Preproceedings of the of the 18th Conference on Computer Science and Intelligence Systems pp. 1275–1280

Passage Retrieval in question answering systems in Polish language

Anna Pacanowska

***Abstract*—This paper describes the submissions to Task 3 of PolEval 2022. Passage Retrieval is a problem of retrieving a passage relevant to the given query. It is an important problem with many practical use cases, especially in question answering. It is very beneficial if a model is generalizable, that is effective in various domains, even the ones it was not trained on. This is a challenge for many state-of-the-art models. In this paper I describe and test many different methods of approaching this problem – from standard techniques, such as BM25 and lemmatization to recently developed methods based on deep learning and transformers.**

I. Introduction

THE aim of Task 3 of PolEval 2022 was to find a tool that retrieves the passages that contain the answer to the given question. The tool should work well not only on the domain it was trained on, but also other domains. The passages and queries are in Polish. I tested and compared different techniques and their combinations in order to find out which one will be the most effective – from the standard statistical approaches to advanced neural network based solutions. My goal was to find a solution that is not only effective, but also does not require great amount of resources to use – it should be memory and time efficient.

The best solution consisted of two stages. In the first I retrieved 1000 candidate passages using BM25 from Elasticsearch[1] on a corpus and queries lemmatized using Morfeusz2 [3]. In the second stage I calculated different scores and joined them using logistic regression. The scores were BM25 on lemmatized texts (with Elasticsearch), on unlemmatized texts and on bigrams. The rest of the scores were calculated using miniLM [13] on the texts translated with OPUS-MT [12]. They were computed on pairs (question, passage), (generated_answer, extracted_answer), (generated_answer question, extracted_answer passage). The answers were generated with GPT3 [1] and extracted with DistilBERT [10]. The code is available on GitHub[2].

II. Task

A. Description

The model's task was to retrieve 10 passages that were most relevant to the given question. To encourage generalizability the data was split into three distinct domains: Wikipedia passages, legal articles and Allegro FAQ. All training and development data came from the first domain and the other domains were present only in the test dataset.

[1]https://www.elastic.co/

[2]https://github.com/aniapacanowska/passage-retrieval

TABLE I
Dataset sizes and number of questions from each domain

domain	questions source	passages source	avg length	corpus size
wiki-trivia	Jeden z dziesięciu	Wikipedia	44	7097322
legal-questions	generated from passages	legal acts	153	26287
allegro-faq	FAQ	help articles	48	921

B. Domains

Each domain contains a separate corpus of passages and different kinds of questions. The domains are very diverse – the samples in each of them have unique characteristics that influence the solution. They vary in passage length, corpus size, question types, number of matches and other important details (Table I).

1) wiki-trivia: The questions come from "Jeden z dziesięciu" and are classical trivia questions. They have short, factoid answers and require only common knowledge. The passages are fragments of articles from Polish Wikipedia extracted using WikiExtractor. The passages are quite short, but the corpus is very large: it contains over 7 million passages. The questions were created first, and the passages were independently matched later. That means that the answer is usually worded in a different way. Sometimes it requires good understanding of the text to notice the passage does in fact contain the answer. Usually there are multiple relevant passages to each question.

2) legal-questions: The passages are Polish legal acts. The questions were artificially generated from passages by people – a volunteer first looked at the provision, and then generated a question for it. This means that, unlike in wiki-trivia, the answers are usually similarly worded and the passage contains a direct answer. The language is quite heavy and contains specialist vocabulary. Answering the questions requires in-depth knowledge of Polish law, not only common knowledge. The questions sometimes are ambiguous and make sense only in the context of the passage they were generated from, for example "Kto sprawuje nadzór nad Akademią?" (which Academy?). The passages are often very long, but the corpus is significantly smaller with about 26 thousand passages.

3) allegro-faq: The questions and passages are fragments of FAQ and help articles from Allegro. There are a lot of 'How to' questions. These questions are often open-ended, there are multiple possible responses that can be worded in many different ways. The answers are usually specific to Allegro. For most questions there is only one matching passage. This is the smallest dataset with only 921 passages.

Technical track: Challenges for Natural Language Processing (1st Symposium CNLPS 2023)

C. Evaluation

The submissions were evaluated using NDCG@10 with binary relevance scores. The overall score was a mean of the NDCG@10 score for each question in the test set. For the test-A dataset only the total score was visible – the correct answers were not public. BM25 was the baseline solution provided by the task's authors.

III. Methods

In this chapter I describe all the methods I used in the experiments.

A. BM25

In the first experiments I tried to improve the baseline solution by using lemmatization. Lemmatization can be used to improve the performance of BM25 – similar technique was used in the best solution to the Question Answering task in 2021 PolEval [7]. There are many different models capable of lemmatizating texts. I tried four different options – Morfeusz2 [3], spaCy [2], a hybrid of these two and modified Morfeusz2.

Morfeusz2 is a dictionary-based morphological analyzer for Polish language. SpaCy is an open-source library for natural language processing. It features much more complex techniques, such as state-of-the-art neural networks.

One of major distinctions between these models is how they handle ambiguity (words with multiple possible lemmas). Morfeusz2 is not capable of choosing the correct lemma based on the context. In the ambiguous cases it simply provides all possibilities. At first I tried to take all of the provided lemmas. This may be beneficial for two reasons. There is no risk I will lose a potential match because of choosing the wrong lemma. The other benefit is that words in the same inflected form always have identical set of lemmas, so they will often result in multiple matches. That means they will be more valuable than words in different grammatical form that have only one matching lemma. This way the words in different forms will still count as a match, but less than ones with exactly the same inflection. On the other hand, it will also result in many false matches of lemmas that are incorrect in the given context.

SpaCy on the other hand always tries to pick the correct lemma. However, it is sometimes wrong and picks the wrong one. It can also return words that are not valid lemmas or even do not exist in the Polish language. Morfeusz2 never makes up invalid words – it has a very large dictionary, but in case the model encounters an unknown word it is returned unchanged.

The hybrid approach tries to get the advantages of both models. It uses spaCy to pick the correct lemma from the ones provided by Morfeusz2. If the spaCy result was not returned by Morfeusz2, it takes the most popular lemma in the corpus. The popularity was measured on the corpus lemmatized using only Morfeusz2. Lastly, I wanted to check if spaCy is really useful in the hybrid model. Perhaps simply always choosing the most frequent lemma could work just as well.

B. Bigrams

I tested also a variation of the BM25 scoring function in which the terms are bigrams instead of words. This might be useful, because a bigram match is a stronger indicator that the passage is relevant than a single word match. This is not a sufficient method on its own, but can be useful in addition to other scores.

C. Deep learning

The next step was to use more advanced and recent models. Each of the following experiments consisted of two stages. In the first stage I retrieved top 100 passages with standard BM25 from Elasticsearch on texts lemmatized with Morfeusz2 lemmatizer (the basic approach). This was necessary, because many of these models require a lot of computing power, so it would not be feasible to run them on each possible pair. The second stage was re-ranking the retrieved passages using the selected method.

D. BERT

The first approach was to utilize BERT in a way that was outlined in [6]. A passage and a query separated with SEP token are encoded by a pre-trained model. The CLS vector is passed to a simple classification layer to predict whether the passage is relevant to the query. The model is fine-tuned on this task. I wanted to find out whether this approach would work well for the wiki domain and if it would be capable of generalizing to other domains. I used HerBERT [4] as the pre-trained Polish model.

E. Translation

Most state-of-the-art architectures for passage retrieval are very large and training them requires a lot of time and computing power. For this reason I decided it would be beneficial to use a pre-trained model. However, I could not find a good model for passage retrieval in Polish. There is much greater choice of models for passage retrieval in English. I used the OPUS-MT model [12] from Huggingface[3] in order to translate all of my data into English. I tested if translating the data and using more powerful models would improve the results.

F. miniLM

MiniLM [13] is a small model trained using knowledge distillation with BERT-base. One of its variants was fine-tuned on MSMARCO passage ranking dataset [5] with the ensemble of BERT-base, BERT-large and ALBERT-large as teacher models. This version achieved the best results for most IR tasks according to the BEIR paper [11], which is why miniLM was my first choice. An additional benefit is that small number of parameters makes the inference really efficient. MiniLM calculates a relevance score for a pair of query and passage.

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

TABLE II
LIST OF ALL SCORES USED (IN DIFFERENT COMBINATIONS) WITH LOGISTIC REGRESSION

Logistic regression features
BM25 (elasticsearch) for lemmatized texts
BM25 for unlemmatized texts
BM25 for bigrams (unlemmatized)
miniLM for (question, passage)
miniLM for (GPT3_answer question, DistillBERT_answer passage)
miniLM for (GPT3_answer, DistillBERT_answer)
miniLM for (chatGPT_answer question, DistillBERT_answerpassage)
miniLM for (chatGPT_answer, DistillBERT_answer)

G. Answer generation

There are two types of question answering models: extractive, which extract the answer from the given passage and generative, which generate the answer based only on the model's knowledge. Similarity between the answer generated by a generative model and the answer extracted from given passage could help miniLM decide whether the passage is relevant. If these answers are the same or similar it can be a good indicator that the passage is relevant. It can be misleading as well – the answer from either model may be incorrect or the same only by accident. I tested two variants. In the first I concatenated the generated and extracted answers to the beginning of the question and the passage, respectively. In the second I calculated the relevance score only between the answers.

As the extractive model I used DistilBERT [10], which is a distilled version of BERT. I tested two different generative models, both based on GPT: GPT-3 and chatGPT (based on GPT-3.5). These models are not open source – it was necessary to use the API provided by OpenAI to connect with them. Even if they were available I would not be able to run them locally as they are too large.

H. Ensamble

The last idea to improve the results was to gather the scores from multiple methods instead of relying on a single model. The test dataset is very different from the train set, so the function combining the scores had to be simple. I decided to use logistic regression.

IV. EXPERIMENTS

A. BM25 with lemmatization

These experiments were conducted with the use of Elasticsearch. I created and indexed a lemmatized corpus of passages with each lemmatizer (Morfeusz2, spaCy, hybrid and Morfeusz2-freq). Next I lemmatized the questions and retrieved the most relevant passages using Elasticsearch. The results on the dev dataset are shown in the Table III.

Lemmatizing the texts significantly improved the results. The impact considerably depended on the chosen model. The score with Morfeusz2 and spaCy was similar, with spaCy slightly higher. Morfeusz2 is even 20 times faster (on CPU) than spaCy, which makes it much easier to use. This is because Morfeusz2 is dictionary-based and spaCy is a complex neural network.

TABLE III
DIFFERENT LEMMATIZATION METHODS (DEV DATASET)

method	NDCG@10
no lemmatization	18.62
spaCy	21.41
Morfeusz2	21.15
hybrid	24.47
Morfeusz2-freq	**25.24**

The hybrid approach clearly outperformed both models. Morfeusz2 almost always provides the correct lemma, but usually together with a few incorrect ones. SpaCy on the other hand sometimes returns incorrect or even invalid lemmas. The hybrid approach eliminates both these issues – Morfeusz2 is used to check if the spaCy lemma is valid. If it is not, the most popular lemma gets chosen.

Surprisingly, the last technique turned out to be the best. The lemma was picked solely based on its frequency in the corpus lemmatized with Morfeusz2. That means spaCy was not necessary at all in the hybrid approach – the alternative method of choosing the correct lemma worked even better.

All methods have problems with proper names (such as surnames or places). There are too many of them for any model to know, so they are not properly lemmatized. For example, the lemmas from texts 'Bilbo Baggins' and 'Bilba Bagginsa' will be different.

B. Deep learning

In the following experiments I tested approaches based on deep learning and transformers. The passages to be re-ranked were fetched using Elasticsearch on a corpus lemmatized with Morfeusz2 (basic approach). I worked with the Huggingface library [14]. The submissions are collected in the Table IV. I did not calculate the results of all models on the dev dataset, because it would unnecessarily take a lot of time and resources. Test-B data appeared in the last two weeks, so I tried only the models that did well on test-A.

The submissions were evaluated on the PolEval website. The other scores in this section were calculated with my script. There are slight differences (around 0.5) that are a result of different behavior on questions where the correct passage is repeated.

In the following sections I describe in detail all of the methods I used.

C. BERT re-ranking

The first method to re-rank the passages was to fine-tune a BERT model (3 in Table IV). The input consisted of a question (sentence A) and a passage (sentence B). On top of BERT there was a simple classification layer (BertForSequenceClassification head), which was trained to predict whether the input pair is relevant based on the CLS vector. I fine-tuned the HerBERT-base model on the train dataset for one epoch. The model did better than the previous methods on the dev

TABLE IV
SUBMISSIONS SENT TO POLEVAL. SIZE: NUMBER OF PASSAGES TO RE-RANK, COMBINATION: METHOD OF JOINING THE SCORES FROM DIFFERENT MODELS, LR-DEV: LOGISTIC REGRESSION TRAINED ON THE DEV DATASET, LR-DEV: LOGISTIC REGRESSION TRAINED ON THE TRAIN DATASET

Methods				NDCG@10		
id	models	combination	size	dev	test-A	test-B
1	Baseline (from PolEval)	-	-	-	50.76	-
2	BM25 (k=1.0, b=0.5)	-	-	19.59	48.82	-
3	BERT	-	100	24.5	17.03	-
4	miniLM	-	100	31.36	58.19	-
5	miniLM	lr-train	100	31.97	59.76	-
6	miniLM	lr-dev	100	-	60.17	51.55
7	miniLM, GPT3	lr-dev	100	-	60.97	52.15
8	miniLM, GPT3, chatGPT	lr-dev	100	-	56.18	48.69
9	miniLM, GPT3	neural network	100	-	53.64	-
10	miniLM, GPT3	-	100	-	58.45	-
11	miniLM, GPT3	lr-dev	1000	-	**62.51**	**54.23**
12	miniLM, GPT3 (selected)	lr-dev	1000	-	-	54.15
13	miniLM, GPT3, chatGPT	lr-dev	1000	-	-	51.82
14	miniLM	lr-dev	1000	-	-	53.20

dataset, but the results on the test dataset dropped three times compared to baseline. The model learned well for the wiki domain, but was completely unprepared for the legal and allegro datasets. It would suggest that this technique of BERT re-ranking generalizes very poorly to other domains.

D. Translation

For the next experiments I had to translate the data to be able to use the models trained in English. The maximum length of the input for OPUS-MT model is 512 tokens. There are many longer passages in the datasets (especially in the legal corpus), so they had to be split. I tried to avoid splitting the sentences – passing only a fragment of a sentence might confuse the translator. However, I noticed that OPUS-MT does not work well for long texts – even if they fit in the limit. It often translates only some fragment of the input and leaves out the rest. This was not problematic for Wikipedia passages, which are usually short, but started to be noticeable for longer texts. So I split the passages into even smaller pieces (at most 0.3*512) and translated each separately. Then I joined all translated fragments.

E. miniLM

The first model I tested on the translated data was miniLM[4] with 6 layers fine-tuned on MSMARCO. Again, there were some problems with passages that were too long to fit into the model (the input can be at most 512 tokens). I split the passages into overlapping smaller pieces. Each fragment starts in the middle of the previous one – that way I want to avoid a situation where part of the answer is in one fragment and the rest is in the other. The fragments are at most 0.7*512 words to take into account that some words translate into multiple tokens. When ranking the passage, its score is the maximum of its fragments' scores. Re-ranking based on the scores computed by miniLM made a significant improvement over the baseline (4 in Table IV).

[4]https://huggingface.co/cross-encoder/ms-marco-MiniLM-L-6-v2

F. Logistic regression

I tried to join the results from the models I tested so far. This way I could aggregate the information stored in different scores. The features I considered were:

- BM25 (elasticsearch) for lemmatized texts
- BM25 for unlemmatized texts
- BM25 for bigrams (unlemmatized)
- miniLM score

I used the logistic regression model from sklearn library [8] and trained it on the train dataset (5). This resulted in a small, but noticeable improvement. Surprisingly, even larger improvement (on the test dataset) was achieved by training on the dev dataset instead (6). It might be because the train dataset is over 7 times larger and logistic regression started overfitting to the wiki-trivia domain.

In the next experiments I continued to use logistic regression and simply added new scores as additional features.

G. Answer generation

The next step was to use the question answering models. I generated an answer for each question using a generative model and extracted the answer from each (question, passage) pair using an extractive model. I did not want to exceed the free quota of the generative models, so the answers were generated only for the dev and test datasets. I was able to fit all my requests into the free trial. I calculated the miniLM score for each pair ('generated_answer question', 'extracted_answer passage') as well as just (generated answer, extracted answer). These results were then added as additional features to the logistic regression (trained on the dev dataset). As the extractive model I used DistilBERT[5] fine-tuned on the SQuAD dataset [9].

1) GPT3: The first generative model I tried was GPT3 – specifically gpt3-davinci, which is described in the documentation as the 'most capable'. I used the API provided by OpenAI. Each prompt was based on the question translated into English.

Normally, the answers given by the model are quite elaborate and if it does not have the requested information, the answer is 'unknown'. I wanted the answers to be concise and informative – even if the model had no information, I wanted it to guess the answer. I decided the short answers are better, because they do not contain unnecessary descriptions which could confuse miniLM. This also significantly reduced the cost of each query, because the responses are priced by the number of tokens. The guessed answers, even if incorrect, still might contain some useful information, such as the type of the answer (a number, a name etc.). I modified the prompts in order to get the desired results. To each question I added "Shortest answer. NOT unknown.". Temperature was set to a low value of 0.1. This way the model would usually get the true answer if it had enough knowledge, but was able to guess if necessary.

[5]https://huggingface.co/distilbert-base-cased-distilled-squad

Adding these answers to the dataset and calculating the scores as described above resulted in another small improvement (7 in Table IV). By inspecting the answers I noticed that they are usually correct for wiki-trivia questions, but often wrong for questions from other domains. This makes sense, because the wiki questions are based on common knowledge. Other domains require specific knowledge of Polish law or the policies of Allegro. The questions can be open-ended, which also makes it more difficult to guess the correct answer.

2) ChatGPT: I tried also the new chatGPT model. This time the questions could be in Polish as well as in English because this model is multilingual. I decided to use the Polish ones because chatGPT is likely better in translation than OPUS-MT that I used until now. However, it can have a downside – the translations might be inconsistent. The same expression in Polish could be translated differently in the answer by chatGPT than in the passage by OPUS-MT.

As before, I wanted to receive a concise answer in English. Interesting thing is that unlike GPT3, chatGPT almost never answers 'unknown'. It is guessing even without the suggestion in the prompt I had to add before. For the system prompt I left the default "You are a helpful assistant". To the questions in the legal domain I added "W Polsce" and in allegro domain "W allegro" in hopes it would help the model answer correctly. I finished each user prompt with "Shortest answer in english". Surprisingly, it turned out that this addition not only did not improve the results, but in fact made them much worse (8).

H. Combining scores

I compared a few different methods of combining the scores I gathered so far. The method should be quite simple to avoid overfitting to only wiki domain. I tried training a simple two layer neural network for classification. This turned out to be much worse than logistic regression (9). Ranking only by miniLM score with GPT3 answers also gave considerably worse results (10).

I. Bigger data

Sometimes the relevant passages are not included in the top 100 retrieved using BM25. The number of relevant passages found in the different positions in the ranking created using BM25 scores can be found in the Table V. The results were calculated using the training dataset and different lemmatizers.

The passages that fall outside of the top 100 will of course never be found by the re-ranker. On the other hand, adding a lot of new possible passages can result in false positives. I repeated some of the previous experiments on the top 1000 passages (retrieved by Elasticsearch with Morfeusz2, as before). In every case the outcome was considerably better. Re-ranking top 1000 passages with miniLM and GPT3 combined with logistic regression resulted in the best score I was able to achieve (11).

The main downside of this solution is a much larger computational cost. The most expensive task is translation, which was a problem especially for wiki domain that has the largest corpus. Another problem was that Elasticsearch sometimes retrieves less passages than requested when the corpus is small compared to the requested size. It is especially visible for allegro-faq, where there are only 921 passages, but Elasticsearch often retrieves even less. This happened for the top 100 as well, but rarely.

TABLE V
NUMBER OF RELEVANT PASSAGES IN DIFFERENT RANKING FRAGMENTS

lemmatization	:10	11:100	101:1000	1001:10000	10001:
none	2918	2755	2648	2078	4049
spaCy	3447	3476	2930	2023	2572
Morfeusz2	3385	3341	3044	2181	2497
hybrid	3952	3841	3064	1897	1694
Morfeusz2-freq	4073	3863	3091	1839	1582

I tried to simplify the re-ranking method and pass only the features that seemed most important: unlemmatized BM25 score, bigrams BM25 score, miniLM score and miniLM score on pairs concatenated with answers. This turned out to be only minimally worse than providing all scores (12).

J. Incorrect answers

In this section I analyze the incorrect answers given by the best model. The data is far too large to fully analyze it manually, so these are only some observations. The model always provides exactly 10 passages even though the number of correct ones is smaller. This means that retrieving a passage without the answer is not an error as long as it is lower in the ranking than the relevant passages. The incorrect passages are often relevant to the topic, but do not contain the answer to the given question. Sometimes the passage describes something similar, but not the same – for example the results for the question about the host of the show 'Zrób to sam' talk about the hosts of shows such as 'Sam tego nie rób'.

There are also some errors in the annotations. Some passages that were correctly retrieved by the model were not marked as relevant. There were also cases in which the annotated passage did not contain the answer. Other times it did have the answer, but without the necessary context. For example for the question "In which book Adam Mickiewicz describes Jankiel's concert?", one of the annotated passages says only that Jankiel is a character in "Pan Tadeusz" movie – does not mention the concert or the book's author. This does not happen only in the allegro domain because there the passages were verified manually. However, I think that despite these problems the datasets are still useful in measuring the performance of the models.

K. PolEval results

The final results are in the Table VI. My models clearly outperformed the baseline. The score of my best model is around the middle between the basic BM25 and the best solution of PolEval 2022.

After the contest ended, the answers for the test datasets were published. I compared the performance of my best model and the baseline on different domains (Table VII). It turned out that wiki domain was clearly the most difficult. The first reason

TABLE VI
FINAL RESULTS

model	test-A	test-B
baseline BM25 (Elasticsearch)	50.38	38.84
my best model	62.51	54.23
PolEval best solution	75.40	69.36

TABLE VII
SCORES ON DIFFERENT DOMAINS

	test-A			test-B		
model	wiki	legal	allegro	wiki	legal	allegro
baseline BM25 (Elasticsearch)	19.76	81.10	49.16	18.45	80.32	48.05
my best model	38.27	77.70	69.96	37.11	79.00	67.92

for that is a very large corpus (over 7 million passages). The other can be that the passages and the questions were created independently and matched later, so the answers are often indirect or differently worded. The score on allegro domain was much better, probably because of a very small number of possible passages (only 921). On both of these domains my model was much better than BM25. The legal domain was definitely the easiest. I believe it is because the questions were generated based on the passages, so the wording was usually very similar. Additionally, the questions often contained some unique words that pointed to a certain passage. It is the only domain where the new solution was slightly worse. The differences between the domains explain the lower score on test-B, where the majority of passages came from the wiki domain.

V. CONCLUSIONS

In this article I have explored various methods of passage retrieval, both traditional statistical approaches as well as recent models based on transformers.

The standard statistical methods, such as BM25 are a good starting point. They can be improved by means such as lemmatization. The choice of the lemmatizer largely impacts the performance. These methods don't have a problem with generalizability, because they are independent of the domain. Deep learning approaches are definitely more capable, but it comes at the cost of a much higher computational complexity. The most optimal way to use them is together with less accurate, but faster statistical methods. A good approach is to fetch a subset of passages with BM25 and then re-rank it with an advanced model, such as miniLM. Retrieving more passages to re-rank improves the results, but significantly increases the necessary resources, so it is important to find a balance. Knowledge distillation is an incredibly useful technique to reduce the computational cost of using a model. Translation is a good method when there are no models pre-trained in the correct language. It can be a great alternative to training a model from scratch, especially with limited resources.

Question answering is another good way to boost the results. Generative models are powerful, but expensive to use. However, they need to be used only once per question, unlike other methods that need to calculate a score for each pair of passage and query. It is important to choose the right model – GPT3 answers helped the re-ranking, but chatGPT did the opposite. Crafting a good prompt matters as well.

The best result was achieved by joining different techniques using logistic regression. Even a model that is less accurate on its own can still be useful to increase the score of a better model.

VI. NOTE

This paper is taken from my master's thesis written under the direction of dr Paweł Rychlikowski at the University of Wrocław.

REFERENCES

[1] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.

[2] Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. spaCy: Industrial-strength Natural Language Processing in Python. 2020.

[3] Witold Kieraś and Marcin Woliński. Morfeusz 2 – analizator i generator fleksyjny dla języka polskiego. *Język Polski*, XCVII(1):75–83, 2017.

[4] Robert Mroczkowski, Piotr Rybak, Alina Wróblewska, and Ireneusz Gawlik. HerBERT: Efficiently pretrained transformer-based language model for Polish. In *Proceedings of the 8th Workshop on Balto-Slavic Natural Language Processing*, pages 1–10, Kiyv, Ukraine, April 2021. Association for Computational Linguistics.

[5] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. MS MARCO: A human-generated MAchine reading COmprehension dataset, 2017.

[6] Rodrigo Frassetto Nogueira and Kyunghyun Cho. Passage re-ranking with BERT. *CoRR*, abs/1901.04085, 2019.

[7] Maciej Ogrodniczuk and Łukasz Kobyliński, editors. *Proceedings of the PolEval 2021 Workshop*, Warsaw, Poland, 2021. Institute of Computer Science, Polish Academy of Sciences. pg. 123-140.

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

[9] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics.

[10] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[11] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[12] Jörg Tiedemann and Santhosh Thottingal. OPUS-MT – building open translation services for the world. In *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation*, pages 479–480, Lisboa, Portugal, November 2020. European Association for Machine Translation. https://aclanthology.org/2020.eamt-1.61.

[13] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in Neural Information Processing Systems*, 33:5776–5788, 2020.

[14] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Perric Cistac, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing, 10 2020.