Can I start? Ok

Well. Let's start. My name is Dmitry Pogrebnoy and I would like to present a work titled "RuMedSpellchecker: correcting spelling errors for natural Russian language in electronic health records using machine learning techniques".

There are many predictive and decision-making models in health care. Such models are often based on electronic texts of patients' medical records. The quality of such models strongly depends on the quality of the original medical records. Electronic health records are usually presented as a plain text. Such text is mostly unstructured, hand-typed, and free of any markup. And of course such text may contain a lot of typos and spelling errors.

Toutanova and others in their study highlighted two main causes of spelling errors. The first reason relates to the writer itself, who may not know how to spell a word correctly or make a typo in a hurry. The second reason relates to problems with typing devices, which can stick or skip presses. And because of these specifics, we get electronic health records with many spelling errors. At the same time, spelling errors in such texts greatly reduce the quality of the final medical models. Therefore a high-quality tool for automatic correction of spelling errors will be able to fix this problem and increase the quality of the models without additional costs.

That's why the purpose of this work is to design a method and implement a tool for automatic correction of spelling errors for the Russian medical texts.

The developed tool should accept raw plain text and return a corrected text with minimal number of spelling errors. The tool should also be able to use the GPU to speed up text processing. In addition, the developed tool should be at least comparable in performance with existing tools. And besides it should outperform the existing open source tools in the quality of correcting spelling errors in Russian medical texts.

Well, now that the goal and all the requirements are specified, so let's move on to the internals.

First of all, it is necessary to highlight that this work only considers spelling errors of six types. Examples of these errors are shown on the screen. The first 4 types of errors are related to letters and the last two types of errors are related to spaces. We decided to move step by step. And in this work, we considered only such spelling errors and didn’t take into account other types of errors, such as punctuation or semantic mistakes.

Now let's look at existing tools.

There are several tools for correcting spelling errors in Russian. However, none of them is intended to correct errors in medical texts. And besides, none of them uses advanced language models to improve the quality of corrections. So this work fills this gap.

Let's take a look at the spelling correction process. First, the medical text is splitted into tokens. Then, for each token, it is checked whether it is suitable for correction. If the token is suitable for correction it is checked whether it is correct or not. If the token is incorrect, then a list of candidates is generated from the prepared index and the most suitable candidate is selected. The corrected result is included into the final text. After all tokens are processed, they are assembled into a single text and the finished result is returned from the tool.

Let's look at the architecture. The architecture of the tool consists of seven components. The main component is the Spellchecker Manager, which is responsible for coordinating other components and implementing high-level logic. The Preprocessor and PostProcessor components are responsible for splitting the incoming text and assembling the result. The Dictionary component contains a dictionary of all correct words. Error model is responsible for generating candidates for fixing incorrect words. The Edit Distance Index component optimizes and speeds up the generation of editing candidates. And the last component is a Language Model that ranks candidates for correction and selects the most suitable word for correction.

In order to build the language model for the new tool we needed to collect a sufficient dataset. So we collected and prepared a dataset with Russian medical texts. We used two public datasets and two private ones. The public datasets are RuMedNLI and RuMedPrimeData, and the private ones are the dataset from the Almazov Medical Center and the dataset from the Russian Academy of Sciences.

All four datasets were pre-processed and combined into one final dataset, which contained over thirty thousand records and took just over 10 megabytes. It's not much, but that's all that was available at the time. Unfortunately for such unpopular languages as Russian there are still no big enough open specific corpora.

So the collected dataset was used to fine-tune the language models. We selected three basic BERT models of different sizes and fine-tuned them to rank candidates. The fill mask task was used for fine-tuning. The fine-tuning parameters of each model were standard, and there were 5 training epochs for each model. In addition, before fine-tuning, the DistilBert model was converted from multilingual to monolingual model for Russian language. So as a result, all three fine-tuned models were published on the Hugging Face service.

The architecture of the developed tool allows to extend the tool and adapt other models for use in the tool as a language model. For this purpose, it is only necessary to implement the special interface and that's all. So as an example, two models RuBioBERT and RuRoBERTa were adapted. These models were obtained in an article by Yalunin et al. from Sberbank AI Lab and were also fine-tuned on Russian medical data, but on a different dataset.

The developed tool is intended for use only for medical texts in Russian. Preferably for anamnesis and medical histories of patients. The tool should be used in the text preprocessing pipeline before any preprocessing steps. It is also advisable to make sure that everything is in order after applying the tool.

And here is an example of a correction. The original form is at the top of the table, and at the bottom is the result of the new tool with the MedDistilBERT model. As you can see, existing tools have problems even with this simple example. Only the new tool produced the correct result, and the other tools didn't fix anything or messed something up.

However, it is impossible to conclude the quality of the corrections of the new tool from a few examples.

So let's look at the tests. We did three tests. One test with single incorrect words, another test with contexts around the incorrect words, and the last test with real anamnesis.

For the first two tests, we used error precision metric, which shows how well the tool corrects incorrect words, and lexical precision metric, which shows how well the tool doesn't mess up correct words. We also used the average of these precision metrics and performance metric, which is the average number of words processed per second. And for the last test with real anamneses, we used the number of correct corrections, the number of unnecessary corrections, and the ratio of these two metrics.

Let's take a look at the test results. This table shows the results of the test for correcting a single word by several popular open source tools. The results of the developed tool are presented at the bottom of the table in the green section. As you can see, the new tool shows an average result in error precision and rather low performance, but it achieves the highest lexical precision. However, the new tool uses a language model that takes into account the context around the incorrect word, so it can work much better with words with context.

And the next table shows the results of the test for words with context. In this test, the new tool reveals itself and outperforms other tools in terms of precision metrics. Despite the high precision scores, the new tool has average performance and doesn't fall much in this metric relative to competitors.

In the test with real anamneses, the new tool also showed a best result in the ratio of correct to unnecessary corrections. Here, dashes mean that after applying the tool, the original text changed so much that it was impossible to count the metrics. And It is also worth noting that LanguageTool made the greatest number of correct corrections, which is two more than the best result of the new tool. So it’s not so many, but still.

Now a few words about performance. The tool is written in Python and this affects performance. Several methods are used to achieve sufficient performance. One of them is the Dumerau-Levenshtein edit distance that is used to limit the number of edit candidates. Calculating the edit distance is a frequent and computationally expensive operation. Therefore a special index is used to optimize this calculation. It stores the pre-calculated part and thus speeds up the computation in runtime. In addition, all performance-critical operations are delegated to high-performance libraries. For example, the editdistpy package is used to calculate the edit distance, and the transformers and accelerate packages are used for model inference, allowing to use suitable GPU for model inference.

So let’s go next. We assembled the developed tool into a pip package and published a Beta version of the tool. This package contains the source code and necessary classes, as well as a dictionary of correct words. The package doesn’t contain models and they are downloaded automatically when needed.

So let 's move on to the conclusion

In this work we presented a new method for correcting medical texts in Russian. In addition, we presented a tool that implements the proposed method. We conducted several tests. And the results showed that the new tool is comparable in performance with existing tools and outperforms them in correction precision. The resulting tool is open-sourced and published as a separate pip package. The fine-tuned models are published on the HuggingFace service.

Well, that's it from me. Here are links to project parts. And I am ready to answer your questions.