

Extracting text snippets

Yermalovich Dzmitry

May 2023

Abstract

This project is a solution to the problem of extracting the required piece of text that matches the query. This model works with Russian texts. The purpose of the study is to understand which approach to solving this problem is better than QA or NER. All code is open-source, here is the link to GitHub: https://github.com/YermalovichDzmitry/Text_Extraction_NLP_2023

1 Introduction

Often, when working with documents, a person has to retrieve: arbitration claims, government procurement, enforcement proceedings. This is required to complete the application form. But a person does not cope with this task quickly: he needs to read the text, analyze it, and select the desired fragment of the text. These actions take time. Therefore, a model was developed that will help the public procurement department extract the necessary piece of text from the document in order to generate an application form.

1.1 Team

This project was developed by Yermalovich Dzmitry. He was involved in data processing, data analysis, model building and error analysis.

2 Related Work

The task was reduced to two - NER and QA.

A very good approach to solving the NER problem was demonstrated in the article “Искусство распознавания: как мы разрабатывали прототип AutoML для задачи Named Entity Recognition”. This article was written in 2022. Here, the researchers very competently prepared the input data for the input of the neural network, namely, they developed an algorithm for splitting long input sequences. Bert models have a limit of 512 tokens. It may turn out that the length of the sentence exceeds the maximum length or punctuation marks are completely absent in the text. In this case, you need to look for alternative positions for splitting: `split_position_alt` is the position of the token closest to the border that does not start with “###”. This is done in order not to break the sequence in the middle of the word. If `split_position` was not found, `split_position_alt` is used [1].

In this work, their algorithm for splitting long sequences was used.

3 Model Description

This problem has been reduced to two: QA and NER. `sbert_large_nlu_ru` was chosen as a model, because it is suitable for Russian texts and nlu models are universal, that is, they can be used for QuestionAnswering, NER, classification, etc.

`sbert_large_nlu_ru` uses WordPiece tokenizer.

3.1 QA

Idea: the model is given a question and text as input, and it predicts the start and end of the answer.

Figure 1 shows the architecture of the Bert model that solves the QA problem.

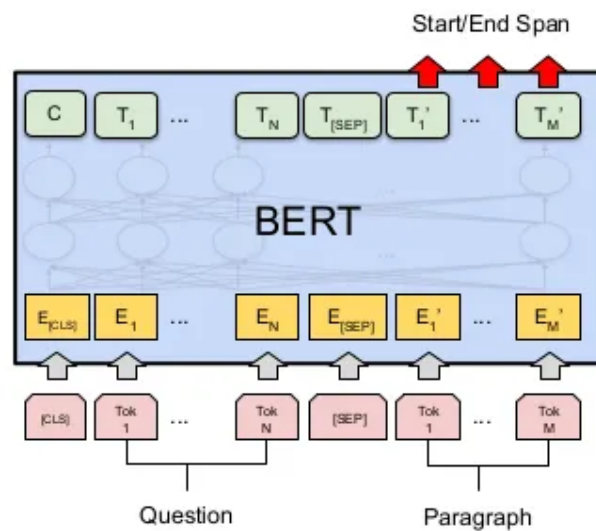


Figure 1 – Bert for QA

The training data was split into train/val/test, with a ratio of 0.8/0.1/0.1

I brought the input data to the structure:

input_ids - indexes of tokens in the dictionary.

attention_mask - which sets 1 where tokens are present, and 0 where they are not.

start_token - start token index already offset relative to the question.

end_token - end token index already offset relative to the question.

segment_ids - 0 - question, 1 – text.

shift - shift. Shift shows how many tokens the answer was shifted relative to the question.

3.2 NER

Idea: the model is given text as input, and it selects entities that are labels.

Figure 1 shows the architecture of the Bert model that solves the NER problem.

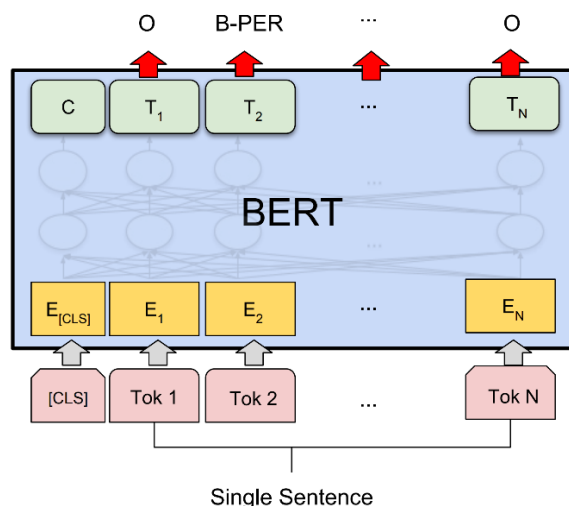


Figure 2 – Bert for NER

Entity dictionaries that I have defined.

label2id = {'[PAD]': 0, 'O': 1, 'B-EXE': 2, 'I-EXE': 3, 'B-GUA': 4, 'I-GUA': 5}

id2label = {0:'[PAD]', 1:'O', 2:'B-EXE', 3:'I-EXE', 4:'B-GUA', 5:'I-GUA'}

BIO concept was used. EXE – “обеспечение исполнения контракта”, GUA – “обеспечение гарантийных обязательств”

In this task, 'input_ids', 'attention_mask', 'labels' was selected. Also I solved the problem with long sequences. I took this decision from the article that I described above.

Preprocessing for NER

The indexes of those texts for which the query did not find an answer were highlighted. Such texts were excluded from the training sample, since the main goal of NER is to find the entities "обеспечение исполнения контракта" or "обеспечение гарантийных обязательств", and if it is not excluded, a request is made to "обеспечение исполнения контракта" and it will do nothing NER didn't find it, but in fact there is a “обеспечение гарантийных обязательств”, then this will confuse the model.

The training data was split into train/val/test, with a ratio of 0.8/0.1/0.1, relative to the skips already eliminated. But these gaps were added later to the test.

4 Dataset

The dataset was provided by Kontur. Given is a text, a request for it, the text of the response, and “answer_start” and “answer_end” of the response in the text.

Table 1 - Data type and number of examples.

Data type	Number of examples
Train dataset	1799
Test dataset	318

Test dataset was closed at the solution stage.

Also, no “answer_start” and “answer_end” were provided for the test data until the end of the competition, so train dataset was split into train/val/test. In this work, the test data on which “answer_start” and “answer_end” were not presented was called closed, and the test data that was formed for the test, open.

There are two types of requests: “обеспечение исполнения контракта” and “обеспечение гарантийных обязательств”.

Table 2 - Labels and their number.

Label name	Number of labels
Обеспечение исполнения контракта	988
Обеспечение гарантийных обязательств	811

The data is balanced by label.

Part of the observations is missing a text fragment to be extracted (an empty string inside the `extracted_part` field with `answer_start` and `answer_end` equal to zero). This means that the text of the document does not contain the required piece of text corresponding to the item in the questionnaire.

Number of empty fragments = 307.

Let's see how many passes there are for each of the labels.

Table 3 - Number of empty fragments per request type.

Label name	Number of passes
Обеспечение исполнения контракта	4
Обеспечение гарантийных обязательств	303

Most of the passes belong to “Обеспечение гарантийных обязательств”.

5 Experiments

5.1 Metrics

To evaluate the model, the `Accuracy` metric was used: the proportion of observations in which the text fragment extracted by the model fully corresponds to the fact.

5.2 Experiment Setup

5.2.1 Experiment Setup for QA Bert

Basic BertConfig from sbert_large_nlu_ru,
learning_rate = 5e-5
Epochs = 5, but after the second epoch the model started to retrain, the model was taken from the second epoch.
clip_grad_norm_c max_norm=max_grad_norm = 1.0
optimizer = AdamW

5.2.1 Experiment Setup for NER Bert

Basic BertConfig from sbert_large_nlu_ru,
learning_rate = 5e-5
Epochs = 7, but after the 5th epoch the model started to retrain, the model was taken from the fifth epoch.
clip_grad_norm_c max_norm=max_grad_norm = 1.0
optimizer = AdamW

5.3 Models Results and Error analysis

5.3.1 QA

After training the model, “accuracy” on the open test dataset was 0.77, and on the closed test dataset 0.79.

Table 4 – QA results.

Test data type	Model name	Tokenizer	Accuracy
Opened	sbert_large_nlu_ru	WordPiece	0.77
Closed	sbert_large_nlu_ru	WordPiece	0.79

After analyzing the errors, it was discovered that in most cases when the model made an error, it correctly identifies the beginning, but the end incorrectly: either it highlights a little more information, or a little less. Examples where the model made an error can be seen in the notebook on GitHub.

5.3.2 NER

Accuracy on all test data = 0.88.

The data on the test is unbalanced, that is, the model was trained without data with start = 0 and end = 0, so the data on which it was not trained was transferred to the test. And the algorithm determines well, start = 0 and end = 0, so the accuracy is very high.

The result is very good, but there are 307 empty ones in this test data, that is, start = 0 and end = 0. It is not clear how accurately the algorithm selects the text, where it actually is. Therefore, in the next test, empty examples were removed and accuracy was assessed using those examples that necessarily contain text.

Accuracy on test data without data with start = 0 and end = 0 = 0.75.

Accuracy on closed test data = 0.78.

Table 5 – NER results.

Test data type	Model name	Tokenizer	Accuracy
Opened without passes	sbert_large_nlu_ru	WordPiece	0.75
Opened with passes	sbert_large_nlu_ru	WordPiece	0.88
Closed	sbert_large_nlu_ru	WordPiece	0.78

Study of incorrectly highlighted sentences.

After analyzing the errors, it was found that in most cases in which the model made a mistake, it correctly determines the beginning, but the end is incorrect: either it highlights a little more information or a little less. Examples where the model made a mistake can be seen in the laptop.

6 Results

Table 6 – QA and NER results.

Test data type	Model name	Tokenizer	Accuracy
Opened without passes NER	sbert_large_nlu_ru	WordPiece	0.75
Opened with passes NER	sbert_large_nlu_ru	WordPiece	0.88
Closed NER	sbert_large_nlu_ru	WordPiece	0.78
Opened QA	sbert_large_nlu_ru	WordPiece	0.77
Closed QA	sbert_large_nlu_ru	WordPiece	0.79

Examining the Results of Two Models

After training the two models, their results were analyzed on a closed test dataset and it was found that 76% of the selected texts matched. This suggests that the models work well. It turned out that many examples in which the models give different results are very similar to each other, that is, they have the same beginning, but a different ending of the selected text. Also in these examples it is very difficult to isolate the end, because in most cases it ends with “___”, or “%”, or with a transcription of numbers. That is, there are such endings that it is not always obvious even to a person where the selected text ends.

7 Conclusion

As a result of the work, two models were implemented and trained, one is BertForQuestionAnswering and BertForTokenClassification. As a result of the study of the results, it was found that the percentage of coincidence between them is 76%. Examples where they have different answers, these answers are very similar, that is, many of them have the same beginning, but the end is slightly different. The result with BertForQuestionAnswering was chosen as predictions, because the results on the opened test dataset are higher, in the accuracy on the opened test dataset BertForQuestionAnswering = 0.77, and BertForTokenClassification = 0.75. Also, the results on the closed test dataset turned out to be higher for BertForQuestionAnswering, BertForQuestionAnswering accuracy = 0.79, BertForTokenClassification accuracy = 0.78.

References

1 Искусство распознавания: как мы разрабатывали прототип AutoML для задачи Named Entity Recognition [Electronic resource]. – Access mode : <https://habr.com/ru/companies/vtb/articles/651525/>.