded portion of texts.

Table 3: Statistics for WikiNews

	mean		median		max		sum	
full stop	10.96	(12.40)	7	(12)	246	(36)	161 032	(12 214)
comma	10.50	(11.73)	7	(11)	347	(71)	142 810	(11 449)
question mark	2.32	(1.51)	1	(1)	50	(6)	1 210	(74)
exclamation mark	2.16	(1.43)	1	(1)	55	(3)	944	(40)
hyphen	3.19	(3.57)	2	(3)	76	(26)	33 558	(2 832)
colon	2.33	(2.27)	1	(1)	100	(92)	12 247	(1 095)
ellipsis	1.54	(0.00)	1	(0)	20	(0)	361	(0)
semicolon	2.23	(2.50)	1	(1)	53	(8)	1 158	(105)
quote	5.34	(5.95)	4	(4)	92	(50)	35 804	(3 291)
words	153.96	(160.82)	122	(158)	4229	(295)	2 269 716	(158 410)

3. Evaluation procedure

The following section gives an account of the evaluation procedure established for the purpose of the punctuation restoration task. It details the procedure in terms of steps necessary to arrive at the final assessment of individual submissions.

The evaluation of the final results assumes that the following three steps are taken. First of all, scores for label classes corresponding to individual punctuation marks are calculated separately for each text. Secondly, scores are aggregated together across all texts. And finally, the same kind of aggregation is carried out; this time, for unique classes of punctuation marks.

The final score for an individual solution submitted for the task relied on the following definitions and metrics.

Document is defined as the sequence of tokens $t_1, t_2, \ldots, t_n$. A single token can be followed by no more than one punctuation mark. The sequence of punctuation marks corresponding to tokens is defined as $\mathbf{y} = y_1, y_2, \ldots, y_n$ where $y_i \in P$ and $P = \{., ? ! - : \ldots ; ”\}$. The reference punctuation mark sequence is denoted with $\mathbf{y^{true}}$ while $\mathbf{y^{pred}}$ stands for the predicted punctuation mark sequence.

Individual punctuation marks are tallied up into *true positives* (TP), *false positives* (FP) and *false negatives* (FN) for separate punctuation mark classes on per-document basis. The values are calculated as follows. Equation 1 shows that the number of TPs equals the number of punctuation marks $p \in P$ where the class of the punctuation mark p is predicted *correctly*.

$$TP_d^p = \sum_{i=0\ldots n} \mathbb{1}[y_i^{true} = y_i^{pred} = p] \tag{1}$$

FPs are calculated as given in Equation 2, which shows that the number of FPs equals the number of punctuation marks $p \in P$ where the class of the punctuation mark p is predicted *incorrectly*.

$$FP_d^p = \sum_{i=0...n} \mathbb{1}[y_i^{true} \neq y_i^{pred} = p] \tag{2}$$

Finally Equation 3 shows that the number of FNs equals the number of punctuation marks $p \in P$ where the class of the punctuation mark p is *incorrectly* omitted.

$$FN_d^p = \sum_{i=0...n} \mathbb{1}[y_i^{pred} \neq y_i^{true} = p] \tag{3}$$

In order to arrive at local scores for individual documents, the values for TP, FP and FN are calculated for each punctuation mark class p present in a given document; then they are fed into Equation 4 to arrive at a harmonic mean of precision and recall (F1-score).

$$P_d^p = \frac{TP_d^p}{TP_d^p + FP_d^p} \quad ; \quad R_d^p = \frac{TP_d^p}{TP_d^p + FN_d^p} \quad ; \quad {F_1}_d^p = 2 * \frac{P_d^p * R_d^p}{P_d^p + R_d^p} \tag{4}$$

The aggregate global score for separate punctuation mark classes p is calculated as the micro-average of scores for individual documents as in Equation 5:

$$P^p = \underset{d \in D}{micro\text{-}avg}\, P_d^p = \frac{\sum_{d \in D} TP_d^p}{\sum_{d \in D} TP_d^p + FP_d^p} \; ; R^p = \underset{d \in D}{micro\text{-}avg}\, R_d^p = \frac{\sum_{d \in D} TP_d^p}{\sum_{d \in D} TP_d^p + FN_d^p} \tag{5}$$

The final global score for a submission is calculated as the weighted (macro) average of F_1 scores for each of the punctuation mark classes:

$$score = \frac{1}{\sum_{p \in P} N_p} \sum_{p \in P} N_p * {F_1}^p \tag{6}$$

Individual submissions are compared with respect to the weighted average of F_1 scores obtained for each punctuation mark class.

4. Submissions and results

A number of solutions were submitted for the punctuation restoration task. Authors of four submissions were willing to present their results accompanied by a brief description of how they tackled the challenge. All authors presented state-of-the-art solutions to the problem.

The four submissions utilized a number of shared approaches. All four submissions employed BERT-derived transformer architectures for sequence encoding. In all cases, the task was considered an exercise in token classification. Authors of two submissions reported using ensemble techniques. In one of the solutions, authors experimented with adding a bi-LSTM layer as the last hidden layer and adding vectors obtained from a wave2vec model to enhance their prediction. Individual submissions are described in Sections 4.1–4.4.

Table 4 reports on the scores of individual submissions. The winning submission *eNeLPol AGH UJ — S1* by Krzysztof Wróbel and Dmytro Zhylko scored 81.29. The second-best submission *CLARIN — HLV* by Norbert Ropiak scored 81.25. The third-best submission *Samurai Labs PWr* by Michał Marcińczuk scored 81.23. The fourth-best submission *Samsung & UAM* by Tomasz Ziętkiewicz scored 78.37.

Table 4: Results of PolEval 2021 Task 1

Submission	**.**	**,**	**?**	**!**	**-**	**:**	**...**	**Total**
eNeLPol AGH UJ — S1	88.68	76.08	80.61	36.36	66.91	82.98	0.00	81.29
CLARIN — HLV	88.50	76.63	80.90	0.00	66.79	81.93	0.00	81.25
Samurai Labs PWr	88.47	77.08	79.84	20.00	66.30	79.73	0.00	81.23
Samsung & UAM	85.65	76.36	69.23	0.00	57.50	77.19	0.00	78.37

4.1. Punctuation restoration with transformers and an overlapping sliding window

The submission by Wróbel and Zhylko (2021) treated punctuation restoration as a token classification task. Pre-processed data is encoded with the large variant of herBERT with an overlapping sliding window and the sequence length of 256. Hyperparameters were optimized on the manually annotated subset. the top result assumed the following hyperparameters: batch size of twelve, ten epochs, the evaluation after every fifty steps and the learning rate at 5e-6. The total of four submissions differ with respect of the training data, validation and applied ensemble techniques. The final submission used retrained models trained on the *rest* data set with data from the *test-A* data set with validation on the *train* data set and relied the most optimal out of seventy models. The winning submission scored 81.29.

4.2. Punctuation restoration with transformer language model with a bi-LSTM layer

The submission by Ropiak et al. (2021) reported investigating language models leveraging the transformer architecture for classification and sentence transformation (seq2seq). A number of possibilities were inquired into to arrive at an optimal classifier, namely, adding an LSTM layer, adding vectors obtained from the wave2vec model and their combinations. The final architecture of the classifier is that of the deep language model (the large variant of herBERT) with a bi-LSTM layer at the last hidden layer of the language model and the fully-connected layer handing one of eight punctuation classes. The final classifier assigned predicted punctuation marks for individual tokens. The submission scored 81.25.

4.3. Punctuation restoration with an ensemble of transformer language models

The submission by Marcińczuk (2021) tackled the task with a unary classifer based on a neural network. The neural network was trained to predict which punctuation mark should be inserted after a token. Two pre-trained, attention-based transformer models for Polish, RoBERTa and herBERT, were used to encode tokens. The attention mechanism was responsible for the retention of the semantics of tokens and their context. A single neural network with the large variant of the Polish RoBERTa was slightly lower on performance than the final submission and scored 80.53. The final solution based on the ensemble of RoBERTa- and herBERT-based models trained on different subsets of training data scored 81.23.

4.4. Punctuation restoration with the transformer-based tagger

The submission by Ziętkiewicz (2021) treated the punctuation restoration task as an exercise in token classification. While processing data, each token followed by a punctuation mark was assigned a class label corresponding to the punctuation mark. The actual punctuation was removed from the text. The classification model was trained on sentence encoded with herBERT. The evaluation assumed that the model predicts which punctuation mark should be inserted after each token or whether it should be left without any punctuation. The final submission scored 78.37.

5. Conclusions and future work

This year's PolEval challenge of PR from read text has confirmed the popularity of pre-trained transformers as an effective approach to sequential classification problems in NLP. At the same time, several methodological questions in the area of increasing the legibility of ASR transcripts remain open. For example, generating punctuation from conversational language transcripts seems to be a separate task requiring dedicated data sets based on annotation guidelines. Also, the choice of evaluation metrics for the PR task is not obvious as PR annotations may exhibit far from perfect inter-rater agreement while still being useful in downstream applications. Data sets designed for evaluating PR and truecasing in transcripts of conversational speech are currently being developed for the next year's edition of PolEval.

Acknowledgements

Research reported in this publication was supported by the National Centre for Research and Development (Narodowe Centrum Badań i Rozwoju) under award project number PO IR.01.01.01-00-1237/19.

References

Hlubík P., Španěl M., Boháč M. and Weingartová L. (2020). *Inserting Punctuation to ASR Output in a Real-Time Production Environment*. In Sojka P., Kopeček I., Pala K. and Horák A. (eds.), *Text, Speech, and Dialogue*, pp. 418–425, Cham. Springer International Publishing.

Marcińczuk M. (2021). *Punctuation Restoration with Ensemble of Neural Network Classifier and Pre-trained Transformers*. In Ogrodniczuk and Kobyliński (2021), pp. 47–53.

Nguyen T. B., Nguyen Q. M., Thu H. N. T., Do Q. T. and Mai L. C. (2020). *Improving Vietnamese Named Entity Recognition from Speech Using Word Capitalization and Punctuation Recovery Models*. In Meng H., Xu B. and Zheng T. F. (eds.), *Proceedings of 21st Annual Conference of the International Speech Communication Association (Interspeech 2020)*, pp. 4263–4267. ISCA.

Ogrodniczuk M. and Kobyliński Ł., editors (2021). *Proceedings of the PolEval 2021 Workshop*, Warsaw, Poland. Institute of Computer Science, Polish Academy of Sciences.

Ropiak N., Pogoda M., Radom J., Gawron K., Swędrowski M. and Bojanowski B. (2021). *Comparison of Translation and Classification Approaches for Punctuation Recovery*. In Ogrodniczuk and Kobyliński (2021), pp. 39–46.

Sirts K. and Peekman K. (2020). *Evaluating Sentence Segmentation and Word Tokenization Systems on Estonian Web Texts*. In Andrius Utka Jurgita Vaičenonienė J. K. and Kalinauskaitė D. (eds.), *Human Language Technologies — The Baltic perspective. Proceedings of the 6th International Conference Baltic HLT 2014*, pp. 174–181. IOS Press.

Sunkara M., Ronanki S., Bekal D., Bodapati S. B. and Kirchhoff K. (2020). *Multimodal Semi-supervised Learning Framework for Punctuation Prediction in Conversational Speech*. In *Proceedings of 21st Annual Conference of the International Speech Communication Association (Interspeech 2020)*, pp. 4911–4915.

Wang X. (2020). *Analysis of Sentence Boundary of the Host's Spoken Language Based on Semantic Orientation Pointwise Mutual Information Algorithm*. In *Proceedings of the 12th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA 2020)*, pp. 501–506, Los Alamitos, CA, USA. IEEE Computer Society.

Wróbel K. and Zhylko D. (2021). *Punctuation Restoration with Transformers*. In Ogrodniczuk and Kobyliński (2021), pp. 33–37.

Yi J., Tao J., Bai Y., Tian Z. and Fan C. (2020). *Adversarial Transfer Learning for Punctuation Restoration*. arXiv:2004.00248.

Ziętkiewicz T. (2021). *Punctuation Restoration from Read Text with Transformer-based Tagger*. In Ogrodniczuk and Kobyliński (2021), pp. 55–60.

Punctuation Restoration with Transformers

Krzysztof Wróbel (Enelpol / Jagiellonian University / AGH University of Science and Technology)

Dmytro Zhylko (AGH University of Science and Technology / Enelpol)

Abstract

This paper presents a contribution[1] to PolEval 2021 Task 1: Punctuation restoration from read text. The system was scored first and achieved 81.29% weighted F1.

Keywords

natural language processing, punctuation restoration

1. Introduction

Punctuation restoration is an important problem for automatic speech recognition (ASR) outputs. ASR systems usually produce output as a sequence of words without interpunction. In this article we aim to present capabilities of pre-trained BERT (Devlin et al. 2019) model to restore missing punctuation using only textual data. We model this problem as sequence labeling task. In Section 2 we describe datasets, preprocessing and used data splits. Section 4 describes evaluation metrics. Sections 5–6 describe model architecture, parameters and applied techniques. Sections 7–8 contain our results and conclusions.

2. Data

Experiments were conducted on different combinations and splits of two distinct datasets: **WikiNews** and **WikiTalks**.

WikiNews is a Wikimedia Foundation project that is based on collaborative journalism, dataset contains high quality, grammatically correct data, punctuation and other types of errors are rare.

[1] `https://github.com/enelpol/poleval2021-task1`

WikiTalks is a dataset of conversational based on the Wikipedia discussions pages associated with Wikipedia articles found in Wikipedia dump, punctuation and grammatical errors are more common.

Table 1 shows descriptions of all shared dataset splits. *rest* split contains original text from WikiNews and WikiTalks. Other splits were read, recognized by ASR systems and manually corrected adding puntuation marks.

Table 2 presents a number of texts in each split with the ratio of WikiNews and WikiTalks texts. Punctuation restoration is an important problem for automatic speech recognition (ASR) outputs. ASR systems usually produce output as a sequence of words without interpunction.

Table 1: Dataset splits with descriptions

Split name	Description
rest	All available text data except data used for model evaluation
train	Original training set, in out experiments mainly used as development set
test-A	Original development set, after initial experiments was combined with *train*
test-B/C/D	Datasets are consecutive iterations of final test dataset (*test-D* was used as final test set)

Table 2: Data splits with number of texts from each dataset

Split name	Total size	WikiNews		WikiTalks	
rest	22 186	13 757	(62.00%)	8429	(38.00%)
train	800	632	(79.00%)	168	(21.00%)
test-A	200	168	(84.00%)	32	(16.00%)
test-B	185	185	(100%)	0	(0.00%)
test-C	274	185	(67.52%)	89	(32.48%)
test-D	244	155	(63.52%)	89	(36.47%)

3. Preprocessing

All data splits except *rest* were processed by organizers and can be found in competition repository[2].

rest split has been processed to resemble train data:

— Quotation marks and plus characters were separated.

— Multiple whitespaces deleted.

— HTML entities were replaced to its characters.

Table 3 shows label statistics in *rest* and *train* splits for each dataset.

[2] `https://github.com/poleval/2021-punctuation-restoration.git`

Table 3: Statistics of punctuation marks in *rest* and *train* splits divided into both datasets. About 15% of all tokens are tokens with punctuation marks

	rest		**train**	
	WikiNews	**WikiTalks**	**WikiNews**	**WikiTalks**
-	6.36%	5.73%	0.00%	0.52%
,	42.75%	41.90%	46.53%	37.86%
;	0.33%	0.58%	0.29%	0.72%
:	3.48%	6.16%	3.55%	5.58%
!	0.25%	1.09%	0.19%	1.43%
?	0.37%	4.49%	0.26%	13.87%
.	46.35%	38.34%	49.18%	39.96%
...	0.12%	1.69%	0.00%	0.06%

4. Evaluation

We used weighted F1 as our main metric as in task specifications.

5. Methods

We used pre-trained HerBERT model (Mroczkowski et al. 2021) as it is currently the best model for Polish language according to KLEJ benchmark (Rybak et al. 2020). Additionally we used single linear layer to make predictions for each word in the text. More specifically we predict punctuation mark that comes after given word, in case of word being split into few tokens we predict punctuation only for *first* token.

We also experimented with Conditional Random Fields (CRF) layer, but didn't get any tangible improvements.

6. Experiments

For all experiments data was previously splitted into chunks that can fit into *experiment* specific parameter of maximum token sequence length (max_seq_len) with *experiment* specific overlap between splits. Training data was also splitted, but without overlap to not worsen label disbalance in data.

6.1. PolEval Submissions

Common model parameters for PolEval experiments:

— model: HerBERT large

— max sequence length: 256

Few models were trained on *rest* corpus with *train* data as validation. Then hyperparameter optimization was performed using *test-A* combined with *train* corpus as validation over parameters:

— model: one of few trained on rest corpus

— batch size: 4, 8, 12

— epochs: 1, 2, 4, 10

— learning rate: 5e-6, 1e-5, 2e-5, 5e-5

— evaluation every 50 steps

70 models were trained and the best run achieved the best result on the final leaderboard. Final model parameters are following: batch size 12, epochs 10, learning rate 5e-6.

Submitted solutions:

— AE1 — models trained on *rest*, *train* and *test-A* corpora, without validation, majority voting

— E1 — models trained on *rest*, with validation on *train*, majority voting

— SE1 — further training of models trained on *rest* corpus with *test-A* corpus, with validation on *train*, majority voting

— S1 — further training of models trained on *rest* corpus with *test-A* corpus, with validation on *train*, the best model from 70 models

7. Results

Table 4 presents final results in PolEval competition. Our submission *S1* won with small margin over other contestants.

Table 4: PolEval best results and scores by different submissions

	test-D weighted F1
AE1	80.09
E1	80.86
SE1	81.27
S1	**81.29**
Norbert Ropiak	81.25
Michał Marcińczuk	81.23

Table 5 presents F1 scores for each punctuation marks. The most influence on weighted F1 has F1 scores of comma and dot.

Table 5: Results of S1 submission per label

Punctuation mark	test-D F1
-	66.91
,	76.08
...	0.00
.	88.68
?	80.61
:	82.98
!	36.36

8. Conclusions

The top three contestants obtained very similar results. Our success depends on proper preprocessing, good correlation between scores on *train* and *test-D* and 70 experiments in hyperparameter optimization process.

Acknowledgments

This work was supported in part 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" and in part by the PLGrid Infrastructure.

References

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

Mroczkowski R., Rybak P., Wróblewska A. and Gawlik I. (2021). *HerBERT: Efficiently pretrained transformer-based language model for Polish*. In *Proceedings of the 8th Workshop on Balto-Slavic Natural Language Processing*, pp. 1–10, Kiyv, Ukraine. Association for Computational Linguistics.

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

Comparison of Translation and Classification Approaches for Punctuation Recovery

Norbert Ropiak, Michał Pogoda, Jarema Radom, Karol Gawron, Michał Swędrowski, Bartłomiej Bojanowski (CLARIN)

Abstract

This paper presents our approach to solving Task 1 from the PolEval[1] 2021 competition – Punctuation restoration from read text. We present the selection of approaches to solve the problem, the architectures of the models tested, and the evaluation metrics used.

Keywords

punctuation recovery, natural language processing, computational linguistics, language model, sequence-to-sequence

1. Introduction

In the field of natural language processing (NLP), punctuation recovery has received increasing attention due to the development of automatic speech recognition tasks. Outputs of automatic speech recognition systems or transcripts of meetings or parliament sessions and sittings typically do not contain any punctuation or capitalization. The lack of this information in the raw text affects human readability and comprehension of the text and poses some difficulties in further processing such text with standard methods. Therefore, restoring punctuation has a positive impact not only on text clarity, but also on other text processing tasks such as tokenization, morpho-syntactic tagging, named entity recognition (Alam et al. 2015), text parsing or machine simultaneous interpretation (Chen and Shi 2020). This is because state-of-the-art models are mostly learned on large corpora that contain punctuation (i.e. Wikipedia, newspaper articles, websites). In addition, texts scraped from the Internet may contain punctuation errors.

Restoring punctuation is not as simple as it might seem, because punctuation rules can be relatively ambiguous in human annotations. Furthermore, there is not always a perfect correspondence between the locations of pauses occurring during reading and sentence splits.

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

2. Related work

Research on restoring punctuation in text has a long history, dating back to the 20th century (Stolcke et al. 1998). Much of the recent work can be classified into one of three approaches: models based on prosodic or audio features (Sinclair et al. 2014), models based on lexical features (Ueffing et al. 2013, Alam et al. 2020, Courtland et al. 2020, Nagy et al. 2021), and hybrid models of the previous two approaches (Szaszák and Ákos Tündik 2019).

Historically, a very wide range of approaches have been tried. Both classical methods such as: the hidden event language model based on N-grams and decision trees on prosodic features (Stolcke et al. 1998) or purely text-based n-gram language model (Gravano et al. 2009), linear/factorial conditional random fields (CRF) (Lu and Ng 2010), various kinds of textual features combined with CRF (Ueffing et al. 2013). And also models based on neural networks (results show that neural networks, achieve better results than CRF on text data (Che et al. 2016)): long short-term memory (LSTM) recurrent neural network (Tilk and Alumäe 2015), bidirectional RNN (B-RNN) with attention mechanism (Tilk and Alumäe 2016), B-RNN with gated recurrent units (Salloum et al. 2017), convolutional neural network (CNN)/biLSTM with GloVe embeddings (Łukasz Augustyniak et al. 2020, Żelasko et al. 2018), character-level LSTM (Gale and Parthasarathy 2017). After the appearance of the BERT-like model, most state-of-the-art solutions were based on this architecture with various additional layers: LSTM, fully connected (FC) (Alam et al. 2020, Courtland et al. 2020, Nagy et al. 2021).

Most research papers approach the punctuation restoration problem as a token classification/-tagging task. However, other approaches such as machine translation (sequence-to-sequence) also appear (Wang et al. 2018, Gangi et al. 2019, Klejch et al. 2017).

3. Methods

In our solution, we mainly focused on exploring the two most popular approaches from the literature review: token punctuation classification and sentence transformation (sequence-to-sequence). Within these approaches, we considered multiple architectures and hyperparameters.

3.1. Token punctuation classification

Most of the state-of-the-art solutions approach the punctuation restoration problem as a classification problem in which one class of punctuation is predicted for each token. An additional assumption that needs to be made is to ensure that for a single word that consists of two or more tokens, two different punctuation classes cannot be classified. Because of the results and the popularity of the classification approach, we chose it as the baseline for our research. For the classifier module, transformer architecture based models were chosen as they are the most commonly used and obtain the best results in KLEJ Benchmark[2] (set

[2] https://klejbenchmark.com/leaderboard/

of nine evaluation tasks for the Polish language understanding). Specifically, herBERT-large (Mroczkowski et al. 2021) was selected as the average best model according to this ranking.

The basic architecture of the classification module is based on the encoding part of the transformer architecture, i.e., the processing of a high-dimensional input representation by encoding layers into a lower dimensional final latent representation. The next step is a linear layer on top of the hidden states output, which returns logits for all classes. In an extension of this architecture, a bi-LSTM layer is added between the output of the hidden states and the linear layer, which is shown in Figure 1.

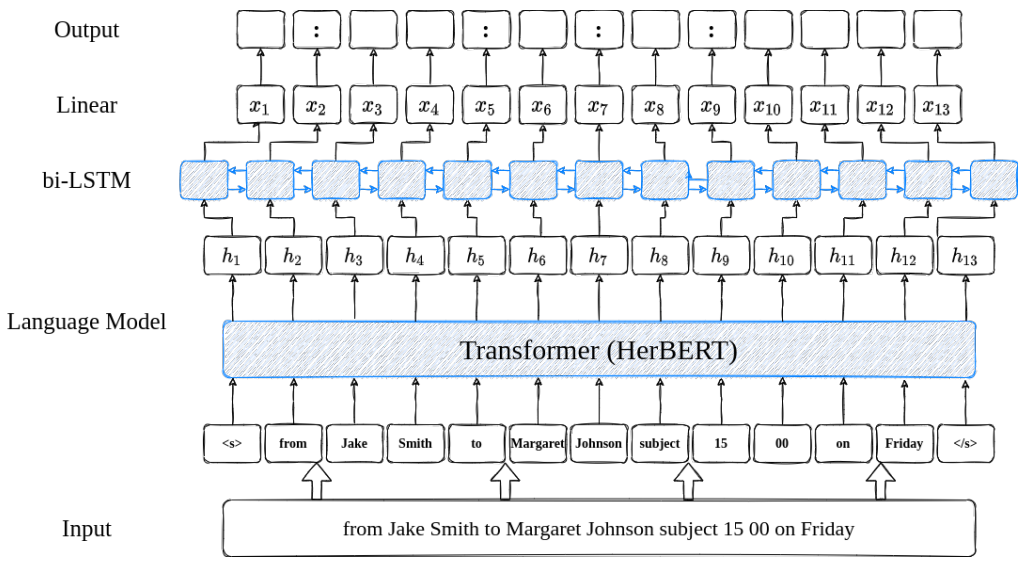

Figure 1: The architecture of classification module with additional bi-LSTM layer

3.2. Sentence transformation

A alternative approach to restoring punctuation is the machine translation/seq2seq problem, where both input and output are text. In this approach we also used a transformer architecture, both encoder and decoder. The idea behind this idea is to apply restrictions on the predicted tokens in the decode step, that model can only insert punctuation tokens into sentence.

4. Experiments

During our study, we conducted many experiments considering different configurations of architectures and training hyperparameters. The models are implemented in PyTorch Lightning (Falcon 2019) with PyTorch (Paszke et al. 2019) back-end. Implementations and pretrained model weights from the transformers (Wolf et al. 2020) library and the datasets

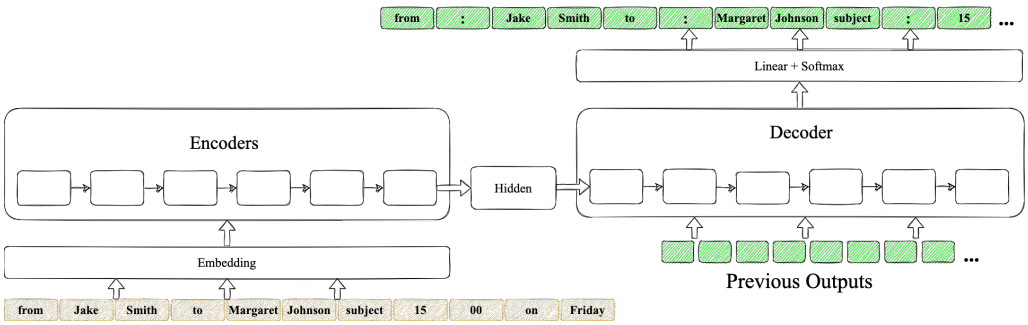

Figure 2: The architecture of a machine translation module for punctuation recovery

(Lhoest et al. 2021) library were used to efficiently load textual datasets. Data versioning and processing pipeline was implemented using DVC (Kuprieiev et al. 2021). During the training, the weights were updated using the Adam optimizer and we used categorical cross-entropy as the loss function.

4.1. Data preprocessing

We used datasets prepared by the organizers to train the models: WikiTalks and WikiNews. In the first preprocessing step, we converted the data from json to jsonlines format (jsonl, where each line represents an individual row of data/json object), recommended by the DATASETS library for efficiency. Then changed all letters to lowercase and prepared a function that maps all the punctuation characters that appear in the dataset to the seven defined in the task description (for example: ".-." → "-", ",-" → ","). In the final step, too long documents (longer than the standard transformer model input of 512 tokens) were split into smaller texts. Chunks were created greedily from left to right, and the split was determined at the location of the dots (naive sentence division).

4.2. Results

The main metric used to evaluate models in first task of PolEval 2021 was weighted F_1 (average of all classes weighted by support — the number of true instances for each label).

$$F_1^w = \frac{1}{n} \sum_{p \in \text{punct}} \text{support}(p) * F_1(p)$$

The auxiliary metrics we checked for comparison of trained models were F1 macro, precision, recall — averages across all classes and values for individual punctuation marks. All metrics and loss function values were logged using the WANDB library (Biewald 2020) for easy comparison between models.

The results for the selected architecture configurations are shown in Table 1. Overall, the classifier models performed better than the seq2seq models. Of the classifiers tested, the

Table 1: Results of the models on the valid and test-A datasets (train:valid — 85:15 ratio)

	valid			test-A		
	F_1	**Weighted F_1**	**Accuracy**	**F_1**	**Weighted F_1**	**Accuracy**
herbert-large	68.15%	79.03%	94.89%	61.86%	79.93%	95.16%
herbert-large-lstm-ft	68.47%	79.09%	94.94%	61.69%	79.69%	95.17%
herbert-large-lstm	67.62%	78.95%	94.84%	61.06%	79.79%	95.09%
herbert-large-lstm-valid	—	—	—	63.07%	83.81%	96.03%
bart	56.20%	68.65%	92.81%	56.24%	74.90%	94.53%
mT5	53.32%	72.23%	93.61%	52.11%	72.16%	94.32%

model with herbert-large and bi-LSTM layer achieved the best results. The final version of this model (herbert-large-lstm-valid) was further trained on a combined training and validation set to provide the largest possible final training dataset.

— herbert-large — classification model with linear layer on top of hidden states output of herbert-large

— herbert-large-lstm-ft — classification model with frozen encoder's layers (weights from herbert-large training) and only trainable bi-LSTM and linear layers.

— herbert-large-lstm — classification model with additional bi-LSTM layer between hidden states and linear layer

— BART — constrained seq2seq model based on BART architecture

— mT5 — constrained seq2seq model based on mT5 architecture

According to the official Poleval 2021 results for the first punctuation recovery task, our submission (herbert-large-lstm-valid) received the second best score. The top in the leaderboard was very close — 0.06% as measured by the F1-weighted metric and our solution was only 0.04% behind the first place. We believe that by further increasing and improving the quality of the training dataset, adding audio features and experimenting with the model architecture, the result could be further improved.

Table 2: Results on the test datasets prepared by the PolEval 2021 competition organizers

	test-A	**test-B**	**test-C**	**test-D**
herbert-large	79.69%	82.34%	74.78%	78.74%
herbert-large-lstm	79.79%	83.51%	75.52%	80.04%
herbert-large-lstm-valid	80.56%	84.16%	76.03%	81.25%
bart	74.90%	76.82%	69.77%	71.78%
mT5	72.16%	73.85%	67.28%	69.84%

There is a huge difference between the classification and seq2seq models, up to 400 times, comparing the inference time on the validation set. By introducing additional constraints to the models from machine translation architecture resulted in increasing processing times.

Table 3: Comparison of inference times with GPU utilization on a validation dataset (409 documents — 4 534 087 characters)

	Inference time
herbert-large	80 s
herbert-large-lstm	80 s
bart	10 292 s
mT5	34 448 s

5. Conclusion

We presented our solution and a comparison of two approaches: classification and machine translation to the punctuation restoration problem, Task 1 of PolEval 2021. We compared the results obtained on the test and validation datasets and the inference time. Despite ignoring audio features the results of the models are satisfactory and ready for further development and extension.

Acknowledgments

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

References

Alam F., Magnini B. and Zanoli R. (2015). *Comparing Named Entity Recognition on Transcriptions and Written Texts*. In Basili R., Bosco C., Delmonte R., Moschitti A. and Simi M. (eds.), *Harmonization and Development of Resources and Tools for Italian Natural Language Processing within the PARLI Project*, pp. 71–89, Cham. Springer International Publishing.

Alam T., Khan A. and Alam F. (2020). *Punctuation Restoration using Transformer Models for High-and Low-Resource Languages*. In *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*, pp. 132–142, Online. Association for Computational Linguistics.

Biewald L. (2020). *Experiment Tracking with Weights and Biases*. Software available from `wandb.com`.

Che X., Wang C., Yang H. and Meinel C. (2016). *Punctuation Prediction for Unsegmented Transcript Based on Word Vector*. In Calzolari N., Choukri K., Declerck T., Goggi S., Grobelnik M., Maegaard B., Mariani J., Mazo H., Moreno A., Odijk J. and Piperidis S. (eds.), *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. European Language Resources Association (ELRA).

Chen Y. and Shi X. (2020). *Improving Machine Simultaneous Interpretation by Punctuation Recovery*. „Journal of Computer Applications", 40(4), p. 972–977.

Courtland M., Faulkner A. and McElvain G. (2020). *Efficient Automatic Punctuation Restoration Using Bidirectional Transformers with Robust Inference*. In *Proceedings of the 17th International Conference on Spoken Language Translation*, pp. 272–279, Online. Association for Computational Linguistics.

Falcon W. A. (2019). *PyTorch Lightning*. `https://github.com/PyTorchLightning/pytorch-lightning`.

Gale W. and Parthasarathy S. (2017). *Experiments in Character-Level Neural Network Models for Punctuation*. In *Proceeedings of Interspeech 2017*, pp. 2794–2798.

Gangi M. A. D., Enyedi R., Brusadin A. and Federico M. (2019). *Robust Neural Machine Translation for Clean and Noisy Speech Transcripts*. arXiv:1910.10238.

Gravano A., Jansche M. and Bacchiani M. (2009). *Restoring Punctuation and Capitalization in Transcribed Speech*. „Procedings of 2009 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)", pp. 4741–4744.

Klejch O., Bell P. and Renals S. (2017). *Sequence-to-sequence Models for Punctuated Transcription Combining Lexical and Acoustic Features*. In *Proceedings of 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5700–5704.

Kuprieiev R., Petrov D., Pachhai S., Redzyński P., da Costa-Luis C., Rowlands P., Schepanovski A., Shcheklein I., Taskaya B., Orpinel J., Santos F., Sharma A., Gao, Kaimuldenov Z., Hodovic D., Kodenko N., Grigorev A., Hathaway E., Dash N., Vyshnya G., Kulkarni M., Hora M., Sativa V., Mangal S., Baranowski W., Wolff C., Benoy K., Maslakov A. and Khamutov A. (2021). *DVC: Data Version Control — Git for Data & Models*. `10.5281/zenodo.5514325`.

Lhoest Q., del Moral A. V., von Platen P., Wolf T., Jernite Y., Thakur A., Tunstall L., Patil S., Drame M., Chaumond J., Plu J., Davison J., Brandeis S., Scao T. L., Sanh V., Xu K. C., Patry N., McMillan-Major A., Schmid P., Gugger S., Liu S., Raw N., Lesage S., Matussière T., Debut L., Bekman S. and Delangue C. (2021). *huggingface/datasets: 1.12.1*. `10.5281/zenodo.5510481`.

Lu W. and Ng H. T. (2010). *Better Punctuation Prediction with Dynamic Conditional Random Fields*. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pp. 177–186, Cambridge, MA. Association for Computational Linguistics.

Mroczkowski R., Rybak P., Wróblewska A. and Gawlik I. (2021). *HerBERT: Efficiently Pretrained Transformer-based Language Model for Polish*. In *Proceedings of the 8th Workshop on Balto-Slavic Natural Language Processing*, pp. 1–10, Kiyv, Ukraine. Association for Computational Linguistics.

Nagy A. M., Bial B. and Ács J. (2021). *Automatic Punctuation Restoration with BERT Models*. arXiv:2101.07343.

Paszke A., Gross S., Massa F., Lerer A., Bradbury J., Chanan G., Killeen T., Lin Z., Gimelshein N., Antiga L., Desmaison A., Kopf A., Yang E., DeVito Z., Raison M., Tejani A., Chilamkurthy S., Steiner B., Fang L., Bai J. and Chintala S. (2019). *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. In Wallach H., Larochelle H., Beygelzimer A., d'Alché-Buc

F., Fox E. and Garnett R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc.

Salloum W., Finley G., Edwards E., Miller M. and Suendermann-Oeft D. (2017). *Deep Learning for Punctuation Restoration in Medical Reports*. In *BioNLP 2017*, pp. 159–164, Vancouver, Canada,. Association for Computational Linguistics.

Sinclair M., Bell P., Birch A. and McInnes F. (2014). *A semi-Markov model for speech segmentation with an utterance-break prior*. In *Proceedings of Interspeech 2014*, pp. 2351–2355.

Stolcke A., Shriberg E., Bates R., Ostendorf M., Hakkani D., Plauche M., Tur G. and Lu Y. (1998). *Automatic Detection of Sentence Boundaries and Disfluencies Based on Recognized Words*. In *Proceedings of the 5th International Conference on Spoken Language Processing (ICSLP 1998)*, p. paper 0059.

Szaszák G. and Ákos Tündik M. (2019). *Leveraging a Character, Word and Prosody Triplet for an ASR Error Robust and Agglutination Friendly Punctuation Approach*. In *Proceedings of Interspeech 2019*, pp. 2988–2992.

Tilk O. and Alumäe T. (2015). *LSTM for Punctuation Restoration in Speech Transcripts*. In *Proceedings of Interspeech 2015*, pp. 683–687.

Tilk O. and Alumäe T. (2016). *Bidirectional Recurrent Neural Network with Attention Mechanism for Punctuation Restoration*. In *Proceedings of Interspeech 2016*, pp. 3047–3051.

Ueffing N., Bisani M. and Vozila P. (2013). *Improved Models for Automatic Punctuation Prediction for Spoken and Written Text*. In *Proceedings of Interspeech 2013*, pp. 3097–3101.

Łukasz Augustyniak, Szymański P., Morzy M., Żelasko P., Szymczak A., Mizgajski J., Carmiel Y. and Dehak N. (2020). *Punctuation Prediction in Spontaneous Conversations: Can We Mitigate ASR Errors with Retrofitted Word Embeddings?* In *Proceedings of Interspeech 2020*, pp. 4906–4910.

Wang F., Chen W., Yang Z. and Xu B. (2018). *Self-Attention Based Network for Punctuation Restoration*. In *Proceedings of the 24th International Conference on Pattern Recognition (ICPR 2018)*, pp. 2803–2808.

Wolf T., Debut L., Sanh V., Chaumond J., Delangue C., Moi A., Cistac P., Rault T., Louf R., Funtowicz M., Davison J., Shleifer S., von Platen P., Ma C., Jernite Y., Plu J., Xu C., Le Scao T., Gugger S., Drame M., Lhoest Q. and Rush A. (2020). *Transformers: State-of-the-Art Natural Language Processing*. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online. Association for Computational Linguistics.

Żelasko P., Szymański P., Mizgajski J., Szymczak A., Carmiel Y. and Dehak N. (2018). *Punctuation Prediction Model for Conversational Speech*. In *Proceedings of Interspeech 2018*, pp. 2633–2637.

Punctuation Restoration with Ensemble of Neural Network Classifier and Pre-trained Transformers

Michał Marcińczuk
(Samurai Labs / Wrocław University of Science and Technology)

Abstract

To solve the punctuation restoration task, we build a unary classifier based on a neural network (NN). The NN was trained to predict if there is, or not, a punctuation mark after a word. We used pre-trained attention-based language models for Polish (Polish RoBERTa and HerBERT) to encode the word as a vector of numeric values. Thanks to the attention mechanism, the vectors encoded the meaning of the word and its context. A single NN with the large variant of Polish RoBERTa achieved a score of 80.53. The final solution, which reached 81.23, was based on an ensemble of models trained with different language models and subsets of the training data.

The system is available under a BSD license at `https://github.com/mczuk/punktor`.

Keywords

natural language processing, deep learning, punctuation restoration, transformers

1. Introduction

Punctuation plays an important role in text readability and natural language processing (NLP). Missing or incorrect punctuation can change the meaning of the text or confuse the reader. Full stops, exclamation marks, and question marks help in text segmentation — splitting text paragraphs into sentences. Hyphens, commas, and colons help interpret and understand a sequence of words — commas separate a list of elements, hours and minutes might be separated by a colon, and so on. User-generated content (UCG), and the output of automatic speech recognition (ASR) might be devoided of punctuation marks. The common approach is to restore punctuation in text post-processing. Recently, transformer-based models were proved to obtain state-of-the-art results in many NLP tasks, including punctuation restoration (Alam et al. 2020, Nagy et al. 2021). In the paper, we present our effort to create a model for punctuation restoration for Polish utilizing existing transformer-based language models

and adopting a neural network architecture used for named entity recognition (Marcińczuk and Radom 2021).

2. Task description

2.1. Problem statement

The goal of the task is to insert missing punctuation marks in the text. The text is provided as a space-separated lower-cased sequence of words. After each word there can be inserted one of seven possible punctuation marks, i.e., dot (.), comma (,), question mark (?), an exclamation mark (!), hyphen (-), colon (:), ellipses (...) or blank (no punctuation). Sample input text and the corresponding expected output are presented in Figure 1.

```
Input:    w środę  23 marca  w godzinach rannych  silne trzęsienie ziemi
          dotknęło północno  wschodnie wybrzeże japonii

English:  on wednesday march 23rd in the morning a strong earthquake
          struck off the northeast coast of japan

Expected: w środę, 23 marca, w godzinach rannych, silne trzęsienie ziemi
          dotknęło północno- wschodnie wybrzeże japonii.
```

Figure 1: Sample input text and the expected output with punctuation

2.2. Datasets

Organizers of the task provided a collection of texts gathered from two sources:

— WikiTalks[1] — each article on Wikipedia has a page for discussion between users on the article's content,

— WikiNews[2] — news from the Wikinews pages.

The texts were automatically scrapped and cleaned. More than 1000 texts were randomly selected and manually corrected. The total numbers of texts in each of the datasets are provided in Table 1.

[1] https://www.wikipedia.org
[2] https://www.wikinews.org

Table 1: Distribution of texts in the train and test datasets

Dataset	WikiNews	WikiTalks	Purpose
train (raw)	13 757	8 429	training
train (corrected)	646	168	training
test-A	174	32	tuning/training
test-D	160	89	final testing

3. Punctuation restoration

3.1. Data preprocessing

For training, we used both train datasets — raw and corrected. The datasets were converted into sequences of words with a length of up to 300 words. Each word was represented by a pair of values. The first value was a word form, and the second a label corresponding to the punctuation mark or none following the token. For example, the sample sentence from Figure 1 was converted to the following representation:

```
w           NONE        trzęsienie  NONE
środę       COMMA       ziemi       NONE
23          NONE        dotknęło    NONE
marca       COMMA       północno    HYPHEN
w           NONE        wschodnie   NONE
godzinach   NONE        wybrzeże    NONE
rannych     COMMA       japonii     FULLSTOP
silne       NONE
```

Table 2 contains statistics of the punctuation marks in the train and test-A datasets.

Table 2: Distribution of punctuation marks in the datasets

Punctuation	train	test-A
full stop	362 321	2 573
comma	338 426	2 498
question mark	22 776	149
exclamation mark	5 995	23
hyphen	69 797	621
colon	40 562	323
ellipses	46	0
blank	4 522 917	34 644

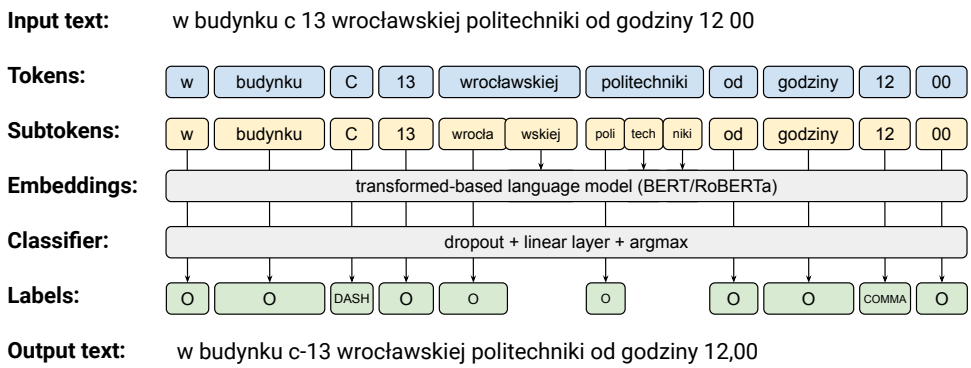

Figure 2: Architecture of the neural network for punctuation restoration

3.2. Architecture of the neural network

Our approach utilizes a token-based unary classifier. The classifier is trained to predict what type of punctuation or none follows a word in a sequence. To classify tokens, we used a single-layer neural network with a dropout. The words were transformed into vectors with the use of transformer-based language model.

In the initial experiment, we used *train* and *test-A* datasets to estimate the learning speed, i.e., how many iterations over the training dataset are required to obtain the best F1 score. Table 3 contains the parameters of the neural network architecture and the training process.

Table 3: Training parameters

Parameter	**Value**
sequence length	512
learning rate	0.00006
optimizer	Adam with weight decay (Loshchilov and Hutter 2019)
scheduler	constant
batch size	32
language models	Polish RoBERTa (Dadas et al. 2020), HerBERT (Mroczkowski et al. 2021)
language models variants	base, large
dropout	0.2
gradient accumulation steps	4
epochs	5

The results in the form of F1 score are presented in Figure 3.

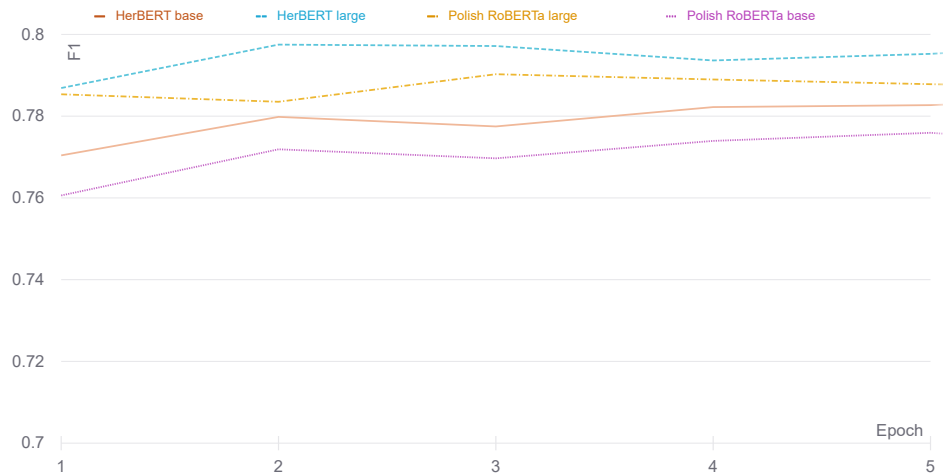

Figure 3: F1 scores on the test-A dataset for various language models and epochs

The experiment showed that:

1. 2–3 epochs were sufficient to maximize the F1 score on the test-A dataset.
2. Both large variants of the language models (HerBERT and Polish RoBERTa) obtained better results than the base model by a margin of two percentage points.
3. The HerBERT large model obtained the best F1 score.

To combine the information from different language models, we used an ensemble of three base models:

— M1 — HerBERT large,

— M2 — Polish RoBERTa,

— M3 — HerBERT base.

We trained each base model for three epochs — the number of epochs was estimated in the initial experiment — on the combined datasets of train and test-A. Then, the outputs were combined using majority voting. In the case of a tie, the output of the model with the highest score on the test set was taken (M1).

4. Evaluation

Table 4 contains the evaluation results on the test-D dataset for each model separately and the ensemble of all models. The best score was obtained by the model trained on the combined datasets and the Polish RoBERTa large language model. The ensemble of base models improved the final score by 0.7 pp — from 80.53 to 81.23.

Table 4: Evaluation of the models on the test-D dataset

	M1	**M2**	**M3**	**Ensemble**
full stop	87.51	88.03	85.49	88.47
comma	77.63	74.86	74.20	77.08
question mark	77.22	79.85	75.10	79.84
exclamation mark	20.00	20.00	36.36	20.00
hyphen	64.43	60.48	63.41	66.30
colon	76.43	79.83	74.94	79.73
ellipses	0.00	0.00	0.00	0.00
all	80.53	79.78	78.17	81.23

5. Summary

In the paper we presented a system for punctuation restoration based on an ensemble of neural network classifiers. We tackled the problem as a sequence classification task, where the token label represents the presence or absence of a punctuation mark. To encode the sequence of words, we used transformer-based pre-trained models available for the Polish language. A single base model obtained 80.53 of F1 measure, and the ensemble of three base models trained with various language models reached 81.23.

References

Alam T., Khan A. and Alam F. (2020). *Punctuation Restoration Using Transformer Models for High- and Low-Resource Languages*. In *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*, pp. 132–142, Online. Association for Computational Linguistics.

Dadas S., Perełkiewicz M. and Poświata R. (2020). *Pre-training Polish Transformer-Based Language Models at Scale*. In Rutkowski L., Scherer R., Korytkowski M., Pedrycz W., Tadeusiewicz R. and Zurada J. M. (eds.), *Proceedings of 19th International Conference on Artificial Intelligence and Soft Computing (ICAISC 2020), Part II*, pp. 301–314. Springer International Publishing.

Loshchilov I. and Hutter F. (2019). *Decoupled Weight Decay Regularization*. arXiv:1711.05101.

Marcińczuk M. and Radom J. (2021). *A Single-run Recognition of Nested Named Entities with Transformers*. „Procedia Computer Science", 192, p. 291–297. Proceedings of the 25th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES 2021).

Mroczkowski R., Rybak P., Wróblewska A. and Gawlik I. (2021). *HerBERT: Efficiently pretrained transformer-based language model for Polish*. In *Proceedings of the 8th Workshop on Balto-Slavic Natural Language Processing*, pp. 1–10, Kiyv, Ukraine. Association for Computational Linguistics.

Nagy A., Bial B. and Ács J. (2021). *Automatic Punctuation Restoration with BERT Models*. arXiv:2101.07343.

Punctuation Restoration from Read Text with Transformer-based Tagger

Tomasz Ziętkiewicz (Adam Mickiewicz University / Samsung Research Poland)

Abstract

This paper presents a system developed at Adam Mickiewicz University and Samsung Research Poland for submission to PolEval[1] 2021, Task 1: Punctuation restoration from read text. The purpose of this task was to: "restore punctuation in the ASR recognition of texts read out loud".[2]

This paper describes our approach to the problem and presents the results obtained.

Keywords

natural language processing, punctuation restoral, speech processing, ASR, inverse text normalization, ASR post-processing

1. Introduction

Punctuation plays an important role in comprehending written text by humans (Moore 2016). It not only makes reading easier and faster, but sometimes can change a meaning of a sentence. It has been shown, that people do care about correct normalization of text they read and that they prefer a text with correct punctuation (Suter and Novak 2021). As it is important to human readers, it can also be crucial for NLP systems, as it can, for example, improve results of syntactic parsing (Jones 1995).

Punctuation marks are not directly pronounced, and commonly ASR models are trained using reference texts without punctuation marks. Therefore, text returned by speech recognition does not contain punctuation characters, and they have to be re-inserted in the post-processing step.

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

The problem of restoring punctuation from read text can be seen as a specific subproblem of postediting speech recognition results, which was one of the tasks of Poleval 2020 competition (Ogrodniczuk and Kobyliński 2020). The punctuation restoral model presented in this paper can be seen as a modified version of a postediting model proposed by the authors in a publication presenting their submission to Poleval 2020 (Ziętkiewicz 2020).

2. Related work

Punctuation restoral is a subclass of a wider task of inverse text normalization, also called text denormalization. The task includes, besides restoring the punctuation characters, restoration of capitalization, inverse normalization of numeric forms, formatting dates and times. In some works these tasks are modelled jointly and others focus on one specific problem like punctuation restoral. Besides rules-based methods, there are two dominating approaches to the problem: sequence labeling or sequence-to-sequence. In the first one, each token is assigned a class indicating which (if any) punctuation characters should be added to the token. The latter one resembles a machine translation task, where the normalized text is treated as a source language and denormalized text is treated as a target language. One of the most recent examples of the latter approach can be found in (Suter and Novak 2021), where authors tackle full denormalization with transformer model. Because the approach doesn't label separate tokens but "translates" whole sentences, the metric reported by the authors is Word Error Rate (WER) rather than precision and recall. The reported WER values for just punctuation characters range from 13.02 to 16.49, depending on the language being modelled.

One of examples of the sequence labeling approach can be found in (Żelasko et al. 2018), where authors propose a punctuation restoral model for English. Best of the reported F1 scores, achieved with BiLSTM architecture, with added timing information are as follows: 67.3, 59.2, 66.1 for dot, question mark and comma respectively.

3. Data

Dataset shared by the organizers, called WIKIPUNCT, consists of 1000 documents from two sources: Polish Wikinews[3] (800 documents) and Polish Wikipedia Talk pages called WikiTalks (200 documents). Wikinews is a free-content news page with news articles written by users. Documents from this source, by their nature, are characterized by high relative quality of punctuation and orthography. The second source, WikiTalks, is a collection of discussions over Wikipedia articles carried out by their authors and editors. As such, the style of these documents is much less formal and their relative quality regarding punctuation and orthography is low.

Datasets are divided into train and test-A subsets. Their statistics are shown in Table 1.

[3] https://pl.wikinews.org/

Table 1: Statistics of datasets provided by organizers

	train	**test-A**
Documents	800	200
Sentences*	11380	2722
Tokens	165 913	40 842
Max document length (tokens)	313	339
Doc length (sent) (min/mean/max)	2/14/36	3/14/28
!	118	23
?	797	149
:	913	323
-	2448	621
,	10 132	2498
.	10 465	2573

Each subset is shared in 3 formats:

— Text with punctuation

— Text without punctuation

— Tokenized text with/without punctuation with forced time alignments for each token

Additionally, 3 more test sets were prepared: test-B, test-C and test-D, but their expected forms were not disclosed to the participants as they were used only for evaluation of submissions.

4. Method

Similarly to the method proposed by Ziętkiewicz (2020), we use a tag-apply approach. Training of a model in this approach consists of the following steps: input and expected output sequences are compared. Based on differences found, edit operation tags are added to the input, describing how to transform it into expected output. This way, a corpus of input sequences tagged with operation tags is created. The corpus is then used to train a tagging model. On inference time, an input sequence is tagged with edit operations by the model. The edit operations are then applied to the input sentence. In case of punctuation restoral problem, edit operations correspond to the punctuation marks to be inserted into plain text.

4.1. Data preparation

Training and test data shared by the organizers is already normalized and tokenized. Contrary to the error correction task described by Ziętkiewicz (2020) there is no need to compare the input and expected text, as the only difference between them is the presence or lack of punctuation characters. Regular expressions are used to match the punctuation in the reference text. Each detected punctuation mark is removed from the text, and the token preceding it is assigned a tag corresponding to the character. In case a punctuation character

is the first in a document (there is no token preceding it), a corresponding label is assigned to a token following it, with a special prefix indicating that the character should be inserted before the token and not after. In case there are multiple characters following a certain token, their labels are joined together. This approach enables to model all possible combinations of punctuation characters in different positions in a sentence, but produces a lot of sparse labels, with very few representatives in the training set. After performing initial experiments, we decided to drop the minority classes and leave only the single punctuation marks in the training data, resulting in just 6 classes of punctuation marks to model.

4.2. Model training

Having prepared a corpus of hypothesis sentences labeled with edit operations, a tagger model can be trained. We us pre-trained herBERT (Mroczkowski et al. 2021) language model fine-tuned for token classification task by adding one linear layer on top of hidden states from the pretrained model. Precisely, BertForTokenClassification class from Hugging Face Transformers library (Wolf et al. 2020) was used.

4.3. Inference

During the inference stage, when the system is used to restore punctuation, the model is used to tag each token with either an empty tag or a tag corresponding to one of the punctuation characters. Once all tokens in a document are labeled with tags, a continuous text can be produced by inserting punctuation characters after tokens tagged with their corresponding labels. The reference text shared by the organizer was consistent in always putting a punctuation character directly after a preceding word and always inserting a space after punctuation characters (even in cases where it shouldn't be in a correct text, i.e. colon was always followed by a space, even in time expressions like "10: 45"). Thanks to this simplification, there was no need to distinguish between cases where space should and should not be inserted after or before a punctuation character. In a real-world scenario this wouldn't be the case and either the model would have to learn a wider class of punctuation characters (every character with and without surrounding spaces), or handcrafted rules would have to be used at inference time to tackle the problem.

5. Results

The only metric used to officially compare submissions to the task was a weighted average F1-score for all punctuation characters, given by the following formula:

$$\frac{1}{n} \sum_{p \in Punctuation} support(p) \times avg_{micro} F_1(p)$$

$$avg_{micro} F_1(p) = 2 \times \frac{P_p \times R_p}{P_p + R_p}$$

Table 2: Results on test-A dataset

Tag	TP	TN	FP	FN	Support	Precision	Recall	F1
!	1	16140	0	12	13	1.000	0.077	0.143
,	785	14894	273	201	986	0.742	0.796	0.768
-	135	15862	62	94	229	0.685	0.590	0.634
.	1001	14937	150	65	1066	0.870	0.939	0.903
:	58	16037	29	29	87	0.667	0.667	0.667
?	40	16079	11	23	63	0.784	0.635	0.702
None	13362	2198	246	347	13709	0.982	0.975	0.978
Weighted average including None class						0.952	0.952	0.952
Weighted average without None class						0.793	0.826	0.806

$$P_p = avg_{micro} Precision(p) = \frac{\sum_{d \in Documents} TP}{\sum_{d \in Documents} TP + FP}$$

$$R_p = avg_{micro} Recall(p) = \frac{\sum_{d \in Documents} TP}{\sum_{d \in Documents} TP + FN}$$

where

— TP is a number of true positive examples for given punctuation character

— FP is a number of false positive examples for given character

— FN is a number of false negative examples for given character.

Internal results achieved by the proposed system on test-A test set are shown in Table 2. Results have been acquired by evaluating the tagger model on the tagged version of the test set (see Section 4.1 for details).

The final model, used to produce the submission uploaded for the challenge, was trained on concatenated train and test-A datasets.

According to official Poleval 2021 results[4], our submission to the shared task achieved lower scores on both test-A (78.55) and test-D (78.34) test sets, than in our internal evaluation. As results on test-A and test-D datasets doesn't differ significantly, the difference is not caused by overtraining. What is supposedly causing the difference are differences in evaluation method used by authors and the fact, that our model has been trained on simplified data ignoring rare punctuation classes like ellipsis or combinations of two or more punctuation characters.

In comparison with other participant's submissions, the proposed one achieve slightly worse results than most of them. This may be caused by the fact, that our model works on text-only data, not making use of the timing information. On the other hand, even the best of the results (81.20 F1-score) does not seem good enough for production settings. This is probably caused by low quality of part of the data (WikiTalks) and relatively small size of the training set.

[4] `http://poleval.pl/results/`

Presumably with better quality and larger training set the results achieved with the proposed approach would be more suitable for practical applications.

6. Conclusion

The proposed model of punctuation restoration from read text, while working on text-only data, achieves results comparable with winning submission of the task. Results indicate, that performance could be improved by providing more high-quality training data. Results also indicate that regardless of the method used, the provided training data is not enough to develop a system suitable for application in production settings.

References

Jones B. E. M. (1995). *Exploring the Role of Punctuation in Parsing Natural Text*. arXiv:cmp-lg/9505024.

Moore N. (2016). *What's the Point? The Role of Punctuation in Realising Information Structure in Written English*. „Functional Linguistics", 3(6), p. 1–23.

Mroczkowski R., Rybak P., Wróblewska A. and Gawlik I. (2021). *HerBERT: Efficiently Pretrained Transformer-based Language Model for Polish*. In *Proceedings of the 8th Workshop on Balto-Slavic Natural Language Processing*, pp. 1–10, Kiyv, Ukraine. Association for Computational Linguistics.

Ogrodniczuk M. and Kobyliński Ł., editors (2020). *Proceedings of the PolEval 2020 Workshop*, Warsaw, Poland. Institute of Computer Science, Polish Academy of Sciences.

Suter B. and Novak J. (2021). *Neural Text Denormalization for Speech Transcripts*. In *Proceedings of Interspeech 2021*, pp. 981–985.

Wolf T., Debut L., Sanh V., Chaumond J., Delangue C., Moi A., Cistac P., Rault T., Louf R., Funtowicz M., Davison J., Shleifer S., von Platen P., Ma C., Jernite Y., Plu J., Xu C., Le Scao T., Gugger S., Drame M., Lhoest Q. and Rush A. (2020). *Transformers: State-of-the-Art Natural Language Processing*. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online. Association for Computational Linguistics.

Ziętkiewicz T. (2020). *Post-editing and Rescoring of ASR Results with Edit Operations Tagging*. In Ogrodniczuk M. and Kobyliński Ł. (eds.), *Proceedings of the PolEval 2020 Workshop*, pp. 23–31, Warsaw, Poland. Institute of Computer Science, Polish Academy of Sciences.

Żelasko P., Szymański P., Mizgajski J., Szymczak A., Carmiel Y. and Dehak N. (2018). *Punctuation Prediction Model for Conversational Speech*. In *Proceedings of Interspeech 2018*, pp. 2633–2637.

PolEval 2021 Task 2: Evaluation of Translation Quality Assessment Metrics

Krzysztof Wołk (Polish-Japanese Academy of Information Technology)

Maciej Szymkowski (Białystok University of Technology)

Abstract

This paper summarizes the second evaluation task during the 2021 edition of PolEval[1]. Task two involved the automatic evaluation of computer-generated texts. Such texts include, but are not limited to, machine translation results, dialogue system responses, synthetically generated texts, etc. The task was based on samples obtained from machine translation, which were evaluated in two ways. In one of the corpora, a reference corpus was available, while the other corpus did not have this convenience. The experiment was conducted for the Polish language.

Keywords

machine translation, question answering, translation evaluation, blind evaluation

1. Introduction

Evaluation of computer-generated texts is a major problem and challenge for automated algorithms. At the same time, it is crucial for modern systems that process natural human speech. When training various systems based on machine learning, it is important to be able to quickly and cheaply evaluate the performance of different versions. Currently, most algorithms are evaluated based on a reference corpus. This approach, although cost- and time-efficient, is quite problematic since it assumes that only one translation is correct. However, this assumption is top-down wrong, because if one gave the same piece of text to several people to translate, the translations would differ from each other even though all versions would be correct.

Therefore, task two on PolEval 2021 had two paths. The first one was to develop an algorithm for Polish language that would perform evaluation based on a reference set. In the second

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

approach, the reference set was not provided. Both tasks were made available in parallel and were independently evaluated. This is also motivated by the fact that in practice researchers and developers encounter data of both types. Although the experiments were performed on samples resulting from machine translation, the concept and methodology itself is much more widely applicable. Namely, in addition to machine translation, dialogue systems, ASR scores, phraseological competence of foreigners, or other computer-generated texts, among others, can be evaluated.

2. Task definition

The task was to investigate metrics for automatic evaluation of machine translation results (Han and Wong 2016, Maruf et al. 2019) or other similar data types (Wołk and Koržinek 2016, Celikyilmaz et al. 2020). We have prepared translations from English to Polish along with reference translations made by a human — a Polish native speaker. In the task we were looking for automatic metrics for evaluation. The task was evaluated at the level of segments often similar to sentences. The results of the task were calculated correlations of the submitted scores with the human evaluations performed manually.

The goal of the task was to achieve the strongest possible correlation with human evaluation of translation quality (average of 6 human ratings), to illustrate and evaluate the usefulness of automatic evaluation metrics as a substitute or supplement to human evaluation, to move automatic evaluation from the level of popular system ranking to a more detailed ranking at the sentence level, to analyze the impact of access to reference data on system evaluation and finally to analyze how well automatic metrics evaluate human translations. The task was be divided into two competitions:

— Task 2A: an evaluation in which reference data, i.e. source data (EN), translated by MT(PL) system translated by human was available for comparison. The data was parallel at the sentence level.

  The data has been published in PolEval repository (`https://github.com/poleval/2021-quality-estimation-nonblind`). Gold-standard test data annotation has been published in its "secret" branch (`https://github.com/poleval/2021-quality-estimation-nonblind/tree/secret/test-B`).

— Task 2B: blind evaluation, in which only the resulting data from machine translation (MT) was available.

  The data has been published in PolEval repository (`https://github.com/poleval/2021-quality-estimation-blind`). Gold-standard test data annotation has been published in the "secret" branch (`https://github.com/poleval/2021-quality-estimation-blind/tree/secret/test-B`).

The sentences resulting from machine translation (MT) were evaluated by 6 independent native speakers and rated subjectively on a standard Likert scale (Graham et al. 2013) from 1 to 5. The evaluation scores of the same sentences by different translators were arithmetically averaged.

The following scale was used:

— 5 — the translation was very good linguistically and contains the whole content of the original sentence without adding anything;

— 4.5 — grade between 4 and 5;

— 4 — translation contains minor errors;

— 3.5 — mark bordering between 3 and 4;

— 3 — the translation contained serious errors, but can be used in practice;

— 2.5 — grade bordering between 2 and 3;

— 2 — errors make it impossible to use the translation, but it contains well-translated fragments;

— 1.5 — rating between 1 and 2;

— 1 — the translation was completely bad.

The results of this manual evaluation were compared with the submitted results of the automatic evaluation on the sentence level. This means that for each sentence/segment it was necessary to calculate the score with own metric. We assumed that a single segment is a single line in the files. The number or order of lines could not be changed. The results should have been rendered in the form of a file `out.tsv`, encoded in UTF-8, where each line will contain only a numeric value normalized to a scale from 0 to 100. The line number of the score should correspond to the line number of the evaluated segment from the test data.

We evaluated the submitted results by calculating the Pearson correlation coefficient (Benesty et al. 2009) from all samples.

3. Format of the dataset

The input file (`in.tsv`) for the non-blind version was a TSV file with the following columns:

— MT — output from a machine translation system;

— EN — source data;

— HUMAN — reference segment translated by a human;

— BLEU — BLEU scores for reference (Papineni et al. 2002)

For the blind version, only the first column was given.

4. Baseline evaluation

We used the BLEU metric in the baseline system. In other words, it was the baseline score that we wanted to improve through the competition. Although the metric is not the most

advanced, it is still the most popular method for evaluating machine translation because of its simplicity. It simply works by comparing the generated text with a reference text created by a human based on n-grams. This simplicity makes this metric quick, easy, and most importantly, cheap to use. Unfortunately it doesn't do well when different text variants are generated, e.g. using synonyms.

5. Results

In Task A, 39 results from four authors were reported. These results are presented in Table 1.

Table 1: Task 2A: Results

Submitter	**Affiliation**	**test-B Pearson score**
Darek Kłeczek (Kłeczek 2021)	skok.ai	0.6137
Artur Nowakowski (Nowakowski 2021)	Adam Mickiewicz University	0.5728
Krzysztof Wróbel (Wróbel 2021)	eNeLPol UJ AGH	0.5677
Baseline	PolEval	0.4733

In Task B, 41 results from four authors were reported. These results are presented in Table 2. Krzysztof Wróbel used different variants of BLEU metric smoothing and the BERT architecture in his submissions. In turn, Darek Kłeczek performed regression based on the Herbert Large model. Artur Nowakowski based his solution on COMET framework. They described their approaches in detail in independent articles (Rei et al. 2020).

Table 2: Task 2B: Results

Submitter	**Affiliation**	**test-B Pearson score**
Krzysztof Wróbel	eNeLPol UJ AGH	0.5138
Darek Kłeczek	skok.ai	0.4840
Artur Nowakowski	Adam Mickiewicz University	0.4793
Baseline	PolEval	0.4482

6. Conclusions and discussion

For both tasks, a total of 80 solutions were submitted from 3 different institutions. These were both solutions based on adaptations of well-known metrics such as BLEU as well as more advanced solutions based on BERT architecture or COMET Framework. The winner of task A was Mr. Darek Kłeczek from skok.ai while the winner of task B was Mr. Krzysztof Wróbel from eNeLPol UJ AGH. All solutions show that the improvement in the quality of assessment in both cases can be attributed not only to the modern architectures based on neural networks but also to the already known metrics after their adaptation to the task, domain and language.

There are many research avenues to achieve the same goal, which is encouraging and opens up further research opportunities in this area. It also provides opportunities for error analysis and implementation of hybrid models.

References

Benesty J., Chen J., Huang Y. and Cohen I. (2009). *Pearson Correlation Coefficient*, pp. 1–4. Springer, Berlin, Heidelberg.

Celikyilmaz A., Clark E. and Gao J. (2020). *Evaluation of Text Generation: A Survey*. arXiv:2006.14799.

Graham Y., Baldwin T., Moffat A. and Zobel J. (2013). *Continuous Measurement Scales in Human Evaluation of Machine Translation*. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pp. 33–41, Sofia, Bulgaria. Association for Computational Linguistics.

Han A. L. and Wong D. F. (2016). *Machine Translation Evaluation: A Survey*. arXiv:1605.04515.

Kłeczek D. (2021). *Simple Recipes for Assessing Translation Quality*. In Ogrodniczuk and Kobyliński (2021), pp. 67–71.

Maruf S., Saleh F. and Haffari G. (2019). *A Survey on Document-level Machine Translation: Methods and Evaluation*. arXiv:1912.08494.

Nowakowski A. (2021). *Approaching English-Polish Machine Translation Quality Assessment with Neural-based Methods*. In Ogrodniczuk and Kobyliński (2021), pp. 73–78.

Ogrodniczuk M. and Kobyliński Ł., editors (2021). *Proceedings of the PolEval 2021 Workshop*, Warsaw, Poland. Institute of Computer Science, Polish Academy of Sciences.

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 the Association for Computational Linguistics*, pp. 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Rei R., Stewart C., Farinha A. C. and Lavie A. (2020). *COMET: A Neural Framework for MT Evaluation*. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2685–2702, Online. Association for Computational Linguistics.

Wołk K. and Koržinek D. (2016). *Comparison and Adaptation of Automatic Evaluation Metrics for Quality Assessment of Re-speaking*. arXiv:1601.02789.

Wróbel K. (2021). *Transformer as Machine Translation Evaluation Metrics*. In Ogrodniczuk and Kobyliński (2021), pp. 79–83.

Simple Recipes for Assessing Translation Quality

Dariusz Kłeczek (skok.ai)

Abstract

This paper presents my contributions to PolEval 2021 Task 2: Evaluation of translation quality assessment metrics[1]. It's a simple meal, based on a healthy dose of transformers, cooked in HuggingFace sauce and salted with some tricks learned in Kaggle competitions.

Keywords

natural language processing, automatic translation, transformers, regression

1. Introduction

In cooking, the ultimate taste of a meal is conditioned on the quality of ingredients, preparation, and some secret source added by a chef. Similarly, when cooking machine learning solutions, we can associate their performance with capacity of the model architecture, the training protocol, and techniques and tricks used by a modeler, which can be learned through research or experience in other competitions.

Before we jump into a machine learning solution, we should probably consider if it is the right way to approach the problem of assessing translation quality. For example, the classic BLEU metric (Papineni et al. 2002) is a rule based approach. The organizers specified that the goal of the task is to achieve the strongest possible correlation with human evaluation of translation quality. It is very likely that human ratings reflect their subjective assessment of translation quality, and our metric should mirror that. We are also provided with a very small dataset that can be used for training.

These factors make this task similar to some other tasks included in the KLEJ benchmark (Rybak et al. 2020) and to a recent Kaggle CommonLit Readability Prize competition[2], which are the inspiration for my machine learning solution and will be described below.

[1] `http://poleval.pl/tasks/task2`

[2] `https://www.kaggle.com/c/commonlitreadabilityprize/`

2. Previous experience

2.1. KLEJ

KLEJ benchmark consists of nine tasks used for the Polish language understanding evaluation. These tasks also require learning a function that mirrors subjective human evaluation of natural language texts. The KLEJ leaderboard is dominated by large transformer-based language models, which are natural candidates for the backbone in our machine learning solution.

2.2. Kaggle CommonLit Readability Prize competition

Kaggle recently hosted a competition with the goal to rate the complexity of English reading passages for grade 3–12 classroom use. The target value was the result of a Bradley-Terry analysis of more than 111,000 pairwise comparisons between excerpts. Teachers spanning grades 3–12 (a majority teaching between grades 6–10) served as the raters for these comparisons[3]. Similar to our task, the dataset was of modest size, and the problem could be framed as regression. Participants evaluated many solutions including rule-based metrics, but at the end simple transformer-based regression models proved to deliver the best results. A number of discussions also revealed tricks to improve the performance of these models, such as removing dropout (Deotte 2021), reinitializing top layers (Zhang et al. 2021) and frequent model evaluation (Baskaran 2021).

3. Strong backbone: Transformers

We're lucky now to have several Polish language models competing to serve as our backbone. KLEJ gives us some indication on the relative strength of these models. Given a small development set it was easy to run a comprehensive grid search in the blind setting to compare their performance on our data. I've selected the base and large HerBERT models (Mroczkowski et al. 2021), base and large variants of Polish Roberta (Dadas 2019) and my own Polbert model (Kłeczek 2020) for the comparison in Table 1. HerBERT large proved to work very well for this task and I adopted it for further experiments.

In the non-blind setting, there were additional considerations — we also had reference translation and the source text in English. In the first two experiments I compared a model using pairs of source English text and the automated translation based on XLM-Roberta-base (0.3913 test-A Pearson score) versus a model using pairs of reference and automated translation in Polish based on HerBERT-base (0.4879 test-A Pearson score). Given this difference, I only focused on the latter approach without using the source texts in English. The final submission is based on HerBERT-large model and trained on a combination of development and test-A sets.

[3] https://www.kaggle.com/c/commonlitreadabilityprize/discussion/240423

Table 1: Average RMSE from 4 BLIND training runs with different hyperparameter settings. Trained on 80% and evaluated on 20% of development set.

Model backbone	**RMSE**
Herbert base cased	0.323824
Herbert large cased	0.303575
Polish Roberta large	0.320343
Polish Roberta base	0.337123
Polbert base cased	0.325650

4. Technique and tricks

I used the HuggingFace transformers (Wolf et al. 2020) and datasets (Lhoest et al. 2021) packages in my experiments, including the model classes and training loop. I use the standard regression head on top of transformer backbone (*AutoModelForSequenceClassification*) with MSE loss function.

I re-initialize the last layer weights of the pretrained transformer model. The hypothesis is that the ultimate layers are specialized in the pre-training task, and it may be beneficial to start from standard initialization when finetuning a model for a new task.

I also set dropout across the model to zero. In the Kaggle competition mentioned above this was one of the key tricks to get regression models to have smoother learning curves.

5. Training process

5.1. Nonblind

My final and best submission for the nonblind task is a regression model based on HerBERT large cased backbone, trained with for 6 epochs with batch size 8, learning rate 2e-5, constant schedule, sequence length 192 and weight decay 0.01. I train the models in 5-fold setting on the sum of development and test-A datasets, evaluating every 10 steps and choosing the checkpoint with lowest validation RMSE score. The final predictions are an average of the 5 models, resulting in 0.6137 test-B Pearson score (winning solution).

5.2. Blind

I used the same approach in the blind version of the same task, except that I only evaluated the single automated translation text without the reference translation. The best experiment resulted in 0.4801 test-B Pearson score, and consisted of HerBERT large cased backbone, trained with for 3 epochs with batch size 8, learning rate 1e-5, constant schedule, sequence length 128 and weight decay 0.001.

I was motivated by the competitors, especially Krzysztof Wróbel with his 0.5138 test-B Pearson score, to look for ways to further improve my score. I tried to replace regression model with classification, and various post-processing methods to better align regression output with the actual score distribution, all in vain. Ultimately, there was some improvement from pseudolabeling. I back-translated 12810 short Polish texts and predicted their score with an earlier version model trained on the development and test-A datasets. I used that pseudolabeled dataset as augmentation and trained another model with this data. This resulted in 0.4793 test-B Pearson score, and when averaged with the 0.4801 model led to my best submission of 0.4840 (second place).

References

Baskaran V. (2021). *How I Improved My Transformer from 0.53 to 0.48 with Couple of Lines of Code*. `https://www.kaggle.com/c/commonlitreadabilityprize/discussion/246817`. Accessed: 2021-10-08.

Dadas S. (2019). *A repository of Polish NLP resources*. Github. Accessed: 2021-10-08.

Deotte C. (2021). *The Magic of No Dropout*. `https://www.kaggle.com/c/commonlitreadabilityprize/discussion/260729`. Accessed: 2021-10-08.

Kłeczek D. (2020). *Polbert: Attacking Polish NLP Tasks with Transformers*. In Ogrodniczuk M. and Łukasz Kobyliński (eds.), *Proceedings of the PolEval 2020 Workshop*, pp. 79–88, Warszawa. Institute of Computer Science, Polish Academy of Sciences.

Lhoest Q., del Moral A. V., von Platen P., Wolf T., Jernite Y., Thakur A., Tunstall L., Patil S., Drame M., Chaumond J., Plu J., Davison J., Brandeis S., Scao T. L., Sanh V., Xu K. C., Patry N., McMillan-Major A., Schmid P., Gugger S., Liu S., Raw N., Lesage S., Matussière T., Debut L., Bekman S. and Delangue C. (2021). *huggingface/datasets: 1.12.1*. `10.5281/zenodo.5510481`.

Mroczkowski R., Rybak P., Wróblewska A. and Gawlik I. (2021). *HerBERT: Efficiently Pretrained Transformer-based Language Model for Polish*. In *Proceedings of the 8th Workshop on Balto-Slavic Natural Language Processing*, pp. 1–10, Kiyv, Ukraine. Association for Computational Linguistics.

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*, pp. 311–318, USA. Association for Computational Linguistics.

Rybak P., Mroczkowski R., Tracz J. and Gawlik I. (2020). *KLEJ: Comprehensive Benchmark for Polish Language Understanding*.

Wolf T., Debut L., Sanh V., Chaumond J., Delangue C., Moi A., Cistac P., Rault T., Louf R., Funtowicz M., Davison J., Shleifer S., von Platen P., Ma C., Jernite Y., Plu J., Xu C., Le Scao T., Gugger S., Drame M., Lhoest Q. and Rush A. (2020). *Transformers: State-of-the-art natural language processing*. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online. Association for Computational Linguistics.

Zhang T., Wu F., Katiyar A., Weinberger K. Q. and Artzi Y. (2021). *Revisiting Few-sample BERT Fine-tuning*. In *Proceedings of the 9th International Conference on Learning Representations (ICLR 2021)*.

Approaching English-Polish Machine Translation Quality Assessment with Neural-based Methods

Artur Nowakowski (Faculty of Mathematics and Computer Science, Adam Mickiewicz University, Poznań)

Abstract

This paper presents our contribution to the PolEval 2021 Task 2: *Evaluation of translation quality assessment metrics*. We describe experiments with pre-trained language models and state-of-the-art frameworks for translation quality assessment in both *nonblind* and *blind* versions of the task. Our solutions ranked second in the *nonblind* version and third in the *blind* version.

Keywords

machine translation quality estimation, machine translation evaluation, pre-trained language models, natural language processing

1. Introduction

Machine translation quality evaluation is the task of assessing translation quality based on a reference translation. In the past, traditional machine translation evaluation metrics such as BLEU (Papineni et al. 2002), METEOR (Banerjee and Lavie 2005), or CHRF (Popović 2015) relied on lexical-level features between the machine translation hypothesis and the reference translation. They remain popular to this day due to their computational speed and the fact that they can be applied to any translation direction.

The rise of Neural Machine Translation (NMT) in recent years has shown that high-quality NMT systems are often mistreated by lexical-level evaluation metrics, as such systems can generate correct translation that is lexically distant from a reference translation.

Recent advances in the field of neural language modeling (Devlin et al. 2019, Conneau et al. 2020) led to the creation of BERT cosine similarity-based metrics, such as BERTSCORE (Zhang et al. 2020), as well as metrics trained on human judgments, such as COMET (Rei et al. 2020) and BLEURT (Sellam et al. 2020). Human judgments include manually assigned

quality scores, such as *Direct Assessment* (DA) (Graham et al. 2013), but may also be derived from post-edited translation to calculate post-editing effort in the form of *Human-mediated Translation Edit Rate* (HTER) (Snover et al. 2006).

Machine translation quality estimation (QE) is a different task than evaluation, as the goal is to predict machine translation quality without access to a reference translation. Research on QE in recent years has shown that it is possible to achieve high levels of correlation with human judgments based only on a source segment and a machine translation hypothesis (Specia et al. 2020). Existing state-of-the-art frameworks for QE include COMET (Rei et al. 2020), which allows QE models to be trained in a reference-free mode and TRANSQUEST (Ranasinghe et al. 2020), which proposes two new architectures for QE: MONOTRANSQUEST and SIAMESETRANSQUEST.

2. Task description

The goal of Task 2 is to investigate metrics for automatic evaluation of machine translation in the English-Polish translation direction.

The organizers prepared distinct datasets for *nonblind* and *blind* versions of the task. The *nonblind* dataset consists of the following data: source segment, machine translation hypothesis, reference translation, and quality score. The *blind* dataset consists only of machine translation hypothesis and its quality score. The segment quality scores were created by averaging the scores assigned by six human annotators. Unlike most of the current human judgment-based QE tasks, where scores are assigned on a continuous scale (Graham et al. 2013), the task utilizes a standard Likert scale allowing ratings from 1 to 5. The evaluation metric used in both versions of the task is Pearson's r correlation score.

The datasets were split into a development set ("dev-0") and two test sets ("test-A" and "test-B"). The first of the test sets ("test-A") was the main test set during the initial testing phase of the competition and was converted to the development set with the release of the final test set ("test-B").

Table 1 presents statistics of the provided datasets: the number of segments, the average number of source tokens, the average number of MT hypothesis tokens, the minimum segment quality score, and the average segment quality score.

Table 1: Statistics of datasets provided by organizers

	Nonblind			**Blind**		
	Dev-0	**Test-A**	**Test-B**	**Dev-0**	**Test-A**	**Test-B**
Segments	485	500	1000	485	500	1000
Avg. tokens (source)	18.22	17.36	17.73	-	-	-
Avg. tokens (MT hypothesis)	16.23	15.49	15.78	17.55	16.49	16.57
Min. score	3.0	2.58	2.92	3.08	2.67	2.0
Avg. score	4.30	4.37	4.38	4.33	4.31	4.40

3. Solutions

3.1. *Nonblind* task version solution

Our final solution to the *nonblind* version of the task is based on COMET. We used the "test-A" dataset as the training data and the "dev-0" dataset as the development data.

COMET uses pre-trained language model as the encoder for the source segment, the machine translation hypothesis, and the reference translation, which are independently encoded. Therefore, we decided to use $\text{HerBERT}_{\text{LARGE}}$ (Mroczkowski et al. 2021) as the pre-trained encoder model. We also experimented with XLM-RoBERTa (Conneau et al. 2020) (XLM-R) as the pre-trained encoder model, but the results were subpar. It is because $\text{HerBERT}_{\text{LARGE}}$ model was trained specifically for the Polish language and initialized with XLM-RoBERTa weights.

We applied gradual unfreezing and discriminative learning rates (Howard and Ruder 2018), meaning that we kept the encoder model frozen for 8 epochs while the feed-forward regressor was optimized with the learning rate of 3e−5. After 8 epochs, the entire model is fine-tuned but the learning rate is reduced to 1e−5 to avoid catastrophic forgetting. All hyperparameters used for training COMET models are presented in Table 3.

We experimented with other state-of-the-art methods for machine translation evaluation as well. We used BERTSCORE with contextual embeddings from the $\text{HerBERT}_{\text{LARGE}}$ model and found that it generates promising results given that it is based on cosine similarity and is not fine-tuned on the task data in any way.

Out of the trained metrics, we also experimented with BLEURT and TRANSQUEST with MONO-TRANSQUEST architecture. The BLEURT model was fine-tuned on the open-source *bleurt-base-128* model[1] with default hyperparameters. The TRANSQUEST model was fine-tuned on the open-source English-to-Any model pre-trained on DA[2] with default hyperparameters. TRANSQUEST is trained only on the source segment and the machine translation hypothesis and does not take into account the reference translation. The final results of all methods used in the *nonblind* version of the task are presented in Table 2.

Table 2: Results of the *nonblind* version of the task on the test-B dataset

Method	**Pearson's *r***
COMET (HerBERT)	**57.28**
COMET (XLM-R)	53.84
BLEURT	57.25
TRANSQUEST	55.70
BERTSCORE	48.74

[1] https://github.com/google-research/bleurt/blob/master/checkpoints.md
[2] https://tharindu.co.uk/TransQuest/models/sentence_level_pretrained

3.2. *Blind* task version solution

Our final solution to the *blind* version of the task is based on COMET as well.

The provided dataset contains only machine translation hypotheses in this scenario. Therefore, we decided to create synthetic source segments by back-translating the provided machine translation hypotheses into English by using the open-source OPUS-MT (Tiedemann and Thottingal 2020) NMT model[3], which is based on the Marian (Junczys-Dowmunt et al. 2018) framework.

We combined all the data from the *nonblind* dataset with the back-translated data from the *blind* dataset. Then, we randomly selected 100 segment pairs as the development set.

The model training procedure is the same as in the *nonblind* solution. The only difference is that the COMET model was trained in the reference-free mode in this scenario. Hyperparameters used for the *blind* model training are presented in Table 3.

Table 3: Hyperparameters used for training COMET models

Hyperparameter	**Nonblind model**	**Blind model**
Pre-trained encoder model	$\text{HerBERT}_{\text{LARGE}}$	$\text{HerBERT}_{\text{LARGE}}$
Optimizer	Adam (default parameters)	AdamW (default parameters)
Learning rate	3e−5 and 1e−5	3.1e−5 and 1e−5
Layer-wise decay	–	0.95
Num. of frozen epochs	8	0.3
Batch size	4	2
Accumulated gradient batches	2	4
Loss function	MSE	MSE
Dropout	0.15	0.15
Feed-forward hidden units	4096, 2048	2048, 1024
Feed-forward activation function	Tanh	Tanh

In this version of the task, we also conducted experiments using TRANSQUEST. TRANSQUEST model architecture, hyperparameters, and used pre-trained model were the same as in the solution to the *nonblind* version of the task. The final results of all methods used in the *blind* version of the task are presented in Table 4.

Table 4: Results of the *blind* version of the task on the test-B dataset

Method	**Pearson's *r***
COMET (HerBERT)	**47.93**
COMET (XLM-R)	43.52
TRANSQUEST	41.71

[3] `https://huggingface.co/Helsinki-NLP/opus-mt-pl-en`

4. Conclusions

We presented our contribution to the PolEval 2021 Task 2: *Evaluation of translation quality assessment metrics*.

The experiments consisted in comparing state-of-the-art methods for translation quality assessment in the English-Polish translation direction. The final solutions are based on the COMET framework. The solutions achieved second and third place in the *nonblind* and *blind* versions of the task, respectively. In the *blind* version of the task, we presented a procedure for creating a synthetic source segment input by back-translating machine translation hypothesis. All of the described methods are also worth further investigation in future experiments, as they generate competitive results.

The code and models used for creating the solutions are open-source and available on GitHub[4].

References

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.

Conneau A., Khandelwal K., Goyal N., Chaudhary V., Wenzek G., Guzmán F., Grave E., Ott M., Zettlemoyer L. and Stoyanov V. (2020). *Unsupervised Cross-lingual Representation Learning at Scale*. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 8440–8451, Online. Association for Computational Linguistics.

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

Graham Y., Baldwin T., Moffat A. and Zobel J. (2013). *Continuous Measurement Scales in Human Evaluation of Machine Translation*. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pp. 33–41, Sofia, Bulgaria. Association for Computational Linguistics.

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, Melbourne, Australia. 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:*

[4] `https://github.com/arturnn/poleval2021-qe`

Fast Neural Machine Translation in C++. In *Proceedings of ACL 2018, System Demonstrations*, pp. 116–121, Melbourne, Australia. Association for Computational Linguistics.

Mroczkowski R., Rybak P., Wróblewska A. and Gawlik I. (2021). *HerBERT: Efficiently Pretrained Transformer-based Language Model for Polish*. In *Proceedings of the 8th Workshop on Balto-Slavic Natural Language Processing*, pp. 1–10, Kiyv, Ukraine. Association for Computational Linguistics.

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 the Association for Computational Linguistics*, pp. 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Popović M. (2015). *chrF: Character n-gram F-score for Automatic MT Evaluation*. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pp. 392–395, Lisbon, Portugal. Association for Computational Linguistics.

Ranasinghe T., Orasan C. and Mitkov R. (2020). *TransQuest: Translation Quality Estimation with Cross-lingual Transformers*. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 5070–5081, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Rei R., Stewart C., Farinha A. C. and Lavie A. (2020). *COMET: A Neural Framework for MT Evaluation*. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2685–2702, Online. Association for Computational Linguistics.

Sellam T., Das D. and Parikh A. (2020). *BLEURT: Learning Robust Metrics for Text Generation*. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7881–7892, Online. 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 the 7th Conference of the Association for Machine Translation in the Americas: Technical Papers*, pp. 223–231, Cambridge, Massachusetts, USA. Association for Machine Translation in the Americas.

Specia L., Blain F., Fomicheva M., Fonseca E., Chaudhary V., Guzmán F. and Martins A. F. T. (2020). *Findings of the WMT 2020 Shared Task on Quality Estimation*. In *Proceedings of the Fifth Conference on Machine Translation*, pp. 743–764, Online. Association for Computational Linguistics.

Tiedemann J. and Thottingal S. (2020). *OPUS-MT – Building Open Translation Services for the World*. In *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation*, pp. 479–480, Lisboa, Portugal. European Association for Machine Translation.

Zhang T., Kishore V., Wu F., Weinberger K. Q. and Artzi Y. (2020). *BERTScore: Evaluating Text Generation with BERT*. In *Proceedings of the 8th International Conference on Learning Representations (ICLR 2020)*. OpenReview.net.

Transformer as Machine Translation Evaluation Metrics

Krzysztof Wróbel (Enelpol / Jagiellonian University / AGH University of Science and Technology)

Abstract

This paper presents a contribution[1] to PolEval 2021 Task 2: Evaluation of translation quality assessment metrics. The system was scored first in subtask 2B and achieved Pearson correlation coefficient of 0.51 (compared to 0.48 of the second place solution).

Keywords

natural language processing, machine translation, evaluation metric

1. Introduction

Machine translations are not perfect and their output has a different quality. It is time-consuming to verify the automatic translation and correct them. This paper shows attempts for scoring translation quality with transformer models (Vaswani et al. 2017).

2. Data

The task has 2 subtasks: blind and nonblind. In nonblind subtasks input data has 3 values: original English sentence, machine-translated sentence in Polish, sentence in Polish translated by a human. In blind subtasks only machine-translated sentences are present. For each task, 3 subcorpora were shared: dev (485 sentences), test-A (500 sentences), and test-B (1000 sentences with unknown answers during competition). We can use nonblind data in the blind task to increase training data size.

[1] `https://github.com/enelpol/poleval2021-task2`

Machine-translated sentences were scored by 6 independent native speakers on a standard Likert scale from 1 to 5 and averaged. For model training purposes the scores were normalized to [0,1] range.

Additional corpus was created by aligning English and Polish subtitles from TED Talks using timestamps. Only utterances that start with capital letters and end with a dot, question mark, or exclamation mark were used. This corpus was machine-translated and additionally, the same sentences as shared in the competition were removed resulting in over 80 thousand sentences.

3. Evaluation

Results are calculated using Pearson correlation coefficient from all samples. Pearson correlation coefficient is invariant under separate changes in location and scale in the two variables, so the model outputs do not need to be rescaled to 1-5 scale.

4. Methods

Using dataset nonblind/dev evaluation of BLEU (Papineni et al. 2002) metrics with different smoothing methods was performed. The best method was used to score sentences in the TED Talks corpus.

Both tasks were treated as text regression problems using transformer models. In the case of a nonblind task, two texts are concatenated using a separator. The hidden state of the [CLS] token is taken as input to one dense layer with linear activation.

5. Experiments

Two pre-trained transformer models were examined: HerBERT (Mroczkowski et al. 2021) large for Polish texts, and XLM-RoBERTa large (Conneau et al. 2020) for mix of English and Polish texts.

5.1. Blind

Model HerBERT large was trained on Ted Talks corpus and further finetuned on blind/dev, nonblind/dev, and nonblind/test-A with validation on blind/test-A (submission B1).

5.2. Nonblind

Unfortunately, training in different combinations of English sentences, machine-translated and human-translated sentences did not give better results than the BLEU metric.

6. Results

Table 1 shows Pearson scores for different smoothing methods available in NLTK toolkit. The best BLEU smoothing method for all datasets is the second method, which adds 1 to both numerator and denominator.

Table 1: Pearson scores on different datasets for GLEU and different BLEU smoothing methods

Method	**dev**	**test-A**	**test-B**
BLEU 0	0.37	0.31	0.32
BLEU 1	0.48	0.41	0.41
BLEU 2	**0.59**	**0.50**	**0.51**
BLEU 3	0.53	0.45	0.46
BLEU 4	0.48	0.41	0.42
BLEU 5	0.46	0.41	0.40
BLEU 7	0.48	0.43	0.42
GLEU	0.53	0.46	0.45

Submissions sent to PolEval competition:

- Baseline 1 – BLEU
- B1 – output from a blind model
- M1/M2 – average from a few models trained on pairs: machine and human translated sentences

Table 2 shows results in blind subtask. Probably additional training data allowed for higher scores than competitors.

Table 2: Results of PolEval competition in blind subtask

	test-B Pearson
M1	0.5076
B1	**0.5138**
Darek Kłeczek	0.4840
Artur Nowakowski	0.4793

Table 3 shows results in nonblind subtask. In this subtask, additional training data did not help as in blind subtask – it needs further investigation.

Table 3: Results of PolEval competition in nonblind subtask

	test-B Pearson
Baseline 1 (BLEU)	0.5103
B1	0.5616
M2	0.5676
M1	0.5677
Artur Nowakowski	0.5728
Darek Kłeczek	**0.6137**

7. Conclusions

The problem of machine translation evaluation is difficult as usually the same start-of-the-art pre-trained models are used for both tasks: machine translation and its evaluation. The additional training data helped to win the blind subtask.

Acknowledgments

This work was supported in part 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" and in part by the PLGrid Infrastructure.

References

Conneau A., Khandelwal K., Goyal N., Chaudhary V., Wenzek G., Guzmán F., Grave E., Ott M., Zettlemoyer L. and Stoyanov V. (2020). *Unsupervised Cross-lingual Representation Learning at Scale*. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 8440–8451, Online. Association for Computational Linguistics.

Mroczkowski R., Rybak P., Wróblewska A. and Gawlik I. (2021). *HerBERT: Efficiently Pretrained Transformer-based Language Model for Polish*. In *Proceedings of the 8th Workshop on Balto-Slavic Natural Language Processing*, pp. 1–10, Kiyv, Ukraine. Association for Computational Linguistics.

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 the Association for Computational Linguistics*, pp. 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser L. and Polosukhin I. (2017). *Attention is All You Need*. In *Proceedings of the 31st International Conference on*

Neural Information Processing Systems (NIPS 2017), p. 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

PolEval 2021 Task 3: Post-correction of OCR Results

Łukasz Kobyliński, Witold Kieraś (Institute of Computer Science, Polish Academy of Sciences)

Szymon Rynkun (University of Warsaw)

Abstract

This paper presents Task 3 of PolEval 2021[1] competition. We discuss the motivation, development of the task and the evaluation results of the submitted solutions. Post-correction of OCR results task focused on the premise that modern deep learning architectures can be used to train models used for correcting OCR output. To create an environment for such an evaluation campaign we have collected scanned pages of Polish language books, their raw OCR output and the manually corrected source text. We have also preprocessed the text and provided an alignment between the raw OCR output and the corrected text, as well as proposed an evaluation metric for potential solutions.

Keywords

optical character recognition, natural language processing, post-editing, OCR correction

1. Introduction

Although Optical Character Recognition methods have a long history of development, their performance is still limited, especially in the case of low-quality source material, historical texts and non-standard document layouts. The extent to which such methods may be improved based on methods relying on image analysis is also limited, as some parts of source documents might simply be unrecognizable because of visual defects or distortions in data. Low quality of document scan, distortions, unusual font can all contribute to unsatisfying OCR results. Apart from those natural difficulties, another obstacle, perhaps not mentioned as often because of its smaller relevance to average user, is disambiguation between meaningful text and page metadata such as page number, current chapter title, stamp marks or decoration, with the

[1] http://poleval.pl

latter sometimes being recognized as uninterpretable string of symbols. This problem is especially apparent when running (automated) series of scans.

In another scenario, the quantity of the training data might not allow to train a representative model for the specific language variant used in a set of documents. For example, in the comparison of recognition results of two popular OCR engines published by the IMPACT project[2] the recognition rate for historical documents (published before 1850, in Polish) was 80% on character level and ca. 60% on word level. Such accuracy levels are far from satisfactory in most applications as they result in error propagation in further stages of text processing.

On the other hand, we may observe considerable progress in Natural Language Processing methods, especially in the area of language modeling approaches based on deep neural networks. Models, such as BERT and its variants, prove to increase the accuracy of many NLP-related tasks, such as named entity recognition, fake news detection, or question answering. Thus, although some of the problems concerning OCR mentioned earlier might be solved by working on scanner mechanism itself, post-editing is another viable solution. In that approach, instead of modifying the OCR system, a new tool is created which takes as input the raw output from an OCR scanner and corrects the mistakes it has made.

The motivation of this task is thus to evaluate the possible improvement in OCR results by using modern NLP approaches to language error correction. The idea is not new, various approaches to correcting OCR output have also been evaluated during the ALTA 2017 shared task (Mollá-Aliod and Cassidy 2017). In a similar task that has been organized as part of ICDAR2017 (Chiron et al. 2017) and later during ICDAR2019 (Rigaud et al. 2019) conference[3], the improvement of OCR results reported by the best overall performer submitted for the challenge ranged from 6% for Spanish to 24% for German[4]. For some specific languages and submitted methods the results were even better, such as 44% for Finnish and 26% for French.

In the proposed task we expect the participants to create a system for transforming a text output from OCR scanner into a text that is free of errors and resembles a text corrected by human annotators as closely as possible. Contestants were given a set of raw OCR scanner outputs alongside the correct transcription of scanned material. The data used in this task is described in further detail in Section 2, the chosen method of evaluation in Section 3, while the results of the submitted systems are summarized in Section 4.

2. Data

The dataset contains Polish language books, published in the period of 1791–1998, which are currently freely licensed. The source material has been adapted from the Wikisource project by collecting original scans and their manual transcriptions from Wikisource and using Tesseract OCR engine to generate (potentially erroneous) text versions of the original scans.

[2] https://www.digitisation.eu/fileadmin/Tool_Training_Materials/Abbyy/PSNC_Tesseract-FineReader-report.pdf

[3] https://sites.google.com/view/icdar2019-postcorrectionocr

[4] https://drive.google.com/file/d/15mxNO-M9PiXBnffi7MOa8wUw33nj1xBp/view

The original page number, book ID and book publishing date (not available for some books) is included in the dataset and may be included in the trained model. Only pages with at least 150 characters are included in order to filter out front pages, covers, titles etc. which are not the target of this task.

For the purposes of the challenge we provide a training dataset, containing reference transcriptions, the material produced by the OCR engine. We also provide a development dataset, which may be used to analyze the performance of a particular model in detail. For both of these datasets an alignment between gold-standard data and OCR-results is also provided. The test-A dataset, for which gold-standard data was not originally provided has been used to generate the live leaderboard of the challenge. The final evaluation has been performed on the test-B dataset, which was revealed ca. one week before the deadline for submissions.

The whole set contains 979 books with a total number of almost 69 000 pages. The year of publication of the selected books ranges from 1791 to 1998, with the majority of them coming from the years 1870–1940. The training set contains roughly 46 000 pages, while the development set contains 5500 pages. A histogram of the number of pages with regard to the year of book's publication has been presented in Figure 1.

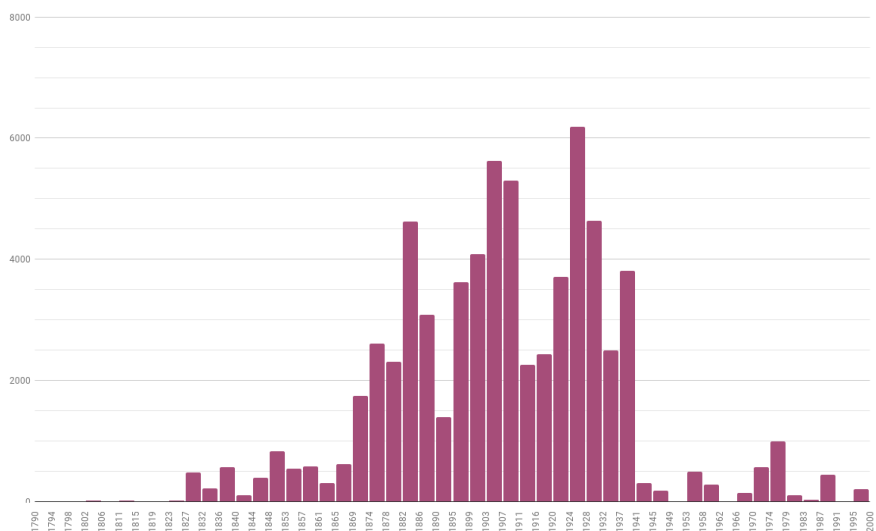

Figure 1: Number of pages per year of publication

Each text is given as a separate line with 4 items separated by TABs:

1. Document ID
2. Page number
3. Year (sometimes unspecified)
4. Text to be corrected (output of an OCR system)

The data has been published in the following repository: `https://github.com/poleval/2021-ocr-correction`.

Figure 2 presents two versions of the same sample sentence excerpted from an early 19th-century book. The first line is a raw output from Tesseract OCR engine, the second is a manually corrected transcript of the same sentence. It is easy to notice some differences in both versions, which illustrate typical problems found in OCR correction. OCR models are usually trained with contemporary texts rather than historical ones. Thus even a highly precise systems tend to make errors in recognizing letters that are out of use in contemporary Polish or in words that used to be spelled differently than nowadays. This is the case of word *piérwszy* (recognized as *pićrwszy*) spelled with the letter *é* that is out of use since the turn of the 20th century. Another example is *nieprzyiemny* recognized as *nieprzyjemny*. The OCR recognized a contemporary spelling of the word even though originally it was spelled according to the early 19-century rules. Another typical error examples are words *nas* and *przywięznie* recognized as *mas* and *przywięzuie* due to a mismatch of a letter *n* with *m* and *u*. It could have been caused by an old typesetting of the original document.

```
A iako piękny ksztalt ciała uprzedza mas na pićrwszy widok, nieprzyjemny
odraża; tak też w dziele każdym styl iego naprzód nas przywięznie
lub odpycha.

A iako piękny kształt ciała uprzedza nas na piérwszy widok, nieprzyiemny
odraża; tak też w dziele każdym styl iego naprzód nas przywięzuie
lub odpycha.
```

Figure 2: A sample of text excerpted from the shared task dataset. The upper line presents output from OCR, the lower line is and manual transcription from original scan.

3. Evaluation

For the evaluation of submitted tasks we have chosen to use the word error rate metric (WER), most commonly used in similar contexts. Word error rate allows us to calculate the distance between the automatically corrected document and the gold-standard, prepared by human editors. The distance is expressed in the following formula, which takes into account the number of words that need to be changed, added or removed in the automatically corrected document to transform it into the gold-standard:

$$WER = \frac{S+D+I}{N} = \frac{S+D+I}{S+D+C} \tag{1}$$

where S is the number of substitutions, I is the number of insertions, D is the number of deletions, C is the number of correct words, N is the number of words in the reference text.

We thus expected the participants, given the input data, to produce just the corrected, running text of particular pages. Word error rate metric was calculated taking into account all pages in the test set.

4. Results

There were 28 submissions in total for the OCR correction task. Five teams decided to submit their solutions for the final test-B dataset, which was used to form the final ranking of the systems. The differences in performance of the systems were substantial and ranged from WER>8 to WER=3.744 for the final test-B dataset. As described in the next subsection, the approaches to the task included using heuristics, custom transformer architectures and a pretrained mT5 model, introduced by Xue et al. (2021). The final results have been presented in Table 1.

Table 1: Results of PolEval 2021 Task 3

Affiliation	System name	test-A WER	test-B WER
None	XXLv1	3.725	3.744
eNeLPol UJ AGH	ED 3 pl		4.302
Adam Mickiewicz University & Applica.ai	uml	5.338	5.328
None	1		7.208
Samurai Labs / Wrocław University of Science and Technology	Heuristics	8.063	8.217

5. Summary of submissions

5.1. Post-correction of OCR results using pre-trained language model (XXLv1)

The solution by Piotrowski (2021) relies on utilizing pretrained mT5 — a multilingual model based on encoder-decoder transformer architecture. The model was finetuned on the task of restoring original text using OCR output as an input. The training examples were mined from a provided dataset without additional augmentation. The final submission was based on the 13B parameter version of the mT5 model.

5.2. OCR correction with encoder-decoder transformer (ED 3 pl)

Wróbel (2021) aligned the input and output data and divided it into 50-word fragments. In the next step, they trained a T5 model and performed an evaluation on a 100-word fragment. ED3 submission used a mt5-base model, while ED3 pl used plt5-large.

5.3. Simple yet effective method of post-correcting OCR errors (uml)

Dyda (2021) first grouped the books with a radix chosen so that each bucket contained a similar number of elements and fined-tuned a Transformer based model to perform machine translation between incorrect OCR-ed text and the target one.

5.4. OCR post-correction with heuristics (Heuristics)

This solution to post-correction OCR results (Marcińczuk 2021) is based on a cascade of heuristics designed to fix specific groups of OCR errors. The authors identified four main types of OCR errors: punctuation, capitalization, similar alphabetical characters, and noise in the form of random characters. We created a set of heuristics for each group to correct the most common errors identified in the train set. Our approach reduced the word error rate (WER) on the test-A set from 18.726 to 8.063. On the final test set, we achieved 8.217 WER with the processing speed of 210 pages per second on a single CPU.

6. Conclusions and future work

In possible future editions of this task, we would like to diversify the OCR data by using several different OCR methods. Due to licensing issues this was unfortunately not possible during this edition, but we feel that limiting ourselves to a popular, but still single OCR algorithm might impact the versatility of submitted solutions. Another way of incorporating differentiation in the data would be to use multiple, random settings in Tesseract and this way simulate different OCR software conditions.

Another aspect of the data that we might have approached differently would be using a variety of sources, instead of scanned books. Such sources as scanned library index cards, typescript or partly damaged sources might have added more difficulty to the original task.

The ideas that were explored by other, similar tasks include separating error correction and error detection task and also suggesting more than one correction. The motivation for such approach is to use the trained model in computer-aided systems, in which the human operator makes the final correction, based on provided hints.

Although PolEval is an evaluation campaign focused mainly on machine learning approaches to natural language processing problems, it does not exclude heuristic-based solutions. For some problems traditional approaches seem more natural and OCR correction could be seen as one of such cases. In our task both heuristic-based and state-of-the-art machine learning-based solutions were submitted and compared, which we recognize as another small yet instructive value of our shared task. All heuristic-based solutions in the task were outperformed by those based on heuristics, which is to some extend expected but also reassuring and soothing conclusion that the field still delivers significantly better solutions to old problems.

References

Chiron G., Doucet A., Coustaty M. and Moreux J.-P. (2017). *ICDAR2017 Competition on Post-OCR Text Correction*. In *Proceedings of the 14th IAPR International Conference on Document Analysis and Recognition (ICDAR 2017)*, pp. 1423–1428.

Dyda P. (2021). *Simple Yet Effective Method of Post-correcting OCR Errors*. In Ogrodniczuk and Kobyliński (2021), pp. 103–115.

Marcińczuk M. (2021). *OCR Post-correction with Heuristics*. In Ogrodniczuk and Kobyliński (2021), pp. 117–121.

Mollá-Aliod D. and Cassidy S. (2017). *Overview of the 2017 ALTA Shared Task: Correcting OCR Errors*. In *Proceedings of the Australasian Language Technology Association Workshop 2017*, pp. 115–118, Brisbane, Australia.

Ogrodniczuk M. and Kobyliński Ł., editors (2021). *Proceedings of the PolEval 2021 Workshop*, Warsaw, Poland. Institute of Computer Science, Polish Academy of Sciences.

Piotrowski M. (2021). *Post-correction of OCR Results using Pre-trained Language Model*. In Ogrodniczuk and Kobyliński (2021), pp. 93–96.

Rigaud C., Doucet A., Coustaty M. and Moreux J.-P. (2019). *ICDAR 2019 Competition on Post-OCR Text Correction*. In *Proceedings of the 15th International Conference on Document Analysis and Recognition (ICDAR 2019)*, pp. 1588–1593.

Wróbel K. (2021). *OCR Correction with Encoder-Decoder Transformer*. In Ogrodniczuk and Kobyliński (2021), pp. 97–102.

Xue L., Constant N., Roberts A., Kale M., Al-Rfou R., Siddhant A., Barua A. and Raffel C. (2021). *mT5: A Massively Multilingual Pre-trained Text-to-Text Transformer*. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 483–498, Online. Association for Computational Linguistics.

Post-correction of OCR Results Using Pre-trained Language Model

Mateusz Piotrowski

Abstract

This paper presents our solution to the PolEval 2021 Post-correction of OCR results task. The solution is based on applying a pre-trained transformer model to restore the original text from OCR output. The final solution was created using a model with 13B parameters and achieved WER of 3.744. The code for reproducing the results is publicly available.[1]

Keywords

natural language processing, OCR post-correction, mT5, pretrained models, scaling laws

1. Introduction

The quality of digitized text corpora is often reduced by the errors induced by OCR models. The decoding process output may contain noise characters, transpositions, or miss fragments of the original text. Some of those incoherences are obvious given the context and language structure.

The automatic correction of these errors may be viewed as a "denoising" task in which we try to restore the original text from its distorted version. Similar tasks are now widely used as a pre-training objective of neural language models. For example, the "masked language modeling" objective from (Devlin et al. 2019) generates training examples by randomly masking or replacing tokens. The model is then trained to predict the masked token based on the context of the sentence.

These similarities make pre-trained language models a natural choice for OCR post-correction. In this work, we show the results of applying the mT5 model (Xue et al. 2021) to this problem. We also analyze the impact of increasing the model size on the quality of the corrected text.

[1] `https://github.com/mntss/poleval-2021`

2. Method

2.1. Model

The proposed solution uses the mT5 — a multilingual transformer model. The model was trained using "span corruption" objective from (Raffel et al. 2020), where whole spans from input texts are dropped and replaced with single special tokens. The neural network is tasked with generating the masked tokens for corresponding spans.

The pre-trained network checkpoints are available in configurations ranging from 300M to 13B parameters. Each model version was pre-trained on mC4 corpus totaling 6.3T tokens including 130B tokens in Polish subcorpus.

The mT5 model follows the encoder-decoder architecture introduced in (Vaswani et al. 2017). In our setup, the OCR output is first encoded into latent representation by the first module. This representation is then provided to the decoder, which generates the corrected text in an auto-regressive fashion. Such configuration allows the model to learn to insert and delete tokens at arbitrary positions using the context of the whole input fragment.

2.2. Data preparation

The training examples were obtained from the provided Wikisource dataset. Longer sequences were split into several examples due to the length limit imposed by the transformer architecture. To correctly divide longer texts, the input-output pairs were aligned using a sequence matching algorithm and a common split point was selected within sufficiently long matching regions.

The resulting examples were further filtered to discard instances where the similarity between input and output was low. This was done to prevent the model from learning to generate additional chunks of text. No augmentation was applied in order to preserve the original distribution of OCR failure modes.

2.3. Training procedure

To compare performance of across different sizes of the model we fine-tuned *base*, *large* and *XXL* versions of the mT5 model. While smaller models could benefit from using longer sequences length to provide better context, we limited the number of tokens to 384 for both input and output to allow for direct comparison. Instead, we increased the batch size to speed up the training. The performance details are presented in Table 1.

To models were fine-tune using TPUv3-8 device[2] — a ML accelerator providing 128GB of high-bandwidth memory which was necessary to enable training the largest model. The *base* and *large* configurations were fine-tuned for 200 000 steps and the *XXL* model was trained for 100 000 steps.

[2] `https://cloud.google.com/tpu`

Table 1: Model configuration details

Model	Parameters	Batch size	Examples/sec	Training time
base	580M	128	150	2d
large	1.2B	64	40	3d 16h
XXL	13B	16	7	2d 16h

3. Result summary

Table 2 shows evaluation results on datasets provided for the task. The *base* configuration was able to reduce the word-error rate of the OCR output by 71.2% across all datasets. The *large* model produced slightly better results achieving 72.8% error reduction.

Table 2: Word-error rates of the corrected text with use of different model sizes, *original* refers to the WER of the raw OCR output

Model	dev-0	test-A	test-B
original	16.550	16.527	16.543
base	4.678	4.792	4.796
large	4.418	4.515	4.559
XXL	3.604	3.725	3.744

Increasing the model size by the factor of 10 allowed to further reduce the error rate and achieve a 77.7% reduction on average. Overall the dependence of achieved performance on the model size in terms of a number of parameters seems to follow power-law relation (Kaplan et al. 2020).

4. Conclusions

This work presented results of applying the pre-trained transformer model to the task of correcting OCR outputs. Simply increasing the model size allowed us to achieve significant improvements in quality and produced the winning submission.

The effectiveness of this simple approach suggests that focusing on improving the capacity to train large neural networks might be a viable alternative to designing complex task-specific heuristics.

Acknowledgments

Research supported with Cloud TPUs from Google's TPU Research Cloud (TRC).[3]

[3] `https://sites.research.google/trc/about/`

References

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

Kaplan J., McCandlish S., Henighan T., Brown T. B., Chess B., Child R., Gray S., Radford A., Wu J. and Amodei D. (2020). *Scaling Laws for Neural Language Models*. arXiv:2001.08361.

Raffel C., Shazeer N., Roberts A., Lee K., Narang S., Matena M., Zhou Y., Li W. and Liu P. J. (2020). *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*. „Journal of Machine Learning Research", 21(140), p. 1–67.

Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser L. and Polosukhin I. (2017). *Attention is All You Need*. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS 2017)*, pp. 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

Xue L., Constant N., Roberts A., Kale M., Al-Rfou R., Siddhant A., Barua A. and Raffel C. (2021). *mT5: A Massively Multilingual Pre-trained Text-to-Text Transformer*. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 483–498, Online. Association for Computational Linguistics.

OCR Correction with Encoder-Decoder Transformer

Krzysztof Wróbel (Enelpol / Jagiellonian University / AGH University of Science and Technology)

Abstract

This paper presents contribution[1] to PolEval 2021 Task 3: Post-correction of OCR results. The system was scored second and achieved word error rate (WER) 4.3% (compared to 18.1% for the original OCR output).

Keywords

natural language processing, OCR correction, Polish

1. Introduction

Optical character recognition (OCR) is still relevant in the digital era, many texts have not been digitized yet and paper documents are still in use. However, OCR systems are not perfect and they need manual correction, which is time-consuming. In this paper, the word error rate is reduced over 4 times using a pre-trained language model without handcrafted rules or any analysis of data.

2. Data and preprocessing

The dataset prepared by competition organizers consists of scanned books published in 1791–1998. The scans are processed by Tesseract OCR and provided with reference manual transcriptions. The date of publication of the book is also provided.

Transformer models have text length limits, therefore a text needs to be split into chunks. To properly match input and output chunks, texts were aligned on token level using the

[1] `https://github.com/enelpol/poleval2021-task3`

Levenshtein distance algorithm and ties in edit operations were resolved with Levenshtein distance on character level. This operation does not change the WER score.

Figures 1 and 2 present example of token and character level alignments. If the text is split between the star and the dot, training chunks after character alignment are more consistent.

Input	kroki	nieprzyjacielskle,	*	.	rzekł
Output	kroki			nieprzyjacielskie,“	rzekł

Figure 1: Example of token level alignment algorithm result

Input	kroki	nieprzyjacielskle,	*	.	rzekł
Output	kroki	nieprzyjacielskie,“			rzekł

Figure 2: Example of character level alignment algorithm result

Table 1 shows basic statistics about the shared datasets. There is enough data to properly train models and reliably test them. Texts have similar average lengths.

Table 1: Number of texts and the average number of characters in texts for all datasets

Dataset	Number of texts	Average number of characters
train	46 097	1378.9
dev	5 508	1372.2
test-A	8 678	1368.6
test-B	8 435	1378.5

3. Method

The solution is based on a sequence to sequence model using T5 (Raffel et al. 2020) architecture. Publicly available models are mT5[2] (Xue et al. (2021) – multilingual) and plT5 (Polish)[3]. They are developed in different sizes. Table 2 shows number of parameters of these models. The size has usually an impact on the accuracy of the system and speed of prediction. The difference between mT5 and plT5 is due to the reduced size of a dictionary.

Training data was constructed by splitting input texts into N tokens chunks with aligned output texts as a reference output for the model.

[2] https://github.com/google-research/multilingual-t5
[3] https://huggingface.co/allegro/plt5-large
https://huggingface.co/allegro/plt5-base

Table 2: Number of parameters of models plT5 and mT5 in different sizes

Model	**Parameters [in billions]**
plT5-large	0.82
plT5-base	0.28
mT5-base	0.58
mT5-large	1.23
mT5-xxl	13.00

4. Experiments

Models were trained with Simple Transformers library[4] for 1 epoch using an effective batch size of 12. The linear scheduler was applied starting from a 0.001 learning rate. Texts were split into 50-word chunks. A problem is with a long time of output generation and therefore only 100 chunks were used as validation data.

The output generation process has two main parameters: the number of beams (number of hypotheses kept and choosing the best one) and whether sampling is enabled. Sampling was performed using *top-K* (Fan et al. (2018) – filtering top K tokens) of 50 and *top-p* (Holtzman et al. (2019) – filtering top tokens with cumulative probability exceeding p) of 0.95.

5. Results

Table 3 shows results of submissions using different models. Word error rate (WER) is a metric to rank the submissions. The winning model used the largest mT5 model. However, it may be less affordable to run as it needs more specialized hardware e.g. TPU. Interestingly, plT5-base has a worse score than the mT5-base, but it may be a coincidence related to conducting only one training process for each model.

Table 3: Results of the competition for datasets test-A and test-B of different submissions. The last row is a submission of the winner of the competition. Two submissions were sent to PolEval competition named “ED 3” and “ED 3 pl”.

	test-A	**test-B**
without correction	18.115	18.114
ED 3 (mT5-base)	4.986	5.001
ED 3 pl (plT5-large)	**4.281**	**4.302**
plT5-base	5.355	5.327
Mateusz Piotrowski (mT5-xxl)	3.725	3.744

The training time of plT5-large model was about 10 hours on GPU Tesla V100.

[4] `https://github.com/ThilinaRajapakse/simpletransformers`

The generation parameters have a significant influence on scores and time. Table 4 shows WER scores using whole or part of a dataset dev.

Table 4: WER scores of mT5-base model on whole or part (number of chunks are in parentheses) dev dataset using a different number of beams

Beams	Sampling (100)	Sampling (1000)	No sampling (1000)	No sampling	Sampling
1	5.593	5.711			
2	4.926	5.027	5.073		
4	**4.898**	5.066	4.960	4.992	
8	4.966	**4.911**	4.984	5.003	**4.930**

Using more than one beam makes a huge difference. Using a small sample of a dataset as validation does not correlate well with the score of the whole dataset. There is no conclusion if sampling is better than no sampling.

Tables 5 and 6 presents generation time on GPU using different parameters and models. Generation time on Tesla K80 might be 5 times larger than on Tesla V100.

Table 5: Time in seconds of output generation for the first 100 texts of the test-A dataset on GPU Tesla K80

	mT5-base	
Beams	No sampling	Sampling
1	25.98	36.61
2	32.78	50.04
4	53.05	72.03
8	89.23	127.86

Table 6: Time in seconds of output generation for the first 100 texts of the test-A dataset on GPU V100

	mT5-base		plT5-large		plT5-base	
Beams	No sampling	Sampling	No sampling	Sampling	No sampling	Sampling
1	5.24	10.07	89.88	95.30	16.65	17.94
2	7.59	13.77	139.14	154.91	23.03	27.97
4	10.52	19.32	234.54	263.50	29.16	34.93
8	17.37	34.32	393.75	430.66	41.82	60.19

Table 7 shows generation time on CPU – it is 20–100 times slower than on GPU.

Table 8 presents influence of batch size on generation time. However, a larger batch size requires more GPU VRAM, which is hardly available.

Table 7: Time in seconds of output generation for the first 100 texts of the test-A dataset on 1 core CPU Intel Xeon Gold 5220 CPU @ 2.20 GHz with mT5-base model without sampling

Batch size	**1 beam**	**2 beams**
1	718.13	847.42
2	541.80	837.29
4	535.24	722.83
8	488.56	580.12

Table 8: Time in seconds for plT5-large, with 2 beams and no sampling on GPU Tesla V100 with 32 GB VRAM

Batch size	**Time (s)**
4	437.17
8	283.36
16	192.73
32	139.14
48	132.75

6. Conclusions

The system shows that sequence to sequence models are appropriate for the OCR correction problem. It must be noted that shared training data is large enough to properly train the models. The more recent transformer models take as input scan images, so they are able to properly transform texts in columns. However, in this competition images were not shared.

Acknowledgments

This work was supported in part 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” and in part by the PLGrid Infrastructure.

References

Fan A., Lewis M. and Dauphin Y. (2018). *Hierarchical Neural Story Generation*. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 889–898, Melbourne, Australia. Association for Computational Linguistics.

Holtzman A., Buys J., Forbes M. and Choi Y. (2019). *The Curious Case of Neural Text Degeneration*. arXiv:1904.09751.

Raffel C., Shazeer N., Roberts A., Lee K., Narang S., Matena M., Zhou Y., Li W. and Liu P. J. (2020). *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*. „Journal of Machine Learning Research", 21(140), p. 1–67.

Xue L., Constant N., Roberts A., Kale M., Al-Rfou R., Siddhant A., Barua A. and Raffel C. (2021). *mT5: A Massively Multilingual Pre-trained Text-to-Text Transformer*. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 483–498, Online. Association for Computational Linguistics.

Simple Yet Effective Method of Post-correcting OCR Errors

Paweł Dyda (Adam Mickiewicz University / Applica.ai)

Abstract

This paper explores simple method of post-correcting OCR's engine output. While dominant solutions require careful alignment between OCR output and corrected text, I decided to rephrase problem as Machine Translation task and allow neural network to find useful representations on its own. This method has its drawbacks due to generative nature of Machine Translation models, however one could easily overcome these limitations by providing more training data. Ablation studies performed post PolEval 2021 competition seek ways for improving the method while keeping it simple.

Keywords

OCR post-correction, Transformers, Machine Translation

1. Introduction

Historical Document Processing is a valuable source of research problems. It is spanning several disciplines from humanities (history, sociology, linguistics, et cetera), digital imaging (e.g. multispectral imaging) to digitization (Optical Character Recognition, Information Retrieval, diachronic document processing including modernisation, OCR post-correction and more).

It is hard to overestimate the importance of preserving and studying cultural heritage; many people will benefit from progress in these areas.

In this article I decided to tackle the problem of post-correcting OCR errors. This relates to PolEval 2021 OCR post-correcting task, but the method is more generic.

The problem with Tesseract, the most popular (and free of charge) OCR framework is large number of errors for historical texts. There are many sources of such errors:

— low quality of imaging (quite often done in the past and impossible to improve due either to source fragility or amount of money needed);

— historical font faces and characters, that Tesseract was not trained to recognize;

— anachronistic words and outdated spelling or grammar rules (or lack of there of).

These issues are especially visible for low- and medium-resource languages, because of relatively low amount of data to train a language model.

At the same time, high number of historical documents such as books and newspapers makes it infeasible to perform manual correction of each and every scanned page. Instead, we have to find a way for correcting errors automatically.

One way to fix them would be to train better OCR models (Wick and Reul 2021), however that requires high number of manually transcribed documents.

Another avenue of research would be to fine-tune existing language model to tackle the problem of historical documents' OCR post-correction. This research paper presents a way to re-use pretrained language models to correct scanned Polish books published between 1751 and 1998.

2. Related works

Historically speaking, OCR errors post-corrections was semi-automated task. For instance, Bapst (1998) used specially crafted application to manually correct OCR errors and then obtain confusion matrix. That matrix was used to automatically correct similar mistakes.

As manual correction of errors is costly and time consuming, we are only interested in fully-automatic error correction methods. Therefore following text will only list approaches that do not require human intervention.

There are few possible ways to automatically correct OCR mistakes:

— use image pre-processing to improve image quality and OCR engine ability to recognize characters;

— use image features as well as OCR output to constrain the resulting words (obtaining corrected text);

— use multiple OCR systems (or multiple models within the same system) and majority votes;

— use post-OCR lexicon-based automated correction;

— use language model on OCR engine's output to correct errors.

Since improving image quality (resizing, sharpening, etc.) is still mostly manual task that requires human judgment, let's skip over it. Suffice to say, that this is interesting research

problem on its own (if it was already resolved, OCR engines would have pre-processing step integrated in their pipelines).

In late 20th century, popular method of correcting OCR errors was to model the output as a Noisy Channel and use statistical language modeling (and sometimes visual features) to obtain valid output. For example Hong (1996) used visual constraint model as well as statistical language model to improve accuracy of OCR. Similarly, Nagata (1998) used statistical confusion matrix (character shape similarities) and Markov chain-based language model to automatically correct OCR errors in Japanese texts.

Specifically, for processing historic documents Hauser (2007) used diachronic lexicon to correct OCR errors in scanned historical German texts. Reynaert (2008) used so-called anagram hashing and Levenshtein distance to lookup correct word in a lexicon for correcting 20th century historical Dutch newspapers' texts.

Recently, due to advancement in GPU computing it became possible to ensemble multiple Optical Character Recognition models. One example of such approach would be work by Wick and Reul (2021) where authors used Open Source OCR framework Calamari to train multiple OCR models and Exponential Moving Average as a voting mechanism to improve recognition of historical documents written in Latin.

Currently, there is a growing body of work where OCR output post-correcting is rephrased as Machine Translation, examples include (Afli et al. 2016, Amrhein and Clematide 2018, Nguyen et al. 2020, Mokhtar et al. 2018).

My research was inspired by the work of Junczys-Dowmunt et al. (2018), where authors used Neural Machine Translation approach for Grammatical Error Correction task.

3. Methodology

3.1. Source dataset

The PolEval 2021 OCR post-correction task consist of Polish-language books published between 1791 and 1998, scanned and recognized by Tesseract version 4.1. As manual transcriptions for these books were available thanks to Wikisource project, it was possible to create training and evaluation datasets.

Each dataset provides unique source document identifiers, page numbers of the source text as well as source text. For many examples, a year the book was published is available (sometimes it is approximate).

3.2. Data preprocessing

Since the grammar rules as well as words commonly used in Polish evolved over time, compelling solution was to group documents by year published.

I decided to select several buckets that should contain similar number of documents. The choice of buckets was arbitrary; it was guided by my experiences with historical Polish documents.

I knew the orthography has changed after Poland regained independence and after the World World II (but remained more or less constant before twentieth century). Table 1 presents counts of documents that were assigned to each bucket.

Table 1: Document counts for a specific bucket

Bucket	Years published	Data counts			
		train	validation	test-A	test-B
19th century	before 1900	7 738	908	1 477	1 406
1900	1900 to 1920	10 294	1 251	1 959	1 992
1920	1920 to 1930	11 310	1 358	2 184	2 066
1930	1930 to 1940	13 525	1 624	2 511	2 420
20th century	post 1940	3 199	366	544	543
other	various	31	1	3	8

The bucket was concatenated to the source text, so that model can learn to recognize which rules to apply.

Apart from grouping documents, the data preprocessing step included optional un-escaping (that is replacing escaped control sequences and backslashes with their literals). This way we received two versions of the transformed dataset:

— un-escaped (with direct literals later ignored in the tokenization process);

— escaped (I thought it might be important to preserve paragraphs and therefore decided on preserving new-line marks in their escaped form).

3.3. Problem rephrasing

It seemed obvious that simple solution would involve rephrasing OCR post-correction as a Machine Translation problem. That is exactly, what I did: I converted the task into translation task between Tesseract's output and correct Polish text.

I used popular Transformers library by Hugging Face (Wolf et al. 2020) to fine-tune pretrained language models on aforementioned task. I used two models (although with different number of parameters):

— mt5 (massively multilingual pre-trained text-to-text transformer; Xue et al. 2021);

— plt5 large[1] by Allegro.

I decided to use 768 tokens as a sequence length hoping that the text will fit. I have fine-tuned base and large version of these machine translation models for three epochs on both escaped and un-escaped datasets.

[1] `https://huggingface.co/allegro/plt5-large`

4. Experimental results

4.1. Submitted results

Table 2 contains results of the experiments as submitted to the evaluation portal.

Table 2: Results of originally performed experiments

Model	Size	Dataset	WER	
			test-A	test-B
mt5	base	un-escaped	6.305	6.360
mt5	large	escaped	6.504	6.681
mt5	large	un-escaped	**5.338**	**5.328**
plt5	large	un-escaped	5.370	5.378

It is possible to draw several conclusions:

— larger is better – models with larger number of parameters show better performance;

— escaped sequences does not help – it is harder to learn correct escaped sequences as they are not common in everyday language;

— multilingual model seem to perform better, although the margin is not high.

4.2. Error analysis

Knowing that the aforementioned evaluation portal is a special case of Gonito platform by Graliński et al. (2016) and thus gEval (Graliński et al. 2019) is used to evaluate submitted solutions, it was natural to check output to find the worst cases. After all, it is important to know where the solution shines and where it fails to generate valid results.

The sample output fragment from performed error analysis reveals an issue:

> *(...) W ciemnym przysionku czekałem czasami, Jak na przesmyku, ukryty za drzwiami, I pokojówkę, gdy wyszła przypadkiem, Wiodłem ze sobą, schwyciwszy ukradkiem....” Pieśń II. w. 159—62. i t. d. Czyżby nie była można iść do tego gdzieś? — rzekł i rzekł — a może nie może. — A może nie? — rzekł i rzekł — a może nie? — rzekł i rzekł — a może nie? — rzekł i rzekł — a może nie? — rzekł i rzekł — (...)*

Unfortunately, the model tends to oscillate repeating same phrase over an over again until it reaches maximum sequence length. This seems to be the reason behind most of the poor quality entries.

Worst features analysis reveals that proposed solution fails to translate new line characters and has hard times translating numbers.

Last but not least, common source of errors are Tesseract’s omissions: sometimes it fails to output even the whole entry (more commonly the part of the text). Since I decided to keep it

simple and not to use multi-modal model such as TILT by Powalski et al. (2021), there is no way to correct such errors.

5. Ablation studies

After seeing the results, natural question arises: can we do better? In order to find an answer, I have performed several studies. All studies if not stated otherwise was performed on mt5-large by fine-tuning it to post-correcting task for three epochs.

5.1. Bucket assignment

The simple document assignment to a bucket, using one's gut feeling does not seem the best scientific method. Therefore I decided to test several methods:

— no buckets at all – maybe the model can guess how to correct the output on its own?

— clustering by n-grams – first I have trained character n-gram model on all data, then ran k-means clustering algorithms to assign dataset entries to appropriate buckets;

— per-year bucket – instead of grouping years together let model learn correction function for each year;

— standard – as defined in Section 3.

Table 3 contains error rates for different bucket assignment functions.

Table 3: The effect of different bucket assignment method

Method	**WER**	
	test-A	**test-B**
standard	5.338	5.328
n-gram clusters	5.157	5.313
no buckets	5.315	5.288
bucket per year	5.152	5.452

The winning method seem to be the one base on character n-grams and k-means clustering. It provides the most balanced scores. Unfortunately, it is resource consuming and data preprocessing takes rather long time.

These results highlights potential issues with test-B dataset part: it seems that it does not exactly follows statistical distribution of the training set. For that reason, training without buckets seems to work better. In reality, model quite often fails to generate any answer at all. But the un-grouped nature of data decreases oscillations resulting in better corrections for other answers.

5.2. Number of epochs

One important aspect of Knowledge Transfer is how long to fine-tune the model. Given insufficient number of weight update steps, model tend to under-fit the data. Conversely, too many update steps will lead to Catastrophic Forgetting and thus model will fail to generalize well on unseen data. Following figures present WER and BLEU scores in function of number of epochs a model was fine-tuned.

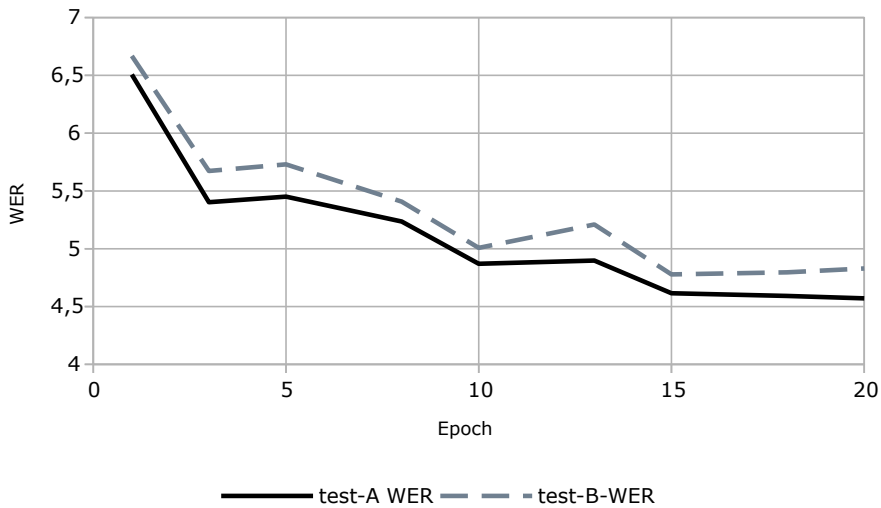

Figure 1: mt5 on un-escaped dataset Word Error Rate in function of epoch

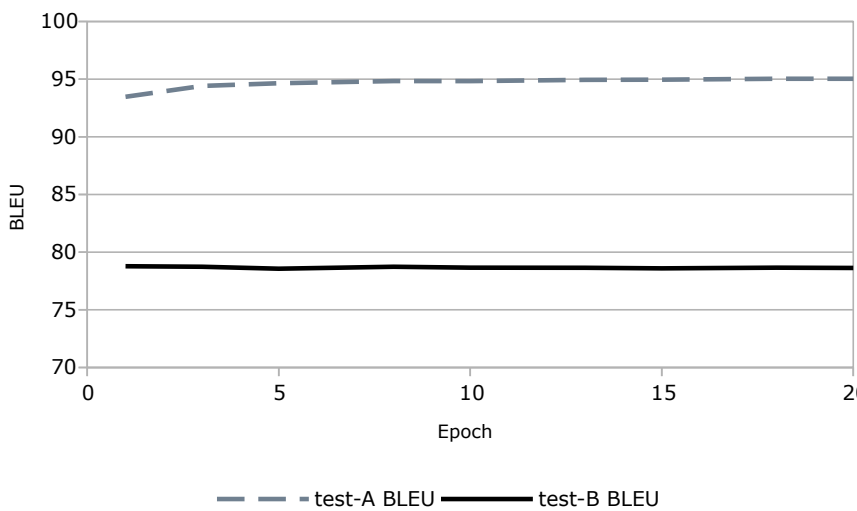

Figure 2: mt5 on un-escaped dataset BLEU in function of epoch

It seems that fine-tuning model for at least 15 epochs improves its performance. However, as it could be observed from figure 1, training model for more than 18 epochs decreases its generalization abilities on test-B dataset part. It is not easy to observe, but BLEU scores are decreasing for both test-A part, while increasing for test-B part.

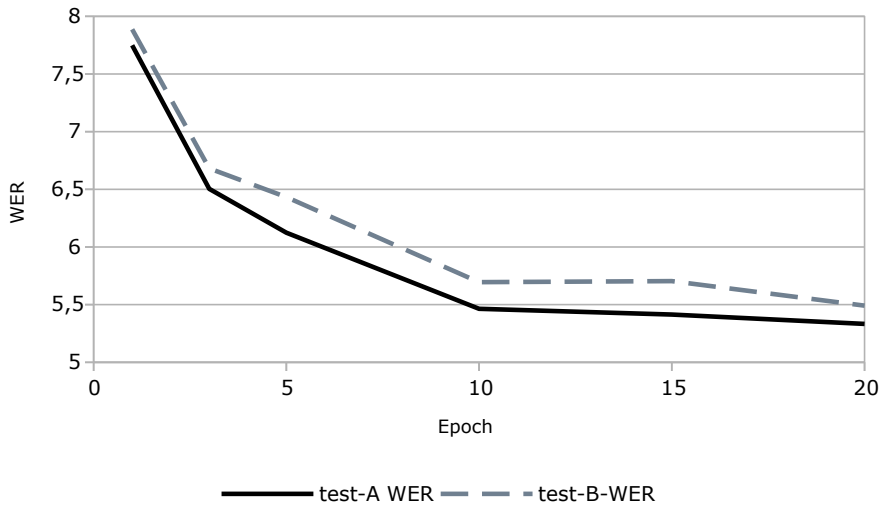

Figure 3: mt5 on escaped dataset Word Error Rate in function of epoch

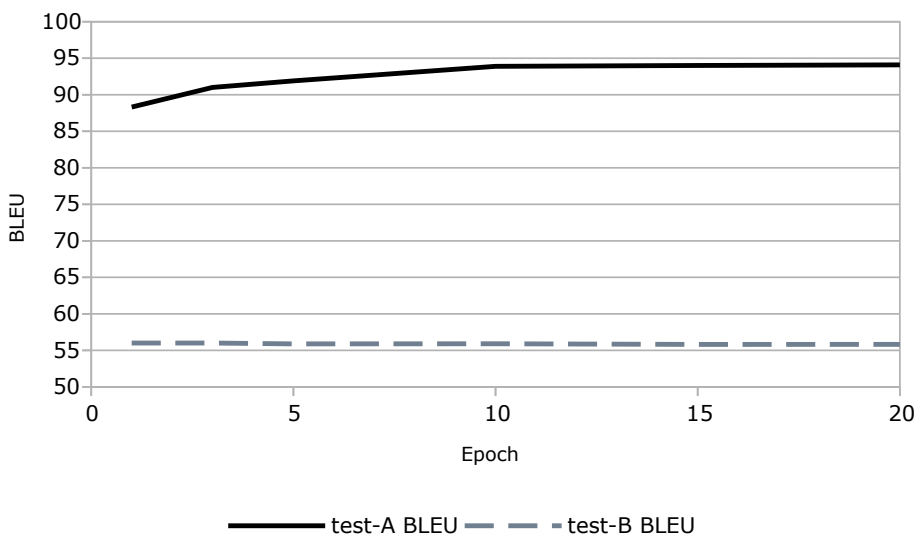

Figure 4: mt5 on escaped dataset BLEU in function of epoch

The performance on escaped dataset is improving constantly over the number of epochs, with an exception of test-B BLEU score, which seem to behave exactly the opposite to un-escaped

dataset case (decreasing after tenth epoch). This suggests that more epochs are needed to correctly mirror escape sequences and the model is under-trained.

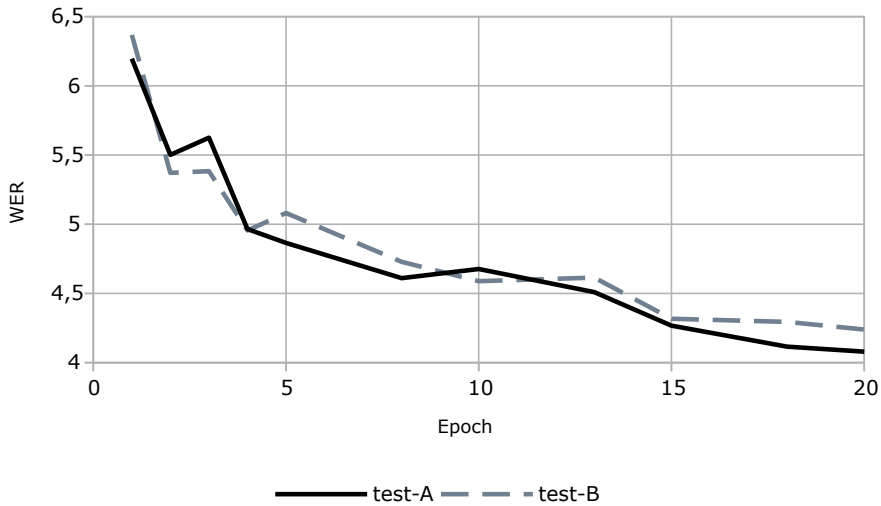

Figure 5: plt5 Word Error Rate in function of epoch.

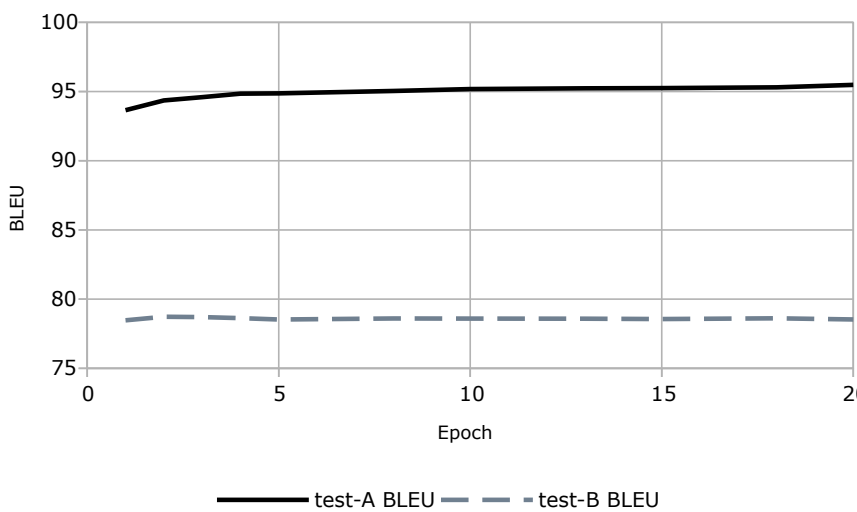

Figure 6: plt5 BLEU scores in function of epoch.

Similarly to mt5 fine-tuned on escaped dataset, plt5 fine-tuned on un-escaped dataset is steadily improving with number of epochs (again, test-B part BLEU score being an exception – it decreases slightly at the end). This model was also under-trained, having fine-tuned it for longer could possibly give me second place in the competition, even without other changes.

It is worth noting, that plt5 overtakes mt5 model after five epochs. The decision to fine-tune models only for three epochs had its consequences.

5.3. Sequence length

It is generally proven (e.g. Brown et al. (2020)) that larger transformer networks are more powerful, that is tend to generalize better and need fewer training examples. However, resource and budget constraints often disallow training them. For this task, I had to deal with sequence length – model size trade-off.
For instance, fine-tuning 3 billion mt5 model was infeasible on Nvidia A100 GPU card I used for that purpose. I would have to use absurdly short token sequences to avoid Out of Memory errors. This would require splitting training examples into small parts and careful alignment between recognized text and provided gold standard – in other words the resulting solution would be nothing but simple.

Instead, I decided to use fixed sequence length of 768 tokens. Longer sequence with mt5-large model produces Out of Memory errors – this is due to quadratic memory complexity of Self Attention Mechanism used by the Transformer model.

The most obvious question is: what if the whole text will not fit in such a short sequence? In order to find an answer, I decided to test mt5-base model with several sequence lengths to observe the effect of a longer sequence.

The following figure presents the effects of fine-tuning mt5-base model on un-escaped version of the dataset.

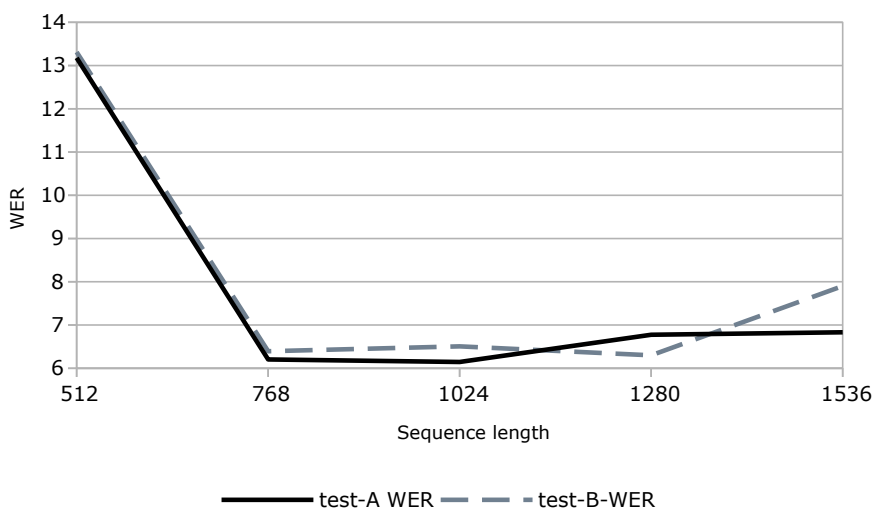

Figure 7: mt5-base Word Error Rate in function of sequence length

The data shows that sequence length of 768 tokens is a reasonable compromise – longer sequences often increase model oscillations (decreasing its performance). Again, differences between test-A and test-B parts highlight statistical distribution issues.

6. Conclusions

In this paper, I presented the simple method of correcting OCR errors and highlighted that given reasonable amount of data it could be a viable solution. As shown in Section 5, the choice of sequence length of 768 was reasonable. The better option would be to use dataset clustered by k-means algorithm on character n-grams and train plt5 model for more than 20 epochs.

7. Acknowledgments

I would like to thank my employer Applica.ai for providing required infrastructure and sponsoring this research.

References

Afli H., Barrault L. and Schwenk H. (2016). *OCR Error Correction Using Statistical Machine Translation*. „International Journal of Linguistics and Computational Applications", 7(1), p. 175–191.

Amrhein C. and Clematide S. (2018). *Supervised OCR Error Detection and Correction Using Statistical and Neural Machine Translation Methods*. „Journal for Language Technology and Computational Linguistics (JLCL)", 33(1), p. 49–76.

Bapst F. (1998). *Reconnaissance de documents assistée: architecture logicielle et intégration de savoir-faire*. Ph.D. thesis, Université de Fribourg.

Brown T., Mann B., Ryder N., Subbiah M., Kaplan J. D., Dhariwal P., Neelakantan A., Shyam P., Sastry G., Askell A., Agarwal S., Herbert-Voss A., Krueger G., Henighan T., Child R., Ramesh A., Ziegler D., Wu J., Winter C., Hesse C., Chen M., Sigler E., Litwin M., Gray S., Chess B., Clark J., Berner C., McCandlish S., Radford A., Sutskever I. and Amodei D. (2020). *Language Models are Few-Shot Learners*. In Larochelle H., Ranzato M., Hadsell R., Balcan M. F. and Lin H. (eds.), *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901. Curran Associates, Inc.

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

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

Hauser A. W. (2007). *OCR Postcorrection of Historical Texts*. Master's thesis, Ludwig-Maximilians-Universität München, Centrum für Informations- und Sprachverarbeitung (CIS).

Hong T. (1996). *Degraded Text Recognition Using Visual and Linguistic Context*. Ph.D. thesis, State University of New York at Buffalo, USA.

Junczys-Dowmunt M., Grundkiewicz R., Guha S. and Heafield K. (2018). *Approaching Neural Grammatical Error Correction as a Low-Resource Machine Translation Task*. 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. 595–606, New Orleans, Louisiana. Association for Computational Linguistics.

Mokhtar K., Bukhari S. S. and Dengel A. (2018). *OCR Error Correction: State-of-the-Art vs an NMT-based Approach*. In *Proceedings of the 13th IAPR International Workshop on Document Analysis Systems (DAS)*, pp. 429–434. IEEE.

Nagata M. (1998). *Japanese OCR Error Correction Using Character Shape Similarity and Statistical Language Model*. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 2*, pp. 922–928. Association for Computational Linguistics.

Nguyen T. T. H., Jatowt A., Nguyen N.-V., Coustaty M. and Doucet A. (2020). *Neural Machine Translation with BERT for Post-OCR Error Detection and Correction*. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries (JCDL 2020)*, p. 333–336, New York, NY, USA. Association for Computing Machinery.

Powalski R., Borchmann Ł., Jurkiewicz D., Dwojak T., Pietruszka M. and Pałka G. (2021). *Going Full-TILT Boogie on Document Understanding with Text-Image-Layout Transformer*. In *Document Analysis and Recognition – ICDAR 2021*, Lecture Notes in Computer Science, pp. 732–747, Cham. Springer International Publishing.

Reynaert M. (2008). *Non-interactive OCR Post-correction for Giga-Scale Digitization Projects*. In Gelbukh A. (ed.), *Computational Linguistics and Intelligent Text Processing*, pp. 617–630, Berlin, Heidelberg. Springer.

Wick C. and Reul C. (2021). *One-Model Ensemble-Learning for Text Recognition of Historical Printings*. In Lladós J., Lopresti D. and Uchida S. (eds.), *Document Analysis and Recognition – ICDAR 2021*, pp. 385–399, Cham. Springer International Publishing.

Wolf T., Debut L., Sanh V., Chaumond J., Delangue C., Moi A., Cistac P., Rault T., Louf R., Funtowicz M., Davison J., Shleifer S., von Platen P., Ma C., Jernite Y., Plu J., Xu C., Le Scao T., Gugger S., Drame M., Lhoest Q. and Rush A. (2020). *Transformers: State-of-the-Art Natural Language Processing*. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online. Association for Computational Linguistics.

Xue L., Constant N., Roberts A., Kale M., Al-Rfou R., Siddhant A., Barua A. and Raffel C. (2021). *mT5: A Massively Multilingual Pre-trained Text-to-Text Transformer*. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational*

Linguistics: Human Language Technologies, pp. 483–498, Online. Association for Computational Linguistics.

OCR Post-correction with Heuristics

Michał Marcińczuk
(Samurai Labs / Wrocław University of Science and Technology)

Abstract

Our solution to post-correction OCR results is based on a cascade of heuristics designed to fix specific groups of OCR errors. We identified four main types of OCR errors: punctuation, capitalisation, similar alphabetical characters, and noise in the form of random characters. We created a set of heuristics for each group to correct the most common errors identified in the train set. Our approach reduced the word error rate (WER) on the test-A set from 18.726 to 8.063. On the final test set, we achieved 8.217 WER with the processing speed of 210 pages per second on a single CPU.

The system is available under the BSD license at `https://github.com/mczuk/ocrpostfix`.

Keywords

natural language processing, OCR, post-correction, heuristics

1. Introduction

The goal of optical character recognition (OCR) is to convert printed text into machine-encoded text. The OCR is one of the earliest tasks from artificial intelligence, but it has not been solved yet. There is still a place for improvement, especially for scans from 18–19th century (Romanello et al. 2021). The quality of text recognition can be improved in several ways — by extending training dataset (Storchan and Beauschene 2019), graphical document pre-processing (Bieniecki et al. 2007) or OCR output post-processing (Ramirez-Orta et al. 2021). In the paper, we deal with the latter, i.e., post-correction of OCR results from Polish-language books published between 1791 and 1998. In the paper, we present a heuristic-based approach, which utilizes several lexical resources — publicly available dictionary for Polish and dictionaries developed from the training dataset.

2. OCR post-correction

2.1. Data analysis

We used a Python module for sequence comparison called *difflib*[1]. This module finds what operations are required to transform one text to the other. There are four possible operation types:

— replace — a part of A should be replaced by a part of B,

— delete — a part of A should be removed,

— insert — a part of B should be inserted,

— equal — both parts of A and B are equal.

Below is a list of operations for a sample input (i) and expected (e) sentences.

```
i: Znałem go dobrze-mówił Jnksa-\nprzyjaźniliśmy się, należeli
   do jednego gro-\nna;
e: Znałem go dobrze - mówił Jaksa - przyjaźniliśmy się, należeli
   do jednego grona;

equal    i[0:16]  --> e[0:16]   Znałem go dobrze --> Znałem go dobrze
insert   i[16:16] --> e[16:17]                ’’ --> ’ ’
equal    i[16:17] --> e[17:18]                 - --> -
insert   i[17:17] --> e[18:19]                ’’ --> ’ ’
equal    i[17:24] --> e[19:26]           mówił J --> mówił J
replace  i[24:25] --> e[26:27]                 n --> a
equal    i[25:28] --> e[27:30]               ksa --> ksa
insert   i[28:28] --> e[30:31]                ’’ --> ’ ’
equal    i[28:29] --> e[31:32]                 - --> -
replace  i[29:30] --> e[32:33]                \n --> ’ ’
equal    i[30:73] --> e[33:76]  przyjaźniliśmy się, należeli do jednego gro
                                                   -->
                                przyjaźniliśmy się, należeli do jednego gro
delete   i[73:75] --> e[76:76]               -\n --> ’’
equal    i[75:78] --> e[76:79]               na; --> na;
```

We processed the training dataset set with the *difflib* module and collected all the replace operations. We used the statistics to identify the most common differences between the raw OCR output and the texts' expected form. We analyzed the differences at the level of characters, words, and words bi-grams. Information about *equal* phrases was utilized to identify how often the identified changes do not occur. Then, based on comparing occurrences, we developed a set of heuristics and lexical resources that handle the most common mistakes.

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

2.2. Cascade of heuristics

Hyphenation

Some words at the end of the line are broken between lines — they are divided into two parts, and the first part ends with a hyphen (-). In some cases, OCR recognized hyphens as other punctuation marks — equal sign (=) or guillemets («, »). The set of four marks was used to join the broken words into a single word.

Punctuation

The most common errors related to punctuation were:

— opening round bracket was recognized as upper letter C,

— exclamation mark (!) was recognized as pipe (|),

— double quotations mark („) was recognized as two commas („),

— full stop (.) was recognized as comma (,) and vice-versa.

Character swaps

For example, *n* might be recognized instead of *r* or *m*, *m* can be recognized instead of *ni*, and so on. For each word not found in the dictionary, we generate a list of possible forms. The forms are generated by replacing characters with their alternatives (see Table 1).

Table 1: Characters swaps

Recognized	Possible target character	Recognized	Possible target character
a	ą, u	m	n
ą	a	o	ó
B	s	ó	é
c	e	r	s, n
ć	é	s	ś
e	c, ę	ś	s
ę	e	u	n
F	P	ż	z, ź
G	C	ź	ż
h	b	v	u
i	t, l	f	p
J	I	eb	ch
l	ł, t	ni	m
ł	l	Si	sł
n	u, m		

If the list of possible forms contains only one form and is present in one of the dictionaries, we replace the word with the form. To check words, we used two dictionaries: (a) a list of words from the training dataset and (b) SGJP dictionary (Woliński et al. 2020).

Word and phrase swaps

In many cases, the simple character swap is insufficient due to ambiguity. For example, one of the most common mistakes is recognition of *m* as *n* and vice-versa. For example, in the case of word *ma* (Eng. have) and *na* (Eng. on) both word forms are present in the dictionary. However, the phrase *ma brzegach* does not occur in the dataset, while *na brzegach* appears several times. The list of word and phrase swaps consists of more than 3k items.

Trashes

OCR systems tend to generate semi-random sequences of characters when they encounter non-text elements on the image — stains, marks, and so on. This was also an issue in this case. The "trashes" took various forms – a sequence of short words, random non-alphabetical characters, for example:

— `NYR UR UR UR UR UR UR UR UB UR UB UR UN WA YN UN UN UN,`

— `«+ «+ «0556655 0% rsa .`

To handle some of the cases, we implemented several heuristics. The heuristics were looking for sequences of short words with a higher density of non-alphabetic characters.

3. Evaluation

To evaluate the performance, we used the word error rate metric (WER). Table 2 contains the performance of our approach on the test-A and test-B datasets. For the test-A dataset, we provide the WER metric after each step of processing. With our approach, we were able to lower the WER on test-A from 18.726 to 8.062. On test-B we obtained 8.217 WER. The test-B dataset (8435 documents) was processed in 40 seconds on a single CPU thread — that is, ca. 210 documents per second.

4. Summary

In the paper, we presented a heuristic-based approach to the OCR post-correction for Polish. The approach utilized lexical resources — a Polish dictionary and a set of resources developed on the training dataset. Our method processed the test-B dataset in 40 seconds on a single CPU thread and obtained 8.217 WER.

Table 2: WER on the test-A and test-B datasets

	test-A	**test-B**
OCR output	18.726	–
hyphenation	9.750	–
punctuation	9.122	–
word swaps	8.945	–
char swaps	8.368	–
trashes	8.237	–
phrase swaps	8.062	–
Final	8.062	8.217

References

Bieniecki W., Grabowski S. and Rozenberg W. (2007). *Image Preprocessing for Improving OCR Accuracy*. In *Proceedings of the 2007 International Conference on Perspective Technologies and Methods in MEMS Design (MEMSTECH 2007)*, pp. 75–80.

Ramirez-Orta J., Xamena E., Maguitman A., Milios E. and Soto A. J. (2021). *Post-OCR Document Correction with Large Ensembles of Character Sequence Models*. arXiv:2109.06264.

Romanello M., Najem-Meyer S. and Robertson B. (2021). *Optical Character Recognition of 19th Century Classical Commentaries: the Current State of Affairs*. arXiv:2110.06817.

Storchan V. and Beauschene J. (2019). *Data Augmentation via Adversarial Networks for Optical Character Recognition*. In *Proceedings of the 2019 International Conference on Document Analysis and Recognition (ICDAR)*, pp. 184–189.

Woliński M., Saloni Z., Wołosz R., Gruszczyński W., Skowrońska D. and Bronk Z. (2020). *Słownik gramatyczny języka polskiego. wydanie IV*.

PolEval 2021 Task 4: Question Answering Challenge

Maciej Ogrodniczuk, Piotr Przybyła
(Institute of Computer Science, Polish Academy of Sciences)

Abstract

We introduce the Question Answering Challenge – a shared task organised at the PolEval 2021. The task involves answering open-domain free-form questions in Polish through an automatic system, without human intervention or accessing external services. We describe the motivation behind the problem, explore various question types and formulations, and lay out the rules of the competition. The solutions submitted by 7 teams are discussed in terms of their evaluation results and the techniques applied.

Keywords

question answering, PolEval, shared task

1. Introduction

The idea of an open domain question answering (QA) system has always seemed more attractive than keyword-based search since it promises a compelling perspective of a seemingly intelligent agent answering users' questions instantaneously rather than requiring them to browse through documents to find the answer to their question. But the QA setup is also natural for knowledge exchange between humans: for years it was used to test general knowledge in quiz shows such as *Fifteen to One* (PL: *Jeden z dziesięciu*) or *Jeopardy!* (PL: *Va banque*). In 2011, this dream came closer to realisation when IBM Watson (Ferrucci et al. 2010) successfully competed with human contestants in a series of *Jeopardy!* games.

The purpose of the work described here is to encourage a similar development for Polish. The Question Answering Challenge (QAC) involves developing a solution capable of providing answers to general-knowledge questions, typical for popular TV quiz shows. With numerous new language models and deep neural network-powered improvements in almost every natural language processing task, the QAC results can show where the state-of-the-art stands

and further motivate development and testing of new solutions for languages other than English.

2. Related work

A task, in which a computer system is expected to formulate an answer in a natural language (e.g. English) that addresses a question in a natural language (typically the same one) is described as *Question Answering* (QA). Given the flexibility of natural languages, a great variety of problems might be formulated as QA tasks, but what they all have in common is the need to apply NLP methods to understand the question and generate an answer.

2.1. QA systems

BASEBALL (Green et al. 1961), the first QA system developed, was only capable of answering questions regarding a strictly defined domain, namely results of baseball matches in a US league during one season. A set of rules was used to convert a question to a query for a structured database, from which an answer was retrieved. Similar approaches were popular in the following years (Simmons 1970), but the manual effort necessary to prepare rules was limiting their abilities. The availability of vast open-domain knowledge on the Internet triggered the second wave of QA systems; e.g. *START* (Katz 1997) converted web text to a set of triples, in which the one matching a query was sought.

The standard architecture for modern open-domain QA includes three stages: question analysis, document retrieval and answer extraction. As in other areas of NLP, current QA approaches employ neural networks trained on large datasets for each of these steps (Zhu et al. 2021). More recently, advances in natural language generation based on the Transformer architecture (Vaswani et al. 2017) have allowed to encode significant amount of knowledge in a pre-trained language model. As a result, solutions like GPT-3 (Brown et al. 2020) can achieve competitive QA performance given a prompt with just a few examples.

2.2. QA data

Given the importance of ML approaches in the QA solutions, it is no surprise that availability of training data has played a critical role in the research so far. The first datasets were made openly available through the QA shared tasks organised within the Text Retrieval Conference (TREC) from 1999 (Voorhees 1999) to 2007 (Dang et al. 2007).

Since then, many more collections, built from vastly different perspectives, have been published and used for evaluation efforts – see review by Zhu et al. (2021). Some aspects in which these datasets differ are (1) whether the knowledge base to extract answers from is provided; (2) whether questions belong to a particular topic or are open-domain and (3) whether an expected answer is a span from the knowledge base, a selected option (true/false or multiple choice), or a free-form text. Datasets that, like in our task, contain free-form answers from any

domain, include *MARCO* (Nguyen et al. 2016), *Quasar-T* (Dhingra et al. 2017) and *DuReader* (He et al. 2018).

In case of many datasets, including the three mentioned above, each question is also accompanied by a few text passages (usually results from a search engine) that should contain the answer. The task of answering a question based on such text is known as Machine Reading Comprehension (MRC). Many QA formulations differ from ours in that they rely on the provided snippets even further, requiring an answer to be a span within one of these passages, rather than a free-form string. A well-known example for this is Stanford Question Answering Dataset (SQuAD), which includes 100,000 questions prepared by crowdworkers based on articles from Wikipedia (Rajpurkar et al. 2016).

2.3. QA for Polish

Similarly to other NLP tasks, the resources and solutions for English are by far the most numerous, but not the only ones available. The first QA system for Polish was developed by Vetulani (1988) as a natural-language interface to *ORBIS* database, implemented in *PROLOG*. As with English, all initial attempts were limited to a fixed domain, such as business information (Duclaye et al. 2002) or public safety (Vetulani et al. 2010). To answer open-domain questions, some systems relied on the structure of Wikipedia; for example RAFAEL (Przybyła 2016) uses it both as a knowledge base and to build a resource for precise entity recognition (Przybyła 2015). Other approaches used a corpus of web pages gathered based on the questions in the evaluation set (Walas and Jassem 2010, Marcińczuk et al. 2013).

Regarding datasets, two of the approaches mentioned above were accompanied by substantial question collections in Polish. The first system used 4721 questions gathered from the *Did you know?* panel on Wikipedia page (Marcińczuk et al. 2013). An important limitation of the collection is the lack of explicit answers – instead, each question is associated with a Wikipedia page with a selection of relevant fragments. The second system, RAFAEL (Przybyła 2016), used a collection of 1130 questions from a Polish quiz TV show *Jeden z dziesięciu* (Karzewski 1997). These questions were included with explicit answers and could thus be used for the present work (see Section 4).

3. QAC task

The problem in QAC is simple: given a text string, expressing an open-domain question in Polish, return a text string consisting a correct answer in Polish. According to the task's rules, the participants are free to choose any approach, as long as it answers the question automatically, without human assistance. The participants can build on top of existing resources, but they need to be downloaded and included locally, so the questions can be answered without access to the Internet. Thus, for example, pre-downloading Wikipedia and indexing it for search is acceptable (and applied by most participants), but querying Google online is not.

In order to facilitate the development of high-performing solutions by the participants, the following resources were made available:

— collections of questions with answers (Section 4), representing various types (Section 4.2) and split into development and test subsets (Section 4.4),

— an automatic evaluation method, allowing for some deviation between the user-supplied and gold-standard answers (Section 5),

— a simple baseline solution to compare against (Section 6).

4. Question collection

For the purposes of the task, a collection of 6000 questions and answers was created. Here we describe the data sources, various types of questions and answers observed and how the dataset was used in QAC.

4.1. Data sources

The data for the task was collected from various sources. The majority of question-answer pairs was collected from websites, where fans of the quiz shows collect or exchange questions[1], file sharing services[2], various online newspaper quizzes[3] and eventually from video-on-demand collections of actual quiz shows broadcast on the Polish television[4]. The 1130 questions gathered for the RAFAEL system (see Section 2.3) were also included.

The data was manually cleaned in numerous stages, including improving the phrasing, removing of duplicates, verification of answers in Wikipedia, adding variant answers or different formulations of the same answer (see Section 4.3).

The following types of questions were removed from the set:

— about issues dependent on time of asking the question or formulating the answer (e.g. the current president of Poland)

— seeking more than two entities in an answer (e.g. three out of five neighbouring countries)

— requesting to sort items (e.g. from the highest to lowest mountain)

— related to the spelling rules (e.g. *ch/h*, *rz/ż*, *ó/u* etc.)

— those requiring longer explanations (e.g. starting with *why*).

[1] Such as `http://zenzycia.blogspot.com/2018/08/pytania-z-teleturnieju-1-z-10-spis.html`, similar to `https://j-archive.com/` where fans of *The Jeopardy!* share questions and answers from episodes aired between 1983 and now.

[2] Such as `chomikuj.pl`.

[3] E.g. `onet.pl`, `kurierlubelski.pl`, `radiozet.pl`, `gazetawroclawska.pl`, `se.pl`, `polskatimes.pl`.

[4] E.g. *To był rok*, *Va Banque* i *Jeden z dziesięciu* na platformie `vod.tvp.pl`.

The dataset was then truncated to 6000 records and split into development and two test subsets.

4.2. Question types

Grouping questions into categories is helpful to understand the challenges posed by the task, but there are many possible ways to do this. Our categorisation is based on previous work for Polish (Przybyła 2016), but includes additional types we observed in our, significantly larger, dataset. Namely, we group question according to what information they seek:

— **Single entity**: a name of a specific entity:

  Q: *Jak nazywa się bohaterka gier komputerowych z serii Tomb Raider?* [Who is the hero in the Tomb Rider video game series?]

  A: *Lara Croft*

— **Multiple entities**: several entities, specified by names[5]:

  Q: *Które dwa morza łączy Kanał Koryncki?* [Which two seas are linked by the Corinth Canal?]

  A: *Egejskie i Jońskie* [Aegean and Ionian]

— **Entity choice**: one of the options, which are given in the question:

  Q: *Co zabiera Wenus więcej czasu: obieg dookoła Słońca czy obrót dookoła osi?* [What takes more time in case of Venus: revolving around the sun or rotating about its own axis?]

  A: *Obrót dookoła osi* [Rotating about its axis]

— **True/false**: veracity value:

  Q: *Czy w przypadku skrócenia kadencji Sejmu ulega skróceniu kadencja Senatu?* [When the term of office of the Polish Sejm is terminated, does it apply to the Senate as well?]

  A: *Tak* [Yes]

— **Other name**: alternative names for a given entity:

  Q: *Jaki przydomek nosił Ludwik I, król Franków i syn Karola Wielkiego?* [What was the nickname of Louis I, the King of the Franks and the son of Charlemagne?]

  A: *Pobożny* [The Pious]

— **Gap filling**: words that complete a given sentence, quote or expression:

  Q: *Proszę dokończyć powiedzenie: „piłka jest okrągła, a bramki są...”* [Finish the adage: „the ball is round and the goals are...”]

  A: *Dwie* [Two]

Moreover, the questions, where an answer involves one or more entities (Single entity, Multiple entities, Entity choice) can be also divided with respect to the type of the entities sought:

[5] In our dataset we only include questions that can be answered by providing two entities.

— **Named entities**: specific entities that are referred to through their names, e.g. people, countries, organisations. This also includes quantities, numbers and dates.

— **Unnamed entities**: more general categories of entities, such as objects, concepts or species:

  Q: *Paź królowej to gatunek których owadów?* [Old World swallowtail is a species of what type of insects?]

  A: *Motyli* [Butterflies]

Finally, the questions can be phrased through one of the following formulations:

— **plain question**,

— **command**: *Proszę rozwinąć skrót CIA.* [Expand the abbreviation 'CIA'.]

— **compound** consisting of more than one sentence: *Ten urodzony w XIX w. Nantes francuski pisarz uchodzi za prekursora literatury fantastycznonaukowej. O kogo chodzi?* [This French writer, born in the 19th century, is considered a pioneer of the sci-fi literature. Who is he?]

4.3. Answers

Answer strings are formulated in the same way, in which a Polish speaker would normally reply to a given query, e.g. when participating in a quiz. Namely, they contain just a few words that satisfy the question, for example a name of a person or place. This is different to many other QA shared tasks, where a sentence, a paragraph, or even a whole document from a knowledge base are considered an acceptable result.

In QAC the answers can, when appropriate:

— contain prepositions:

  Q: *W którym mieście trasa drogi krzyżowej przebiega ulicą Via Dolorosa?* [In which city a traditional Good Friday procession is held on the Via Dolorosa street?]

  A: *W Jerozolimie* [In Jerusalem]

— be inflected:

  Q: *Symbolem którego pierwiastka jest Cr*? ['Cr' is the symbol of which element?]

  A: *Chromu* [Of chrome[6]]

— contain punctuation:

  Q: *W jakim filmie mężczyzna w białym garniturze zrywa lilie ze stawu?* [In which film a man in white suit is seen in a pond, picking lilies?]

  A: *„Noce i dnie"*

[6] In Polish this meaning is achieved by inflecting the word instead of adding a preposition.

Table 1: Summary of the question collection

			Development	Test A	Test B	Total
Number of questions			1000	2500	2500	6000
Average question size in tokens			8.43	8.41	9.49	8.87
Median question size in tokens			8	8	9	8
Min question size in tokens			3	3	3	3
Max question size in tokens			21	22	32	32
Number of questions with	1	**answer variants**	890	2178	1890	4958
	2		104	307	570	981
	3		5	13	29	47
	4		0	1	11	12
	5		1	1	0	2

— for person names, include the first name and surname:

Q: *Który brytyjski pisarz wprowadził do literatury pojęcie Wielkiego Brata?* [Which British writer introduced the concept of 'Big Brother'?]

A: *George Orwell*

In some cases, more than one answer text is associated with a question, e.g. *Richard I*, *Richard Cœur de Lion* and *Richard the Lionheart*. Their meaning is identical and this redundancy is necessary to make sure the automatic evaluation procedure accepts different synonyms of the right answer. Also, when two entities are requested, both *A and B* and *B and A* are recorded as the possible answers.

4.4. Data format and characteristics

The whole QAC question collection (see summary in Table 1) was split into the following subsets provided to the participants:

— a small development set, including question and answers,

— test set A, including question and answers,

— test set B, without answers.

Test set B and the answers to test set A were provided in the last week of the competition, which allowed the participants to include test set A in their training data for the final submission. The collections are published in the GitHub repository of the task[7].

The development dataset includes two UTF-8-encoded text files: `in.tsv` with questions and `expected.tsv` with tab-separated answers. The submission file is supposed to contain just answers in separate lines.

Questions contain 8–9 tokens on average and are mostly simple, asking for a single entity. The longest answer has 7 tokens and is associated with the question about two labours of

[7] `https://github.com/poleval/2021-question-answering`

Heracles related to horses (stealing the Mares of Diomedes and cleaning the Augean stables). The data contain various numbers of accepted answers – usually corresponding to different formulations (e.g. full name of a person or just their surname), but sometimes referring to entirely different names for a certain entity, e.g. *trembita*, *trombita*, *bazuna* or *ligawka* for the name of an alpine horn made of wood.

5. Evaluation method

Submissions were evaluated by comparing the known answer (gold standard) to the one provided by the participating systems (predictions). The number of matching answers divided by the number of questions in the test set was considered the **accuracy** of a given approach.

Checking if the two answers match depends on the question type:

— For non-numerical questions, we assessed textual similarity. To that end, a character-wise Levenshtein distance was computed between the two (lowercased) strings and if the obtained value was less than ½ of the length of the gold standard answer, we accepted the candidate answer.

— For numerical questions (e.g. *In which year...*), we assessed numerical similarity. Specifically, we used a regular expression to extract a sequence of characters that could be interpreted as a number (Arabic or Roman numeral). If such sequences could be found in both answers and represented the same number, we accepted the prediction.

For questions, where more than one answer text was available, the answer that had the best match with the candidate was used.

6. Baseline

The `WIKI_SEARCH` baseline solution used the question as a query to the Wikipedia search service and treated the title of the first returned article as an answer, as long as it didn't overlap with the question.

Specifically, the following procedure was followed:

1. Split the question into tokens using spaCy (model `pl_core_news_sm`) and ignore the one-character tokens,
2. Send the space-separated tokens as a query to the Search API of the Polish Wikipedia[8],
3. For each of the returned articles:
   (a) Split its title into tokens with spaCy,
   (b) If none of the tokens of the title has at least 50% overlap (measured as in Section 5) with any of the tokens of the question:

[8] `https://www.mediawiki.org/wiki/API:Search`

i. remove the part of the title starting from '(', if found

ii. return the title as an answer,

(c) Otherwise, continue to the next result,

4. If no answer is found at this point, remove the first of the question tokens and jump back to (2).

7. Submissions and results

The competition attracted 50 submissions in total. In its initial phase (with submissions evaluated on test-A dataset) 22 solutions (including the official baseline from Section 6) were submitted by only 2 external participants. In the final phase 28 submissions were made by 7 participants.

Table 2 reports on the accuracies of individual highest-scoring submissions for each participant. The winning system by Mateusz Piotrowski scored 71.68%, surpassing other competitors by more than 20 points. Two independent second-best submissions by Aleksander Smywiński-Pohl and Piotr Rybak scored 50.96%. `WIKI_SEARCH` entry corresponds to the official baseline and the two other lowest-scoring submissions were marked as baselines by their authors.

Table 2: Accuracy scores at PolEval 2021 Task 4

Submission	**test-B accuracy**
Mateusz Piotrowski	71.68%
Aleksander Smywiński-Pohl et al.	50.96%
Piotr Rybak	50.96%
Darek Kłeczek	46.44%
Karol Gawron	36.12%
`WIKI_SEARCH`	9.60%
Filip Graliński	4.16%
BI Insight	0.96%

The authors of the four top-scoring submissions agreed to describe their systems and present them at the PolEval workshop. You can find short summaries of each solution in Sections 7.1–7.4 and relevant papers later in this volume.

7.1. Search-augmented question answering system using multilingual transformer model

The solution by Piotrowski (2021) used a combination of Wikipedia knowledge retrieval and multilingual transformer model. First, questions were used as search queries to retrieve a set of relevant paragraphs from the Polish Wikipedia. Then the results ranked with a BM25

scoring function, along with a question, were fed to a neural network, which is responsible for generating the final answer. The mT5 model (Xue et al. 2021) was used to generate answers.

7.2. Dense Passage Retriever

The solution by Smywiński-Pohl et al. (2021) used a combination of approaches dependent on question type. For extractive question answering the authors used Dense Passage Retrieval (Karpukhin et al. 2020) trained on "Czy wiesz" dataset (Marcińczuk et al. 2013). Boolean questions were answered according the Natural Language Inference model, trained on CDSCorpus (Wróblewska and Krasnowska-Kieraś 2017, Krasnowska-Kieraś and Wróblewska 2019) and a dump of Wikipedia (Smywiński-Pohl 2019), utilizing HerBERT large (Mroczkowski et al. 2021).

7.3. Retrieve and refine system for Polish question answering

The solution by Rybak (2021) consisted of two modules. The first one searched for Wikipedia paragraphs containing the answer to a question. For training such a model, 10 000 question-paragraph pairs were manually annotated and merged with the "Czy wiesz" collection (Marcińczuk et al. 2013, Rybak et al. 2020). The encoder was trained with the HerBERT Large model (Mroczkowski et al. 2021). Then it was used to find the 10 most relevant paragraphs for each question. In the second step a generative model was trained on plT5 Base[9] which takes the question and the top 10 paragraphs as input and generates the answer as output. The second model was trained on the test-A validation set, a thousand additional questions from "Jeden z dziesięciu" quiz and a subset of Multi-lingual Knowledge Questions & Answers (Longpre et al. 2020).

7.4. Retriever-reader approach with data-driven improvements

The solution by Kłeczek (2021) was based on retriever-reader architecture using Deepset Haystack framework (Deepset 2021) with ElasticSearch document store. The dataset was further supplemented with Polish Wikipedia dump, word definitions from Słowosieć (Maziarz et al. 2016) and the Polish subset of mC4 dataset (Xue et al. 2021) filtered for URLs of popular educational websites. BM25 was used as a retriever and reader was XLM-Roberta Large pre-trained on original English SQUAD (Chan et al. 2021). Finally the results were postprocessed by converting numbers to numeric form and answering all yes/no questions with *yes*.

Acknowledgements

This work was supported by the *Polish National Agency for Academic Exchange* through a *Polish Returns* grant number PPN/PPO/2018/1/00006.

[9] `https://hf.co/allegro/plt5-base`

We would like to thank Aleksander Smywiński-Pohl for spotting a few errors in our dataset.

References

Brown T. B., Mann B., Ryder N., Subbiah M., Kaplan J., Dhariwal P., Neelakantan A., Shyam P., Sastry G., Askell A., Agarwal S., Herbert-Voss A., Krueger G., Henighan T., Child R., Ramesh A., Ziegler D. M., Wu J., Winter C., Hesse C., Chen M., Sigler E., Litwin M., Gray S., Chess B., Clark J., Berner C., Mccandlish S., Radford A., Sutskever I. and Amodei D. (2020). *Language Models are Few-Shot Learners*. In *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901.

Chan B., Möller T., Pietsch M. and Soni T. (2021). *Multilingual XLM-RoBERTa large for QA on Various Languages*. `https://huggingface.co/deepset/xlm-roberta-large-squad2`.

Dang H. T., Kelly D. and Lin J. (2007). *Overview of the TREC 2007 Question Answering Track*. In *Proceedings of The Sixteenth Text REtrieval Conference (TREC 2007)*. NIST.

Deepset (2021). *Haystack*. `https://github.com/deepset-ai/haystack`. GitHub repository.

Dhingra B., Mazaitis K. and Cohen W. W. (2017). *Quasar: Datasets for Question Answering by Search and Reading*. arXiv:1707.03904.

Duclaye F., Sitko J., Filoche P. and Collin O. (2002). *A Polish Question-Answering System for Business Information*. In *Proceedings of the 5th International Conference on Business Information Systems (BIS 2002)*, pp. 209–212. Department of Information Systems, Poznań University of Economics.

Ferrucci D. A., Brown E. W., Chu-Carroll J., Fan J., Gondek D., Kalyanpur A., Lally A., Murdock J. W., Nyberg E., Prager J. M., Schlaefer N. and Welty C. A. (2010). *Building Watson: An Overview of the DeepQA Project*. „AI Magazine", 31(3), p. 59–79.

Green B. F., Wolf A. K., Chomsky C. and Laughery K. (1961). *BASEBALL: an Automatic Question Answerer*. In *Proceedings of Western Joint IRE-AIEE-ACM '61 Computer Conference*, pp. 219–224. ACM Press.

He W., Liu K., Liu J., Lyu Y., Zhao S., Xiao X., Liu Y., Wang Y., Wu H., She Q., Liu X., Wu T. and Wang H. (2018). *DuReader: a Chinese Machine Reading Comprehension Dataset from Real-world Applications*. In *Proceedings of the Workshop on Machine Reading for Question Answering*, pp. 37–46, Melbourne, Australia. Association for Computational Linguistics.

Karpukhin V., Oguz B., Min S., Lewis P., Wu L., Edunov S., Chen D. and Yih W.-t. (2020). *Dense Passage Retrieval for Open-Domain Question Answering*. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6769–6781, Online. Association for Computational Linguistics.

Karzewski M. (1997). *Jeden z dziesięciu — pytania i odpowiedzi*. Muza SA.

Katz B. (1997). *Annotating the World Wide Web Using Natural Language*. In *Proceedings of the 5th RIAO Conference on Computer Assisted Information Searching on the Internet (RIAO 97)*, pp. 136–155.

Kłeczek D. (2021). *Simple Recipes for Question Answering*. In Ogrodniczuk and Kobyliński (2021), pp. 159–161.

Krasnowska-Kieraś K. and Wróblewska A. (2019). *Empirical Linguistic Study of Sentence Embeddings*. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 5729–5739, Florence, Italy. Association for Computational Linguistics.

Longpre S., Lu Y. and Daiber J. (2020). *MKQA: A Linguistically Diverse Benchmark for Multilingual Open Domain Question Answering*. arXiv:2007.15207.

Marcińczuk M., Ptak M., Radziszewski A. and Piasecki M. (2013). *Open Dataset for Development of Polish Question Answering Systems*. In Vetulani Z. and Uszkoreit H. (eds.), *Proceedings of the 6th Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics*, pp. 479–483. Wydawnictwo Poznańskie, Fundacja Uniwersytetu im. Adama Mickiewicza.

Marcińczuk M., Radziszewski A., Piasecki M., Piasecki D. and Ptak M. (2013). *Evaluation of a Baseline Information Retrieval for a Polish Open-domain Question Answering System*. In *Proceedings of the International Conference Recent Advances in Natural Language Processing (RANLP 2013)*, pp. 428–435. Association for Computational Linguistics.

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

Mroczkowski R., Rybak P., Wróblewska A. and Gawlik I. (2021). *HerBERT: Efficiently Pretrained Transformer-based Language Model for Polish*. In *Proceedings of the 8th Workshop on Balto-Slavic Natural Language Processing*, pp. 1–10, Kiyv, Ukraine. Association for Computational Linguistics.

Nguyen T., Rosenberg M., Song X., Gao J., Tiwary S., Majumder R. and Deng L. (2016). *MS MARCO: A Human Generated MAchine Reading COmprehension Dataset*. In *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches (CoCo 2016)*, Barcelona, Spain. CEUR Workshop Proceedings.

Ogrodniczuk M. and Kobyliński Ł., editors (2021). *Proceedings of the PolEval 2021 Workshop*, Warsaw, Poland. Institute of Computer Science, Polish Academy of Sciences.

Piotrowski M. (2021). *Search-Augmented Question Answering System Using Multilingual Transformer Model*. In Ogrodniczuk and Kobyliński (2021), pp. 137–140.

Przybyła P. (2015). *Gathering Knowledge for Question Answering Beyond Named Entities*. In Biemann C., Handschuh S., Freitas A., Meziane F. and Métais E. (eds.), *Proceedings of the 20th International Conference on Applications of Natural Language to Information Systems (NLDB 2015)*, pp. 412–417, Passau, Germany. Springer-Verlag.

Przybyła P. (2016). *Boosting Question Answering by Deep Entity Recognition*. arXiv:1605.08675.

Rajpurkar P., Zhang J., Lopyrev K. and Liang P. (2016). *SQuAD: 100,000+ Questions for Machine Comprehension of Text*. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, Austin, Texas. Association for Computational Linguistics.

Rybak P. (2021). *Retrieve and Refine System for Polish Question Answering*. In Ogrodniczuk and Kobyliński (2021), pp. 151–157.

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

Simmons R. F. (1970). *Natural Language Question-Answering Systems: 1969*. „Communications of the ACM", 13(1), p. 15–30.

Smywiński-Pohl A. (2019). *Results of the PolEval 2019 Shared Task 3: Entity Linking*. In Ogrodniczuk M. and Łukasz Kobyliński (eds.), *Proceedings of the PolEval 2019 Workshop*. Institute of Computer Science, Polish Academy of Sciences.

Smywiński-Pohl A., Zhylko D., Wróbel K. and Król M. (2021). *Answering Polish Trivia Questions with the Help of Dense Passage Retriever*. In Ogrodniczuk and Kobyliński (2021), pp. 141–150.

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 30*, pp. 5998–6008. Curran Associates, Inc.

Vetulani Z. (1988). *PROLOG Implementation of an Access in Polish to a Data Base*. In *Studia z automatyki XII*, pp. 5–23. PWN.

Vetulani Z., Marciniak J., Vetulani G., Dąbrowski A., Kubis M., Osiński J., Walkowska J., Kubacki P. and Witalewski K. (2010). *Zasoby językowe i technologia przetwarzania tekstu POLINT-112-SMS jako przykład aplikacji z zakresu bezpieczeństwa publicznego*. Wydawnictwo Naukowe UAM.

Voorhees E. M. (1999). *The TREC-8 Question Answering Track Report*. In *Proceedings of The Eight Text REtrieval Conference (TREC 2000)*, vol. 7. National Institute of Standards and Technology, NIST.

Walas M. and Jassem K. (2010). *Named Entity Recognition in a Polish Question Answering System*. In Kłopotek M. A., Marciniak M., Mykowiecka A., Penczek W. and Wierzchoń S. T. (eds.), *Intelligent Information Systems*, pp. 181–191. Publishing House of University of Podlasie.

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

Xue L., Constant N., Roberts A., Kale M., Al-Rfou R., Siddhant A., Barua A. and Raffel C. (2021). *mT5: A Massively Multilingual Pre-trained Text-to-Text Transformer*. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 483–498, Online. Association for Computational Linguistics.

Zhu F., Lei W., Wang C., Zheng J., Poria S. and Chua T.-S. (2021). *Retrieving and Reading: A Comprehensive Survey on Open-domain Question Answering*. arXiv:2101.00774.

Search-Augmented Question Answering System Using Multilingual Transformer Model

Mateusz Piotrowski

Abstract

This paper presents our submission to PolEval 2021 Quiz Question Answering Challenge task. The proposed system relies on using the mT5 model to generate answers using context information retrieved from Wikipedia. The final solution was able to correctly answer 71.68% questions from the evaluation set. The code for reproducing the results is publicly available.[1]

Keywords

natural language processing, information retrieval, mT5, neural language models, multilingual model, question answering

1. Introduction

Question-answering in an open-domain setting is a challenging task that requires a solution that is able to both efficiently retrieve relevant knowledge and distill it into a concise answer. The first step, which is effectively an information retrieval problem, can be approached using well-known search techniques relying on keyword matching. The second step requires text comprehension abilities to interpret the question and infer the correct answer. Recently, deep neural networks are a tool of choice for this kind of task.

2. Method

The proposed question-answering process is divided into two steps. First, a question is used as a search query to retrieve a set of relevant passages from the knowledge base. In the second step, the top-ranked results are fed along with a question to a neural network, which is responsible for generating the final answer.

[1] `https://github.com/mntss/poleval-2021`

2.1. Search module

We use a dump of Polish Wikipedia to serve as a knowledge source for our solution. In order to retrieve relevant fragments from this corpus, we use ElasticSearch[2] — an open-source search engine based upon Apache Lucene.

The articles from the Wikipedia corpus were split into paragraphs which were then inserted as separate documents into ElasticSearch to create a search index. To increase the search recall, both the article text and search query were lemmatized using a Polish model from spaCy library.[3]

The questions were used as a search query and the returned paragraphs were ordered based on similarity to the search query according to a BM25 scoring function. The input to the neural network was formed by concatenating the question and the original text of top-ranking results until reaching a 512 token limit.

2.2. Model

We use the mT5 model (Xue et al. 2021) for generating the answers — a multilingual transformer model based on original encoder-decoder architecture. The model was pre-trained on a massive text corpus and has been shown by Roberts et al. (2020) to possess good question-answering capabilities even without providing additional context from the knowledge base. This implies that the model itself already has access to the knowledge encoded implicitly in model parameters during pre-training.

The encoder-decoder architecture has the advantage over encoder-only transformer models. The latter were also applied to question answering problems (e.g. Karpukhin et al. 2020), however, such configuration is only capable of performing *extractive* answering which selects a span from context document. On the other hand, the decoder-module is capable of generating arbitrary text, which is necessary for questions that require inferring the correct answer from a context.

The multilingual models can also make a use of a wider array of available datasets. Training using a mixture of examples from different languages allows capturing language-independent features which may improve the model performance on a given task. This property is especially useful in cases where training data is scarce.

2.3. Training procedure

We train the model using a combination of examples from dev-0 and test-A sets provided by organizers. We extract context passages from the Polish Wikipedia dump from December 2020 provided as a part of the TensorFlow Datasets project.[4]

[2] `https://github.com/elastic/elasticsearch`
[3] `https://spacy.io/models/pl`
[4] `https://www.tensorflow.org/datasets/catalog/wikipedia#wikipedia20201201pl`

We also test a setup where we further extend the training dataset with English examples from the TriviaQA dataset introduced by Joshi et al. (2017). The questions already come with context passages, however we reassign them by following the process from Section 2.1. The paragraphs are inserted to ElasticSearch and then ones returned by a search query are selected as a context. This approach is meant to produce more similar examples across languages. The TriviaQA sample contains 87 622 examples, while the combination of dev-0 and test-A sets contains 3500 examples. To avoid overfitting the examples from the Polish dataset we sample the examples with a 20:1 ratio.

For comparison purposes, similarly to our submission for Task 3, we fine-tune *base*, *large* and *XXL* variants of the mT5 model. To verify the capacity of those models to encode real-world knowledge in network parameters, we evaluate a setting where the input consists only of questions from combined datasets without the Wikipedia context. In this setup, the models were trained for 10 000 steps. For Polish only and combined dataset 5000 and 20 000 steps were used respectively.

3. Result summary

In this section, we present the comparison of results obtained by training different configurations. Table 1 presents the scores obtained by different configurations. Even without using Wikipedia the largest model was able to correctly answer over 36% of the questions, significantly more than the smaller models. When retrieved-context passages were provided with the question, the best configuration was able to achieve the winning score of 71.68%. Extending the training dataset with examples from TriviaQA had a major impact only on *base* model while providing modest improvement for *large* and *XXL* configurations.

Table 1: Percentage of correctly answered questions from test-B set

Model	**No context**	**PL only**	**Combined**
base	17.00	47.64	52.12
large	22.76	58.12	59.20
XXL	36.20	70.76	71.68

4. Conclusions

In this paper, we presented a simple system for question answering. The combination of Wikipedia knowledge retrieval and multilingual transformer model allowed to correctly answer 71.68% of the questions from the evaluation set.

The ability of the neural network to generate accurate answers is partly due to the knowledge obtained during the pre-training process. Roberts et al. (2020) suggest it can be further improved by modifying the unsupervised objective, by restricting the masked tokens to named entities to focus on real-world knowledge.

The search module also leaves an area for improvement. For example Guu et al. (2020) and Karpukhin et al. (2020) report advantages of using "dense retrieval". Instead of relying on sparse document representations such as TF-IDF, the passages are encoded using embeddings and then ranked according to the inner product between document and query.

Acknowledgments

Research supported with Cloud TPUs from Google's TPU Research Cloud (TRC).[5]

References

Guu K., Lee K., Tung Z., Pasupat P. and Chang M. (2020). *Retrieval Augmented Language Model Pre-training*. In *Proceedings of the 37th International Conference on Machine Learning (ICML 2020)*, vol. 119 of *Proceedings of Machine Learning Research*, pp. 3929–3938. PMLR.

Joshi M., Choi E., Weld D. and Zettlemoyer L. (2017). *TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension*. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Karpukhin V., Oguz B., Min S., Lewis P., Wu L., Edunov S., Chen D. and Yih W.-t. (2020). *Dense Passage Retrieval for Open-Domain Question Answering*. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP 2020)*, pp. 6769–6781, Online. Association for Computational Linguistics.

Roberts A., Raffel C. and Shazeer N. (2020). *How Much Knowledge Can You Pack Into the Parameters of a Language Model?* In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 5418–5426, Online. Association for Computational Linguistics.

Xue L., Constant N., Roberts A., Kale M., Al-Rfou R., Siddhant A., Barua A. and Raffel C. (2021). *mT5: A Massively Multilingual Pre-trained Text-to-Text Transformer*. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 483–498, Online. Association for Computational Linguistics.

[5] `https://sites.research.google/trc/about/`

Answering Polish Trivia Questions with the Help of Dense Passage Retriever

Aleksander Smywiński-Pohl, Dmytro Zhylko
(AGH University of Science and Technology / Enelpol)

Krzysztof Wróbel (Enelpol / Jagiellonian University / AGH University of Science and Technology)

Magdalena Król (AGH University of Science and Technology)

Abstract

This paper discusses the problem of Question answering using Dense Passage Retriever in Task 4 during 2021 edition of PolEval. Our goal was to show the process of automatic answering trivia questions using language models and Wikipedia database. The best solution created by the authors utilized Dense Passage Retrieval approach for extractive question answering combined with Natural Language Inference for boolean questions. The training data for document retrieval and extractive question answering were obtained by employing distant supervision. The obtained solution reached 50.96% accurracy giving second place in the competition.

Keywords

question answering, dense passage retrieval, Polish language model, trivia questions, Wikipedia

1. Introduction

Question answering (QA) is one of the most interesting tasks within Natural Language Processing (NLP). Since Alan Turing has defined his famous test for intelligence (Turing 2009), question answering remains one of the most challenging issues related to human and artificial intelligence. However, a system able to answer any question a human can answer easily remains an unfinished goal. To achieve this goal a growing number of QA datasets is made available: SQuAD (Rajpurkar et al. 2016), WikiQA (Yang et al. 2015), CNN/DailyMail (Hermann et al. 2015), MS MARCO (Bajaj et al. 2016), TriviaQA (Joshi et al. 2017), Natural

questions (Kwiatkowski et al. 2019), to name just the most popular of them. Yet, as with many NLP resources, these datasets are available only for English. In Polish, there are only a few datasets related to QA: "Czy wiesz"[1], Polish Legal Question Answering Dataset (LQuAD-PL, to appear) and the dataset which was made available during the PolEval 2021 competition. The last dataset is an interesting one since it includes only the questions and the answers as the training data. Most of the remaining datasets contain excerpts relevant for answering the question.

There are several of competing approaches related to question answering. They might be roughly categorized into: extractive QA, abstractive QA and closed-book QA. In an end-to-end system the first two approaches are usually supplemented by a selective QA module, frequently called a retriever. A typical complete QA pipeline for the first two approaches is as follows: from a large body of documents (e.g. passages from Wikipedia) the retriever selects a much smaller subset. In the next step the reader inspects them and either selects a span of consecutive tokens (for extractive QA) or forms an answer, using the selected passages as input in a sequence-to-sequence setting (for abstractive QA). Closed-book QA is the most recent trend, which is made possible thanks to the availability of extremely large language models, such as GPT-3 (Brown et al. 2020), T5 (Raffel et al. 2020) and mT5 (Xue et al. 2021). These generative solutions are able to answer questions using only the world knowledge preserved in the model itself. Although they do not achieve scores as high as the remaining approaches, they reflect the human ability of answering a question without consulting any external knowledge sources.

The experiments conducted by the authors during the PolEval 2021 competition were designed to explore the first and the last approach. The majority of them employed the extractive QA paradigm, which is implemented in DPR (Dense Passage Retriever; Karpukhin et al. 2020). Yet, for yes/no questions this approach seems to be invalid. For this reason, we employed an approach based on Natural Language Inference. We also conducted some experiments following the closed book approach, which utilizes a sequence-to-sequence architecture.

2. Data

The data provided by the organizers included questions and answers from the Polish TV competition called "Jeden z dziesięciu" (translated literally as "One out of ten", a Polish version of the British TV show "Fifteen to one"). It included three subsets: dev-0 (1000 question and answer pairs), test-A (2500 question and answer pairs; the answers were provided during the last stage of the competition) and test-B (2500 question and answer pairs; the answers were provided after the competition finished).

According to the rules of the competition, both dev-0 and test-A could be used to train the QA model. We have decided to use test-A as a base for the training set and dev-0 as a base for the validation set.

[1] `http://nlp.pwr.wroc.pl/en/tools-and-resources/resources/czy-wiesz-question-answering-dataset`

A sample QA pair from the dataset is given below:

(PL)	Urodę której części twarzy podkreśla mascara?	rzęs
(EN)	Which part of the face is highlighted by mascara?	eyelashes

Polish is an inflected language. The organizers decided to use that feature and required that the answer is correctly inflected for number, case and possibly gender. This is not fully visible in the translation, since English does not inflect for case, but the answer "rzęs" is a plural form in genitive, while a typical entry in a dictionary would be "rzęsa" which is a singular form in the nominative case. Matching the answer exactly would be impossible for the extractive approach for most of the examples, making that approach impractical. But the authors of the competition decided to use a different scoring function. They compute a Levenshtein distance between the provided answer and the reference answer. If the distance divided by the length of the reference answer was smaller than 0.5, the answer was treated as correct. Since the inflected part of the word is usually shorter than half of the word, for most of the answers a nominal case or some other case appearing directly in the text was enough to treat the answer as correct, assuming the proper word or words were selected.

Although this approach was feasible for scoring the final results, it was not helpful for training the QA models (both the retriever and the reader). For the retriever, it was necessary to decide whether a given text snippet includes the answer. For the reader, it was necessary to precisely indicate the consecutive sequence of tokens, which are the answer. To resolve these issues, a lemmatizer from the KRNNT library was employed (Wróbel 2017) – when matching the snippet for the answer the literal and the base forms of the tokens in the answer were matched against the literal and the base forms of the tokens in the snippet. The answers were also pre-processed in order to remove the leading preposition since some of them included such a term (e.g. "we Włoszech" – "in Italy").

3. Method

3.1. Distant supervision

As explained in Section 2, the dataset didn't include text snippets that could be used to train the retriever and the reader models directly. A distant supervision approach was employed to resolve this issue. The authors decided to use only Polish Wikipedia as the reference corpus with the snippets. The Wikipedia dump was processed using WikiExtract library[2]. The authors used a dump of Wikipedia from 2019, which is available on the PolEval website[3] (Smywiński-Pohl 2019). The WikiExtract library removes elements such as navigation structure and so-called infoboxes, leaving a clean Wikipedia text, preserving the Wikipedia links in a separate file[4]. The only additional step that was necessary was splitting the text into snippets. Since the text was already sentence-split, it was possible to create snippets containing whole

[2] `https://github.com/cycloped-io/wikiextract`
[3] `http://2019.poleval.pl/index.php/tasks/task3`
[4] The links were not utilized in the experiment.

sentences. The authors decided to use snippets containing up to 90 words, allowing for longer sequences to preserve full sentences.[5]

To select a set of candidate snippets for a given question, we have used a dense passage representation provided by DPR. The model training followed the same procedure as described in Section 3.2 (HerBERT-base was used as the base model), but the training data included only "Czy wiesz" and LQuAD-PL datasets. The model was used to select 100 answer candidates for each question. The candidates were lemmatized and matched against the answers. The longest common subsequence algorithm was used to match the parts of the answer. The longest subsequence of consecutive tokens was returned as the result, allowing for gaps up to 2 tokens. It was required that at least 80% of the tokens from the answer have to be present in the candidate snippet, to treat it as a positive example. Otherwise, the snippet was treated as a negative example. The positive candidate examples were inspected in the order provided by the DPR retriever, and the snippet with the largest number of matched tokens was selected as the reference, positive example. In the case of a tie, a snippet higher on the DPR rank list was selected. The snippets prepared in that way served as training examples for both the retriever and the reader. The negative examples were treated as the hard negative examples in the DPR parlance, since they were similar in content to the positive example. In the case of the reader, the matched sequence served as the reference answer.

The retriever dataset needs one additional remark: the training set included a single positive example selected according to the above rules and a number of hard negative examples. Yet, this scheme may not be applied to the validation dataset! This might be obvious, since the validation dataset has to reflect the test dataset, however, the usual practice allows to select the validation dataset as the subset of the given training set. This mistake was in fact made by the authors, leading to detrimental results in a number of experiments.

3.2. Retriever

We have used Dense Passage Retriever (DPR; Karpukhin et al. 2020) as the retriever module. We have employed HerBERT-base[6] and HerBERT-large[7] as the starting language models (Mroczkowski et al. 2021). They were trained on "Czy wiesz", LQuAD-PL and the training subset of the dataset described in the Section 3.1. The top-1 accuracy of the models was 63.6% and 68.0% respectively. The best models were selected according to the accuracy on the validation set. All models were trained for 20 epochs with the best checkpoints on 15-th and 16-th epoch for smaller and larger model accordingly. FAISS (Johnson et al. 2021) was used to store the dense representation of the snippets. Since we didn't have to optimize for retrieval speed, we have used the flat access mode. The server required approx. 40GB of RAM to store all Wikipedia passages (approx. 3.750 million passages).

[5] This value was somehow arbitrary, but corresponds roughly to 100 tokens-length reported in the DPR paper (Karpukhin et al. 2020).

[6] `https://huggingface.co/allegro/herbert-base-cased`

[7] `https://huggingface.co/allegro/herbert-large-cased`

3.3. Reader

The reader model we used comes also from DPR, which combines a re-ranker with a model able to select the span of tokens containing the answer for the question. DPR uses a simple approach regarding the selection of the best answer: at first, it sorts the passages provided by the retriever and selects the first passage; then, it selects the answer tokens in that passage according to the reader model. It should be noted that the model used to re-rank the passages has access to full attention between the question and the passage, while the retriever optimizes the dot-product between the dense representations of the question and the answer. Yet, both models are trained using a similar approach – the negative examples are the passages provided explicitly in the training set and in-batch negatives, i.e. the other passages that are present in the batch. This allows for more efficient training and provides one important hyperparameter, i.e. the number of hard negatives used for training. This value combined with the batch size determines the total number of negatives for a given positive example and directly impacts the memory requirements of this method.

3.4. Boolean questions

Boolean questions, such as "Czy w państwach starożytnych powoływani byli posłowie i poselstwa?" ("Were envoys and legations called in the ancient countries?") cannot be answered using the extractive approach. Such questions were identified by a regular expression, matching "Czy" ("whether") at the beginning of the questions and excluding this word in its remaining part. The exclusion was necessary, since questions requiring selection of one of the provided options, such as "Czy sombrero to kapelusz, danie czy taniec?" ("Is sombrero a hat, a dish or a dance?") also have "Czy" at the beginning.

The approach applied to the boolean questions was the following: all passages returned by DPR were selected as the first sentence for Natural Language Inference task, while the question that yielded these passages was treated as the second sentence. If the answer to the question was "yes", the pairs of sentences were marked as entailment – if it was "no", they were marked as a contradiction. A separate set of questions with random passages were marked as neutral for the given question. HerBERT-large model was fine-tuned on the CDSCorpus (Krasnowska-Kieraś and Wróblewska 2019, Wróblewska and Krasnowska-Kieraś 2017) and then further tuned on the above dataset. During inference for the question and the retrieved passages the model was used to determine the relation between them. Pairs with neutral relations were discarded. The answer was selected as a majority vote between entailment and contradiction, with entailment ("yes") winning in the case of a tie.

3.5. Closed-book QA

A closed-book method was also tested on the provided dataset. This approach does not require a set of passages to read the answer, since it employs a sequence-to-sequence paradigm, where the question is the input sequence, while the answer is the output sequence. The only knowledge available is the model itself. This method may yield any reasonable results

only if the pre-trained model is very large and it predicts the next token (i.e. it is a *classical* language model). The number of such models supporting Polish is very small: PapuGaPT-2 (Wojczulis and Kłeczek 2021), plT5-large[8] and mT5 models family (Xue et al. 2021). Since the training of such models is time-consuming, we have decided to test only the last family, aiming at using the largest models: mT5-large, mT5-xl and mT5-xxl. The last model is especially problematic, since its size is larger than the VRAM capacity of the most popular V100 GPUs. Transformers library provides experimental support for running that model on multiple cards (via parallel model feature[9]) and even on one card (e.g. Deepspeed (Rasley et al. 2020) provides a CPU-offload feature which is integrated into Transformers[10]).

Yet, these features are experimental and setting up a working pipeline is currently a cumbersome experience. A (costly) alternative is usage of Google TPU-pods which provide direct and well-tested support for very large models (mT5 was trained on Google TPU platform). For that reason we have decided to test the closed-book approach using the Google infrastructure.

4. Experiments

Approximately 30 experiments were conducted in order to estimate the impact of the different features and parameters on the final results:

— using different strategies regarding the title of the article,

— using different snippet searching strategies (sparse vs. dense passage retrieval),

— using a different number of negative contexts,

— using different sizes of the model,

— using different training data,

— using different strategies for the validation dataset.

Since the research was driven by the competition, none of the above features and strategies were tested systematically. Still, some observations may be drawn based on the partial results.

The most important observation was a correct preparation of the validation dataset – in many experiments this dataset was taken directly from the training dataset, while that dataset included only one snippet with the correct answer. The experiments conducted in that setting showed that with the growing number of candidate snippets, the result deteriorated. Changing the validation dataset to include all snippets returned by the retriever showed that it is necessary to include 80 snippets to obtain the best result. Fixing this error resulted in test accuracy jumping from 24.84% to 46.84% (22 pp.).

The second important observation regarded the size of the model used to train the reader. By substituting HerBERT-base with HerBERT-large, we have obtained almost 11 pp. improvement on the validation dataset – the best result for the *base* model was 49.62%, while the best

[8] `https://huggingface.co/allegro/plt5-large`
[9] `https://github.com/huggingface/transformers/pull/7772`
[10] `https://huggingface.co/transformers/main_classes/deepspeed.html`

result for the *large* model was 60.61%. This result is consistent with the observation that the winning submission used mT5-xxl model, which is the largest pre-trained model including support for Polish.

The third important observation was assessing the impact of the number of negative examples on the results on the validation dataset. Changing the default value of 3 negative examples to 10 gave more than 11 pp. improvement (21.22% to 32.88%), while 15 negative examples gave further 17 pp. improvement (32.88% to 49.62%). So the correct number of negative examples has a great impact on the final result.

5. Results

The best result obtained by the team was 50.96 giving the second place on the leaderboard (equally with Piotr Rybak's submission) – cf. Table 1.

Table 1: The final results of the competition

Contestant name	**Result**
Mateusz Piotrowski	71.68
Aleksander Smywiński-Pohl	**50.96**
Piotr Rybak	50.96
Darek Kłeczek	46.44
Karol Gawron	36.12
BI Insight	0.96

That result was obtained combining the following elements:

1. The DPR retriever used to generate the passages was trained on "Czy wiesz" dataset, LQuAD-PL and the distantly-supervised training part of the PolEval Task-4 (i.e. the data described in Section 3.1).
2. The DPR retriever model was based on HerBERT-base.
3. Top-80 passages were used during inference.
4. The DPR reader model was based on HerBERT-large and it was trained only on the training part of the PolEval Task-4.
5. There were 12 negative contexts used during training.
6. The batch size was limited to 1.
7. The number of epochs/steps was 1.92.
8. Boolean questions were answered according the the NLI model, trained on CDSCorpus and the data from Section 3.1 and utilized HerBERT-large.

Regarding the closed-book approach, the best result on the validation dataset was 19.60%. It was obtained using mT5-xl model trained for 1000 steps.[11] Due to high cost of the TPUs we have not run the inference on the test dataset.

6. 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" and in part by the PLGrid Infrastructure.

References

Bajaj P., Campos D., Craswell N., Deng L., Gao J., Liu X., Majumder R., McNamara A., Mitra B., Nguyen T., Rosenberg M., Song X., Stoica A., Tiwary S. and Wang T. (2016). *MS MARCO: A Human Generated MAchine Reading COmprehension Dataset*. arXiv:1611.09268.

Brown T. B., Mann B., Ryder N., Subbiah M., Kaplan J., Dhariwal P., Neelakantan A., Shyam P., Sastry G., Askell A., Agarwal S., Herbert-Voss A., Krueger G., Henighan T., Child R., Ramesh A., Ziegler D. M., Wu J., Winter C., Hesse C., Chen M., Sigler E., Litwin M., Gray S., Chess B., Clark J., Berner C., McCandlish S., Radford A., Sutskever I. and Amodei D. (2020). *Language Models are Few-Shot Learners*. In Larochelle H., Ranzato M., Hadsell R., Balcan M. and Lin H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems (NeurIPS 2020)*, vol. 33, pp. 1877–1901. Curran Associates, Inc.

Hermann K. M., Kočiský T., Grefenstette E., Espeholt L., Kay W., Suleyman M. and Blunsom P. (2015). *Teaching machines to read and comprehend*. In *Proceedings of the 28th International Conference on Neural Information Processing Systems (NIPS'15). Volume 1*, pp. 1693–1701, Cambridge, MA, USA. MIT Press.

Johnson J., Douze M. and Jégou H. (2021). *Billion-Scale Similarity Search with GPUs*. „IEEE Transactions on Big Data", 7, p. 535–547.

Joshi M., Choi E., Weld D. and Zettlemoyer L. (2017). *TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension*. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Karpukhin V., Oguz B., Min S., Lewis P., Wu L., Edunov S., Chen D. and Yih W.-t. (2020). *Dense Passage Retrieval for Open-Domain Question Answering*. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP 2020)*, pp. 6769–6781, Online. Association for Computational Linguistics.

[11] Finally due to financial and time constraints we haven't decided to use the mT5-xxl model.

Krasnowska-Kieraś K. and Wróblewska A. (2019). *Empirical Linguistic Study of Sentence Embeddings*. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 5729–5739, Florence, Italy. Association for Computational Linguistics.

Kwiatkowski T., Palomaki J., Redfield O., Collins M., Parikh A., Alberti C., Epstein D., Polosukhin I., Devlin J., Lee K., Toutanova K., Jones L., Kelcey M., Chang M.-W., Dai A. M., Uszkoreit J., Le Q. and Petrov S. (2019). *Natural Questions: A Benchmark for Question Answering Research*. „Transactions of the Association for Computational Linguistics", 7, p. 452–466.

Mroczkowski R., Rybak P., Wróblewska A. and Gawlik I. (2021). *HerBERT: Efficiently pretrained transformer-based language model for Polish*. In *Proceedings of the 8th Workshop on Balto-Slavic Natural Language Processing*, pp. 1–10, Kiyv, Ukraine. Association for Computational Linguistics.

Raffel C., Shazeer N., Roberts A., Lee K., Narang S., Matena M., Zhou Y., Li W. and Liu P. J. (2020). *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*. „Journal of Machine Learning Research", 21(140), p. 1–67.

Rajpurkar P., Zhang J., Lopyrev K. and Liang P. (2016). *SQuAD: 100,000+ Questions for Machine Comprehension of Text*. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, Austin, Texas. Association for Computational Linguistics.

Rasley J., Rajbhandari S., Ruwase O. and He Y. (2020). *Deepspeed: System Optimizations Enable Training Deep Learning Models with over 100 Billion Parameters*. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 3505–3506.

Smywiński-Pohl A. (2019). *Results of the PolEval 2019 Shared Task 3: Entity Linking*. In Ogrodniczuk M. and Łukasz Kobyliński (eds.), *Proceedings of the PolEval 2019 Workshop*. Institute of Computer Science, Polish Academy of Sciences.

Turing A. M. (2009). *Computing Machinery and Intelligence*. In Epstein R., Roberts G. and Beber G. (eds.), *Parsing the Turing Test: Philosophical and Methodological Issues in the Quest for the Thinking Computer*, pp. 23–65. Springer Netherlands, Dordrecht.

Wojczulis M. and Kłeczek D. (2021). *papuGaPT2 — Polish GPT2 Language Model*. `https://huggingface.co/flax-community/papuGaPT2`.

Wróbel K. (2017). *KRNNT: Polish Recurrent Neural Network Tagger*. In *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.

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

Xue L., Constant N., Roberts A., Kale M., Al-Rfou R., Siddhant A., Barua A. and Raffel C. (2021). *mT5: A Massively Multilingual Pre-trained Text-to-Text Transformer*. In *Proceedings*

of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 483–498, Online. Association for Computational Linguistics.

Yang Y., Yih W.-t. and Meek C. (2015). *WikiQA: A challenge dataset for open-domain question answering*. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 2013–2018, Lisbon, Portugal. Association for Computational Linguistics.

Retrieve and Refine System for Polish Question Answering

Piotr Rybak (ML Research at Allegro.pl)

Abstract

This paper presents our solution to PolEval 2021 task on Question Answering[1]. The goal of the task was to develop a system to answer factoid questions about general knowledge. Our solution consists of two components, a passage retriever that finds relevant Wikipedia passages and an answer generator to generate the answer based on retrieved passages. In the official evaluation, our system obtained an accuracy of 50.96% and ranked second (ex aequo with one other system).

Keywords

natural language processing, question-answering, HerBERT, plT5

1. Introduction

Question answering (QA) is a broad term defining a task of answering a question, typically formulated in natural language. Over the years, many QA subtasks emerged differing in scope, domain, or availability of additional information. One popular subtask is an open-domain question answering (Voorhees and Tice 2000). Its goal is to answer a factoid question using a large collection of documents that might contain the correct answer.

Although many different methods were proposed to solve open-domain QA, recently the best results are obtained by two-stage systems (Chen et al. 2017). First, a retriever selects a small subset of passages that hopefully contains the answer, and then a reading comprehension model reads the retrieved passages and extracts (or generates) the answer.

The goal of this PolEval 2021 QA task is to build an open-domain QA system to answer factoid questions inspired by a popular TV show called *Jeden z dziesięciu*. The organizers provided three datasets, one development (1000 question-answer pairs) and two test sets (2500 question-answers pairs each). The additional challenge of this task is the lack of the training set.

[1] `https://github.com/poleval/2021-question-answering`

2. Related work

There are only a few previous works regarding question answering systems for Polish language. Marcińczuk et al. (2013) gathered a set of question-answer pairs from different Polish sources and compared two re-ranking methods for retrieving correct passages. The first complete open-domain QA system which uses a plain-text knowledge base is RAFAEL (Przybyła 2016). It retrieves relevant passages and extracts from them named entities that answer factoid questions. Recently, Borzymowski (2020) automatically translated the established SQuAD2.0 (Rajpurkar et al. 2018) dataset into Polish and trained a reading comprehension model[2] to extract answer from matching Wikipedia passage.

3. Dataset

There is no established dataset for Polish open-domain question answering. To train a system for this task, we decided to gather multiple available resources and combine them into the final training set.

Jeden z dziesięciu is a Polish TV show during which participants answer the host's questions from various fields. We created a dataset consisting of 1004 questions from past episodes[3].

Multi-lingual Knowledge Questions & Answers (MKQA) (Longpre et al. 2020) is a dataset of 10,000 queries sampled from the Google Natural Questions dataset (Kwiatkowski et al. 2019) and then manually translated into 26 different languages. We used Polish part of the dataset and filtered out questions without answers, with open-ended answers, about current events, TV shows, and movies. Eventually, we left 1875 questions that we considered useful.

PolEval organizers published three QA datasets: one development (1000 question-answer pairs) and two test sets (2500 question-answers pairs each). Since the final evaluation used only the second test set, we used the development and the first test set as the training set.

Final dataset dataset combines the aforementioned datasets, which resulted in the training set of 3879 questions and the official test set of 2500 questions.

4. Passage retriever

Our system consists of two main components, a passage retriever and an answer generator. The goal of the passage retriever is to understand the question and find the passage which contains the answer. To achieve that we need two elements. First, a large number of documents to serve as a knowledge base from which we will retrieve passages containing answers. Second, a way to correctly search for such passages.

[2] `https://hf.co/henryk/bert-base-multilingual-cased-finetuned-polish-squad2`

[3] We parsed pdfs available here: `http://tvturnieje.blogspot.com/p/jeden-z-dziesieciu.html`

4.1. Passages

After manually analyzing a small subset of questions, we decided to use Wikipedia as a main source of documents. Manual verification showed that most question requires general encyclopedic knowledge which means that relevant passages can be easily found on Wikipedia. Additionally, we added a list of Polish proverbs and idioms as some questions required such information.

Each Wikipedia article was parsed and split into paragraphs using Wikiextractor (Attardi 2015). The final passages consisted of article titles concatenated with the content of individual paragraphs trimmed to the first 300 characters.

4.2. Neural retriever

Recently, the best results in passage retrieval are obtained by neural encoders (Karpukhin et al. 2020) trained to encode matching questions and passages into similar dense representations. Unfortunately, the organizers of the task provided only question-answer pairs (not question-passage) so it is not straightforward to train such an encoder.

To overcome the lack of a manually annotated dataset of question-passage pairs, we decided to create such a dataset ourselves. First, we used a simple Bag-of-Words retriever as well as a more sophisticated model such as Universal Sentence Encoder (Yang et al. 2019) to retrieve passage candidates for questions from the training set. After annotating a sample of question-passage pairs, we used them to train the first version of the neural retriever and retrieve more relevant candidates. We repeated this procedure multiple times. We also trained the encoder using the information about the answer to boost the performance of the retriever and obtain a higher ratio of positive question-passage pairs. Overall, we annotated 10k pairs out of which there were 2215 matching passages for 1347 unique questions[4].

We combined our manually annotated dataset with *Czy wiesz?* dataset (Marcińczuk et al. 2013, Rybak et al. 2020) to train the final neural retriever. We used HerBERT Large (Mroczkowski et al. 2021) and fine-tuned it using minimizing contrastive loss (Hadsell et al. 2006) for 3 epochs. We used Adam (Kingma and Ba 2014) optimizer with learning rate equal to 10^{-5} and batch size of 15 question-passage pairs.

5. Answer generator

The goal of an answer generator component is to thoroughly analyze both question and the passages provided by a neural encoder and extract the final answer. Typically, this could be solved by locating and returning the span of passage that contains the correct answer. However, after examining the dataset, we realized that for many questions the correct answer doesn't occur directly in the retrieved passage. For example, there are around 6% yes/no questions in the test dataset. Therefore, we decided to use encoder-decoder architecture to

[4]Final dataset is published here: `https://hf.co/datasets/allegro/polish-question-passage-pairs`

solve this task. The model takes a question and retrieved passages as an input and is trained to generate an answer as an output.

We used plT5 Base model[5] as a base for our answer generator. The training consisted of two phases. First, we fine-tuned the model for 10 epochs using only positive question-passage pairs from our manually annotated dataset used previously for training the neural retriever. This is a form of curriculum learning, as the model sees only a single passage for each question and it always contains the correct answer. As a next step, we used the neural retriever to find 10 most relevant passages for each question in the training set and concatenated them together into a single sequence. We used such a dataset to fine-tune the answer generator for additional 5 epochs. This phase of fine-tuning was more difficult for the model, since not always the correct answer was present in retrieved passages. In both phases, used Adam (Kingma and Ba 2014) optimizer with learning rate equal to 10^{-4} and batch size of 15 sequences.

6. Results

The task uses a custom metric to evaluate whether the predicted answer is correct or not. For non-numerical questions, we look at text similarity between predicted and gold standard answers. If Levenshtein distance (Levenshtein 1965) is lower than 50%, then we accept the predicted answer as correct. For numerical answers, we require an exact match. In both cases, there might be multiple gold standard answers. In particular, for numerical values it doesn't matter if answers will be written as numbers (e.g. 42) or words (e.g. "forty-two"). To calculate the final score, we calculate the accuracy across all questions.

Our final system obtains accuracy of 50.96% and ranked second in the official evaluation (ex aequo with one other system). To analyze how our system performs for different types of questions, we divided the test dataset into multiple slices. The results can be found in Table 1.

On average, the model performs similarly for questions with numerical and non-numerical answers. However, for questions about dates the model scores much higher, 61.22% when asking for years and 85.19% if the answer is a century.

Binary questions also obtain higher accuracy than average, which is expected since there are only two possible answers for such questions. However, when looking separately at results for 'yes' and 'no' answers, we see that model mostly learned to predict the majority class ('yes' in this case). This can be also explained by the bias in system architecture. It is much easier to find a relevant passage if an answer is positive rather than negative since Wikipedia will contain only positive passages.

One of the most common types of questions is asking for an object name based on its definition, e.g. "What is the name of a *double hull boat*?". Almost 17% of all questions belong to this slice. Intuitively this might be considered a relatively easy type of question. The main difficulty is for the neural retriever to find a Wikipedia entry with a definition as similar to the question as possible. Then, the answer generator can just return the name of the Wikipedia article.

[5] `https://hf.co/allegro/plt5-base`

Table 1: Evaluation results across different slices of test dataset

Dataset slice	Number of questions	Accuracy
Number	268	50.37
Century	54	85.19
Year	49	61.22
Other	165	35.76
Binary	157	66.88
Yes	97	98.97
No	60	15.00
Definition	417	41.97
Person	304	54.18
Choice	226	58.41
Capital	21	95.24
Proverb	10	80.00
Other	1124	47.06
Total	2500	50.96

Surprisingly, the model performs rather poorly on this type of question and obtains an accuracy of 41.97%, much lower than average.

Another interesting slice is 'Choice'. Those questions present two choices and ask to select one of them, e.g. "Is the epilogue *the end* of the work or its *introduction*?". Although, our system scores higher than average on those questions (58.41%), it is not much higher than random, considering that there are usually only two alternatives to select from. One explanation for such low results could be the inability of the answer generator to learn to choose one of those alternatives and instead to generate a completely different answer. However, manual inspection shows this is not the case. In 95% of questions, the model predicted one of the proposed alternatives. It learned to choose one of them but it doesn't know which one is correct. Effectively, it selects the correct answer only slightly better than randomly.

Finally, the model achieves high results when asked about countries' capital (95.24%) or to finish the proverb (80.00%). For those questions, it is relatively easy to find a relevant passage.

7. Conclusions

In this work, we described our solution to PolEval 2021 task on Question Answering. The system consists of two components, the neural retriever and the answer generator. Overall, we obtained an accuracy of 50.96% and ranked second in the official evaluation. However, the error analysis on test set slices shows that model quality is unsatisfactory. For binary questions, it predicts the majority class. For 'Choice' questions accuracy is only a little higher than random. The open-domain QA for Polish language remains an unsolved task with a need for further improvements.

References

Attardi G. (2015). *Wikiextractor*. `https://github.com/attardi/wikiextractor`. GitHub repository.

Borzymowski H. (2020). *Multilingual BERT Fine-tuned on Polish SQuAD2.0*. `https://huggingface.co/henryk/bert-base-multilingual-cased-finetuned-polish-squad2`.

Chen D., Fisch A., Weston J. and Bordes A. (2017). *Reading Wikipedia to Answer Open-Domain Questions*. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1870–1879, Vancouver, Canada. Association for Computational Linguistics.

Hadsell R., Chopra S. and LeCun Y. (2006). *Dimensionality Reduction by Learning an Invariant Mapping*. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, pp. 1735–1742. IEEE.

Karpukhin V., Oguz B., Min S., Lewis P., Wu L., Edunov S., Chen D. and Yih W.-t. (2020). *Dense Passage Retrieval for Open-Domain Question Answering*. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6769–6781, Online. Association for Computational Linguistics.

Kingma D. P. and Ba J. (2014). *Adam: A Method for Stochastic Optimization*. arXiv:1412.6980.

Kwiatkowski T., Palomaki J., Redfield O., Collins M., Parikh A., Alberti C., Epstein D., Polosukhin I., Devlin J., Lee K., Toutanova K., Jones L., Kelcey M., Chang M.-W., Dai A. M., Uszkoreit J., Le Q. and Petrov S. (2019). *Natural Questions: A Benchmark for Question Answering Research*. „Transactions of the Association for Computational Linguistics", 7, p. 452–466.

Levenshtein V. I. (1965). *Binary Codes Capable of Correcting Deletions, Insertions, and Reversals*. „Soviet physics. Doklady", 10, p. 707–710.

Longpre S., Lu Y. and Daiber J. (2020). *MKQA: A Linguistically Diverse Benchmark for Multilingual Open Domain Question Answering*. arXiv:2007.15207.

Marcińczuk M., Ptak M., Radziszewski A. and Piasecki M. (2013). *Open Dataset for Development of Polish Question Answering Systems*. In Vetulani Z. and Uszkoreit H. (eds.), *Proceedings of the 6th Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics*, pp. 479–483. Wydawnictwo Poznańskie, Fundacja Uniwersytetu im. Adama Mickiewicza.

Marcińczuk M., Radziszewski A., Piasecki M., Piasecki D. and Ptak M. (2013). *Evaluation of Baseline Information Retrieval for Polish Open-domain Question Answering System*. In *Proceedings of the International Conference Recent Advances in Natural Language Processing (RANLP 2013)*, pp. 428–435, Hissar, Bulgaria. INCOMA Ltd.

Mroczkowski R., Rybak P., Wróblewska A. and Gawlik I. (2021). *HerBERT: Efficiently Pretrained Transformer-based Language Model for Polish*. In *Proceedings of the 8th Workshop on Balto-Slavic Natural Language Processing*, pp. 1–10. Association for Computational Linguistics.

Przybyła P. (2016). *Boosting Question Answering by Deep Entity Recognition*. arXiv:1605.08675.

Rajpurkar P., Jia R. and Liang P. (2018). *Know What You Don't Know: Unanswerable Questions for SQuAD*. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 784–789, Melbourne, Australia. Association for Computational Linguistics.

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

Voorhees E. and Tice D. (2000). *The TREC-8 Question Answering Track Evaluation*. In *Proceedings of the 8th Text Retrieval Conference*, pp. 83–106. NIST Special Publication 500-246.

Yang Y., Cer D., Ahmad A., Guo M., Law J., Constant N., Abrego G. H., Yuan S., Tar C., Sung Y.-H. et al. (2019). *Multilingual Universal Sentence Encoder for Semantic Retrieval*. arXiv:1907.04307.

Simple Recipes for Question Answering

Dariusz Kłeczek (skok.ai)

Abstract

This paper presents my contributions to PolEval 2021 Task 4: Question Answering Challenge[1]. My final solution is based on deepset Haystack framework with BM25 retriever and XLM-Roberta-Large reader, trained on English SQUAD. Significant gains in my solution were driven by expanding the document store beyond Polish Wikipedia.

Keywords

natural language processing, open domain question-answering, transformers

1. Introduction

I was excited about this task since it was introduced in this year's PolEval competition. The goal was to develop a solution capable of providing answers to general-knowledge questions typical for popular TV quiz shows, such as Fifteen to One. In the sections below I'll describe my journey to the final solution.

2. Closed book approach with PapuGaPT2

I wanted to start with a simple baseline. Together with Michał Wojczulis I recently trained Polish GPT2 model — PapuGaPT2 (Wojczulis and Kłeczek 2021). I fine-tuned this model on the development set so that it learned the format of questions and answers and run inference on the test questions. This approach resulted in 18% accuracy on test A.

3. Retriever-reader approach

Even though the zero-shot approach started well, I didn't believe it is a promising path to achieve a good final score. The default approach for open domain question answering

[1] `http://poleval.pl/tasks/task4`

are systems based on retriever-reader architecture, and I chose to build it using Haystack framewerk by deepset.ai (Deepset 2021). In the first experiment, I used Polish Wikipedia chunked into 100-word long fragments as the knowledge base, BM25 as the retriever, and mBERT-base pretrained by Henryk Borzymowski on SQUAD data machine-translated into Polish (Borzymowski 2020). This experiment scored 16.64% accuracy on test A. After having the basic setup working, I run several experiments to see how far we can push the score.

3.1. Reader model

I got a significant gain from using XLM-Roberta trained on English SQUAD (Chan et al. 2021) as the reader model, which improved test A score by about 10 points vs mBERT trained on Polish SQUAD translation.

3.2. Retriever model

I experimented with training a DPR model (Karpukhin et al. 2020) to replace BM25 as the retriever. I didn't want to use translated data given my experience with readers, so I tried to use distant supervision using development set data, however this worked very poorly and I sticked to BM25.

3.3. Postprocessing

My reader model uses an extractive approach, and the task requires a very specific answer format. For that reason, I developed several postprocessing heuristics. I answer all 'yes/no' questions with 'yes'. I convert numbers in words into a numeric format. For questions that require a choice between several options, I try to answer them with the model, however if that fails (generated answer doesn't match any of the options) — I just pick one of those options.

3.4. Data-driven improvements

Some of the biggest gains I've had in the accuracy were driven by adding more data to the document store. Beyond the Polish Wikipedia dump, I added Polish word definitions from Polish Wordnet (Maziarz et al. 2016) and Polish proverbs. I thought that many questions sound like they might have been asked at school, and thought that websites providing help for students might be a good data source. I didn't want to crawl the internet, so instead I used the Polish subset of mc4 dataset (Xue et al. 2021) and filtered it for URLs containing one of the following strings: 'brainly', 'sciaga', 'bryk.pl', 'opracowania'. Unfortunately, this was a very long process and my colab instance could only run for 24 hours — pursuing this further might have provided additional gains.

4. Conclusions

My best score (46.44 test B accuracy) resulted in 4th place. I missed the podium, but learned a lot and I am looking forward to learn from the top 3 teams. It was an exciting task and I thank the organizers for preparing it.

References

Borzymowski H. (2020). *Multilingual BERT Fine-tuned on Polish SQuAD2.0*. `https://huggingface.co/henryk/bert-base-multilingual-cased-finetuned-polish-squad2`.

Chan B., Möller T., Pietsch M. and Soni T. (2021). *Multilingual XLM-RoBERTa large for QA on Various Languages*. `https://huggingface.co/deepset/xlm-roberta-large-squad2`.

Deepset (2021). *Haystack*. `https://github.com/deepset-ai/haystack`. GitHub repository.

Karpukhin V., Oguz B., Min S., Lewis P., Wu L., Edunov S., Chen D. and Yih W.-t. (2020). *Dense Passage Retrieval for Open-Domain Question Answering*. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP 2020)*, pp. 6769–6781, Online. Association for Computational Linguistics.

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

Wojczulis M. and Kłeczek D. (2021). *papuGaPT2 — Polish GPT2 Language Model*. `https://huggingface.co/flax-community/papuGaPT2`.

Xue L., Constant N., Roberts A., Kale M., Al-Rfou R., Siddhant A., Barua A. and Raffel C. (2021). *mT5: A Massively Multilingual Pre-trained Text-to-Text Transformer*. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 483–498, Online. Association for Computational Linguistics.