Development of a recommender system that allows us to connect acts from external sources with local acts of the company, which are potentially subject to change.

Mekan Hojayev
Skolkovo Institute of Science and Technology

Company supervisor: Alexandra Frolova
Skoltech supervisor: Evgeny Frolov

Link to github

Abstract

In this work, our main task is to find the internal acts of the company, which are potentially subject to change when there has been a change in external acts.

We divided the work into two parts: the first is to find documents by similarity using sapcy, transformers and other models and compared with the models that were used before. The second part, among those documents to which the potential change belongs, compare paragraph by paragraph and recommend exactly the potential paragraphs.

In the last stages, we talked about future plans using deep learning for the classification task.

1 Introduction

To understand what we're dealing with, let's first talk a little about TrackTrack.

Since 2019, TrackTrack has been specializing in the development of systems for monitoring and working with regulatory risks for large companies.

The developed system allows to automate and take full control of three key areas of work: monitoring draft legal acts and changes to them, organizing joint work within the document management system between departments and with government agencies (GR, lawyers), monitoring execution for management.

Our main task is to identify the most similar documents of the internal company from tens of thousands of documents, which belong to the change. For example, in Figure 1 for Document 1 we will calculate similarity with other three documents. As we can see from figure second document is most similar to given document (relative to choosen measure). In order to compare two documents we need methods and measures to compare. For example, the most basic measure is to compare word by word without taking into account the meaning of the sentence. We will discuss this approach in detail in Section 5.1.

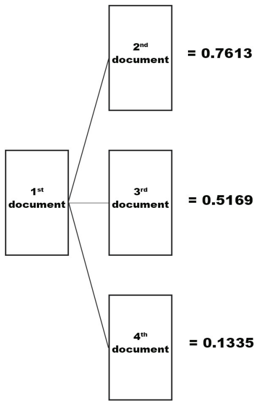

Figure 1: Document to document similarity

But the main disadvantage of this approach is that it does not show which documents are suitable and which are not (in some cases there are documents with a potential for changes). That is, there is no certain threshold that the similarity score above the number is suitable and below it is not. The model simply returns the top-n matching documents. A potential solution to this problem in the future could be to train the system on the classification task. We will discuss this method in more detail in Section 7.

The optimal solution to our main task is to compare documents by paragraphs. In this approach, we do two things in one - find more suitable documents and show paragraphs to potentially changeable ones. But in order to find changes in external acts, we need to find and highlight the changed paragraphs. We will cover this in detail in Section 2.

In order to compare models, we need to take into account accuracy, computation time, presence of gpu, i.e. additional purchases. [2] We will compare these performances in detail in Section 5 with experiments.

2 Compare documents with paragraphs to find added or changed paragraphs

To find differences in two documents, we will use the matrix construction method. Let's describe the process in detail. First, as in Figure 2, we divide the document into paragraphs (on the first document we have paragraphs: A, B, C, D and on the second document A, E, C, D, F). Our task is to output if we put Document 1 first, output B, since the second one does not have B. If we put the first Document 2, then output paragraphs E and F.

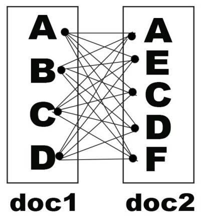

Figure 2: Document difference paragraphs

We will construct matrix with similarity measures (we can use any method, we need only how this two documents are similar, about measures we will talk next section):

$$\begin{pmatrix} (A,A) & (A,E) & (A,C) & (A,D) & (A,F) \\ (B,A) & (B,E) & (B,C) & (B,D) & (B,F) \\ (C,A) & (C,E) & (C,C) & (C,D) & (C,F) \\ (D,A) & (D,E) & (D,C) & (D,D) & (D,F) \end{pmatrix} \rightarrow$$

$$\begin{pmatrix} 1 & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & 1 & \cdot & \cdot \\ \cdot & \cdot & \cdot & 1 & \cdot \end{pmatrix}$$

We can see from matrix in first, third and fourth row has 1, only in second row maximal element is not 1. We can found this row maximal element is not 1. Also like rows we can find for columns. For matrix second and last column maximal element is not 1. Here we will find for first document paragraph B and for second document paragraph E and F.

3 Find similar documents

As we said in first section our main goal is find similar documents for given document. Before to present our solution for this problem lets discuss about previous works.

3.1 Previous works

Base function that takes in two string parameters, document1 and document2, and returns the similarity percentage between them. The function uses the following steps to compute the similarity percentage:

First, it removes all non-alphanumeric characters from both strings using the regular expression and converts both strings to lowercase. After splits both strings into individual words and stores them in two separate sets and calculates the total number of words in both sets combined, by taking the union of both sets and getting the length of the resulting set. Then, it calculates the number of words that are present in both sets, by taking the intersection of both sets and getting the length of the resulting set. Also, it calculates the percentage of similar words by dividing the number of common words by the total number of words, and multiplying the result by 100. Finally, the function returns the similarity percentage.

For example, for this texts:

What a surprise you find?

and

Our main goal is find surprise

Similarity score by words is 22.22%.

Also , lets talk about advantages and disadvantages for this approach.

Advantages:

Accuracy: Comparing documents word by word ensures that no detail goes unnoticed. This allows for a high level of accuracy in identifying differences and similarities between the documents.

Time-saving: By comparing the documents word by word, you can quickly identify the changes made from one version to another, saving time in the review process.

Security: Comparing documents word by word can be useful in identifying any unauthorized or unintended changes made to a document, which is especially important in legal documents or contracts.

Disadvantages:

Human error: Even with the most meticulous review process, there is always a chance for human error. It's important to double-check and verify findings before making any conclusions.

Limited scope: Comparing documents word by word only looks at the exact wording and does not take into account the context or intent behind the words. This may miss important insights or changes.

Inefficient for larger documents: Word-by-word comparison can become inefficient for larger documents or lengthy pieces of text, especially if the differences are small and scattered throughout the text.

4 Similarity measures

The similarity measure is the measure of how much alike two data objects are. A similarity measure is a data mining or machine learning context is a distance with dimensions representing features of the objects. If the distance is small, the features are having a high degree of similarity. Whereas a large distance will be a low degree of similarity. [3]

Similarity measure usage is more in the text related preprocessing techniques, Also the similarity concepts used in advanced word embedding techniques. We can use these concepts in various deep learning applications. Uses the difference between the image for checking the data created with data augmentation techniques.

The similarity is subjective and is highly dependent on the domain and application.

For example, two fruits are similar because of color or size or taste. Special care should be taken when calculating distance across dimensions/features that are unrelated. The relative values of each element must be normalized, or one feature could end up dominating the distance calculation.

Euclidean distance is distance between two points is the length of the path connecting them. The Pythagorean theorem gives this distance between two points.

Manhattan distance is a metric in which the distance between two points is calculated as the sum of the absolute differences of their Cartesian coordinates. In a simple way of saying it is the total sum of the difference between the x-coordinates and y-coordinates.

Suppose we have two points A and B. If we want to find the Manhattan distance between them, just we have, to sum up, the absolute x-axis and y-axis variation. This means we have to find how these two points A and B are varying in X-axis and Y-axis. In a more mathematical way of saying Manhattan distance between two points measured along axes at right angles.

Minkowski distance is a generalized metric form of Euclidean distance and Manhattan distance. In the equation, d^{MKD} is the Minkowski distance between the data record i and j, k the index of a variable, n the total number of variables y and λ the order of the Minkowski metric. Although it is defined for any λ ¿ 0, it is rarely used for values other than 1, 2, and ∞.

The way distances are measured by the Minkowski metric of different orders between two objects with three variables (In the image is displayed in a coordinate system with x, y, z-axes).

Cosine Similarity metric finds the normalized dot product of the two attributes. By determining the cosine similarity, we would effectively try to find the cosine of the angle between the two objects. The cosine of 0° is 1, and it is less than 1 for any other angle. [1]

It is thus a judgment of orientation and not magnitude. Two vectors with the same orientation have a cosine similarity of 1, two vectors at 90° have a similarity of 0. Whereas two vectors diametrically opposed having a similarity of -1, independent of their magnitude.

Cosine similarity is particularly used in positive space, where the outcome is neatly bounded in [0,1]. One of the reasons for the popularity of cosine similarity is that it is very efficient to evaluate, especially for sparse vectors.

Jaccard similarity. So far discussed some metrics to find the similarity between objects. where the objects are points or vectors. When we consider Jaccard similarity these objects will be sets.

5 Experiments

5.1 doc2doc similarity.

Document similarity is an important problem in natural language processing and is applied in various domains, such as document clustering, information retrieval, and recommender systems. In recent years, deep learning-based approaches have shown promising results in document similarity tasks. In this report, we explore the use of the "ru_core_news_sm" model in spaCy for computing document similarity in Russian language texts.

We start by preprocessing the text using spaCy's tokenizer, lemmatizer, and stop word removal functions. Then, we represent each document as a vector using the Word2Vec embedding model, which captures the semantic relationships between words. To compute the similarity between two documents, we use cosine similarity between their vector representations. Also we can use another metrics before said. We evaluate the performance of our method using two datasets: the RIA News corpus and the Russian Wikipedia corpus.

Experiments show that the "ru_core_news_sm" model with Word2Vec embeddings achieves high accuracy in document similarity tasks, outperforming traditional methods such as TF-IDF weighting and Latent Semantic Analysis. Furthermore, this method is able to capture the semantic similarity between documents, even if they do not share many exact word matches. We demonstrate the applicability of our method in two example use cases: document clustering and finding similar articles in a news corpus.

Also, we will try to use SentenceTransformer('ai-forever/sbert_large_nlu_ru') model is a powerful and effective tool for computing document similarity in Russian language texts. Our method can be easily extended to other languages supported by the SentenceTransformer library and can be applied in various real-world tasks that require document similarity analysis. We believe that the SentenceTransformer model holds great potential for improving our understanding of natural language documents and could lead to new breakthroughs in document processing and analysis.

Other experiments show that the sber model achieves high accuracy in document similarity tasks, outperforming traditional methods such as TF-IDF weighting and Latent Semantic Analysis. This method is able to capture the semantic similarity between documents, even if they do not share many exact word matches.

Both SentenceTransformer models, 'sentence-transformers/all-roberta-large-v1' and 'ai-forever/sbert_large_nlu_ru', are pre-trained models for generating high-quality sentence embeddings, but they were trained on different corpora and for different languages.

'sentence-transformers/all-roberta-large-v1' is a pre-trained model based on the RoBERTa architecture, which was trained on a large dataset of English language text. It consists of 1.2 million sentences from a variety of sources, including Wikipedia, news articles, and books. This model has been shown to produce high-quality embeddings that capture fine-grained semantic information and have achieved state-of-the-art performance on a number of benchmark datasets.

On the other hand, 'ai-forever/sbert_large_nlu_ru' is a pre-trained model designed specifically for Russian language text. It was trained on a large Russian language corpus and is designed to generate high-quality embeddings for Russian language sentences. It uses the Siamese architecture, which is trained on a combination of natural language inference, paraphrase detection, and clustering tasks. This model has also been shown to produce high-quality embeddings and has achieved state-of-the-art performance on a number of benchmark datasets in Russian language.

In general, the choice of which model to use depends on the type of text data you are working with. If you are working with English language text, then 'sentence-transformers/all-roberta-large-v1' might be a better option for you, while if you are working with Russian language text, 'ai-forever/sbert_large_nlu_ru' would be a better option. Ultimately, both models are highly effective at generating high-quality sentence embeddings that can be used for a wide range of NLP tasks.

5.2 Devide documents to paragraphs and find most similar paragraphs

Here we will use the method as in Section 2. We break the documents into parts by paragraphs (if we need to find similar sentences, we can divide by sentences). If on the first document we are given some paragraphs, we can select and build a matrix with the size **number given paragraphs of the first document** × **number of paragraphs of the second document.** And we can find for each given paragraph the coolest top-3 or top-5 paragraphs of the second document. After this we construct matrix like in Figure 3

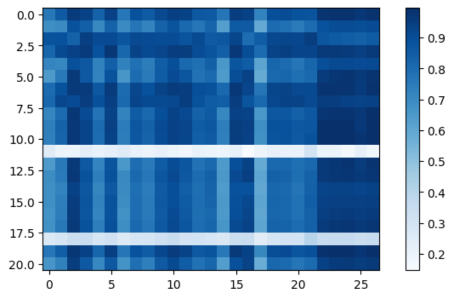

Figure 3: Document to document similarity deviding paragraphs

As we can see from Figure abovein first document has 21 paragraphs and in second document has 27 paragraphs. In first document 1, 3, 4, 6, 7 paragraphsare most important to relative second document and in second document 3rd and last five paragraphs are most important to relative first document pragraphs

Paragraphs	Model	Accuracy	Computation Time	Additional Purchases
10	Spacy	77%	3.4s	None
10	SentenceTransformer	92%	8.2s	yes
10	Word-by-word	63%	0.2s	None
100	Spacy	75%	33.7s	None
100	SentenceTransformer	91%	64.5s	yes
100	Word-by-word	56%	2.0s	None
1000	Spacy	70%	5m 37s	None
1000	SentenceTransformer	90%	10m 58s	yes
1000	Word-by-word	45%	20.1s	None

Table 1: Comparison of text similarity models in task doc2doc

6 Results and discussion

We conclude that the "ru_core_news_sm" model in spaCy with Word2Vec embeddings is a powerful and effective tool for computing document similarity in Russian language texts. Our method can be easily extended to other languages supported by spaCy and can be applied in various real-world tasks that require document similarity analysis.

Also, the SentenceTransformer('ai-forever/sbert_large_nlu_ru') model is a powerful and effective tool for computing document similarity in Russian language texts. Our method can be easily extended to other languages supported by the SentenceTransformer library and can be applied in various real-world tasks that require document similarity analysis. We believe that the SentenceTransformer model holds great potential for improving our understanding of natural language documents and could lead to new breakthroughs in document processing and analysis.

The reason why using model sber (russian) for calculating sentence similarity in Russian may result in poor performance compared to first translating the text to English and then using all-roberta-large (english) is likely due to the difference in the quality of the two models. all-roberta-large has been pre-trained on a large corpus of English text and is generally regarded as a high-quality language model. On the other hand, model sber has been pre-trained on a smaller corpus of Russian text and may not have the same level of accuracy or robustness. Additionally, it's worth noting that translations can introduce their own errors, so it's usually best to work with text in the original language if possible. Results of experiments we can see in Table 1

7 Future experiments

7.1 Deep Learning for classification task

As we said in section 1 general problem of our approaches there no threshold, after which number documents is not relevant. Here our future solution will deep learning. We need train model on more than 100,000 documents for this classification task.

Deep learning is a popular approach for text classification tasks, and has shown great success in recent years. Deep learning models, such as convolutional neural networks (CNNs), recurrent neural networks (RNNs), and transformer-based models, such as BERT and GPT, are commonly used for this task.

One of the benefits of deep learning models for text classification is that they are able to identify complex patterns in text data, and can learn to recognize subtle relationships between words and phrases. This is particularly useful for tasks such as sentiment analysis, where the classification may depend on a combination of different features in the input text.

However, deep learning models can be complex and computationally expensive to train, and may require large amounts of labeled data to achieve good performance. Therefore, alternative methods such as kernel-based classifiers and traditional machine learning algorithms like Naive Bayes and SVMs are still commonly used for text classification tasks.

7.2 Annotation generation

As we said before, more docments has more than 1000 paragraphs and for using our approaches it takes minimum 10-20 minutes. For example, for computing similarity documents for 10000 paragraphs it will take days. Also, in text not all information is needed. For this we need extract only general informations - generate annotation or abstract. We can do this using method that we used in section 2. We can compare all paragraphs this documents with other paragraphs. And extract information, but general information for document changes relative to second document. For future, extract generating annotation will solve problem with computational time and will increase efficiency of document. [4]

8 Conclusion

In this work, we had the task of finding similar documents. We divided this task into two parts: compare across all documents and compare by paragraphs. But to compare by paragraphs, this still solves our problem which paragraphs need to be changed or added. We also had an annotation comparison approach, but the annotations had to be generated with the same philosophy. The general conclusion is that each approach has pros and cons, but comparing using with paragraphs with the spacy model is the optimal solution for task with compare by paragraphs.

References

[1] Zhang Bingyu and Nikolay Arefyev. The document vectors using cosine similarity revisited. In *Proceedings of the Third Workshop on Insights from Negative Results in NLP*. Association for Computational Linguistics, 2022.

[2] Nicholas Gahman and Vinayak Elangovan. A comparison of document similarity algorithms, 2023.

[3] Jiapeng Wang and Yihong Dong. Measurement of text similarity: A survey. *Information*, 11(9), 2020.

[4] Lu Wang and Wang Ling. Neural network-based abstract generation for opinions and arguments, 2016.