ESILV
ENGINEERING SCHOOL
DE VINCI PARIS

Project Report
Machine Learning for NLP

Alexandre Cogordan
Alexandre Brosseau

Table of contents

Link to the code: *https://bit.ly/projet-nlp*

Our analysis

Firstly, we started by analysing the corpus you gave to us. We took a look at what queries and documents looked like to get a better idea of how to proceed.

Then, we analysed the basic code. We figured out that it was a bit hard to understand it as it was made out of one block. Therefore, we decided to divide the code into many functions, such as clean_sentence or generate_corpus functions, each with its own purpose. Furthermore, we modified those functions a bit to make the code clearer and somewhat faster. This will enable us to better understand the functioning and impact of each part of our code.

Once this code clarification stage had been completed, we could move on to the development part of our solution.

Creating our solution

Pre-Processing

Firstly, we realised that the pre-processing stage was rather succinct. So we decided to go further on that step by implementing the clean_sentence function, which is integral to the preprocessing stage.

Removal of Noise: It eliminates common stopwords, URLs, and numbers. This step is crucial for reducing noise in the text, ensuring that our analysis focuses on the most meaningful parts of the content.

Standardisation of Text: The function converts all text to lowercase and then tokenizes it into individual words. Lowercasing ensures uniformity, aiding in the accurate comparison of words, while tokenization breaks down the text into manageable and analyzable units.

Lemmatization: Words are converted to their base or lemma form. This process is essential for treating different forms of a word as a single entity, which is particularly useful in understanding the overall context and meaning of the text.

Filtering Based on Word Length: The function also filters out words shorter than a specified length, focusing on more significant words and discarding potential noise or irrelevant tokens.

Models

We decided to keep the original BM25 model in order to see how far we could improve its results. But that's not all, we've also decided to use BM25 in combination with other models such as word2vec or its derivatives. We searched for any pre-trained word2vec vectors, specific to the medical field, on the internet. We found an alternative to word2vec, which is glove. We also found BioWordVec and Wikipedia-Pubmed, which are two sets of vectors specially trained on medical publications.

We should bear in my mind that any pre-processing used in our case has only been influencing our own word2vec model, That is explained by the forbidden use of using models as without them, we cannot use our corpus but rather the one that has been trained prior to their vectorisation. Therefore, the inability to exert direct modifications to the imported vectors is something to keep in mind.

In order to see the influence of each of these solutions, we implemented a function that computes a score based on the results of each of the solutions (combine_score function). Therefore, we can weigh the value of techniques and modify them as we want, in order to optimise our results. This function takes an alpha parameter, which is different for each of the techniques. So we were able to make the most of each of the models.

To further explain this concept, here is the formula we've have devised:

$$\left(\alpha \times bm_{score}\right) + (1 - \alpha) \times \sum_{i=1}^{number\ of\ models} \left(model.coef_{i} * model.values_{i}\right)$$

This way, we manage to give a weighted average value using an alpha coefficient for the original bm score and give each other model a summarised corresponding coefficient.

These scores are given to the ncdg score function to be evaluated with the trueDocs values.

Furthermore, to be able to compute this first hand and have coherent calculations, we normalised all values. This can be seen in the function ‘normalise’.

Evaluation of our factors using a graph

We managed to highlight a number of factors that had a high impact on the final scores. This allowed us to change the range of their values in order to get the optimal score. Amongst these factors, we have:

- **Keeping the relevant documents**: This decides whether to only keep the queries that are highly relevant to the documents or not (queries that have a relevancy score of 2 in the dev.2-1-0.qrel document).
- **Our model’s vector size**: This represents the dimensionality of continuous vector representations assigned to words during training. Hence, the higher the vector size, the more intricate relationships we gather but also the more the computational resources are used.
- **Our model’s vector window**:
- **Our model’s minimum word count**: Refer to “*Filtering Based on Word Length*”.
- **Our alpha value:** We use this value to create a weighted average between from the BM25 and the cosine similarity scores from all of our models using a weighted average.
- **The choice between using cbow (sg = 0) or skip-gram (sg = 1)**

Please find the tests can be found in the ‘Further evaluations’ section of our code.

Bear in mind that some of these evaluations have been evaluated on different scores. For example, the vector size was only evaluated on the word2vec as it only affects it and not the other models nor the bm25 score.

Chosen factors

Find below, the corresponding chosen values for each factor after evaluation:

- **Keeping the relevant documents**: False

- **Our model's vector size**: 300
- **Our model's vector window**: 50
- **Our model's minimum word count**: 1
- **Our alpha value:** 1
- **The choice between using cbow (sg = 0) or skip-gram (sg = 1):** 0

Challenges & Explorations

Challenges

We've had a number of unsuccessful attempts to try to better our scores. Although not exhaustive, we believe that some of them needed to be discussed and could be subject to our suppositions on how they made the scores worse or the solution slower.

After finding the models that we wanted, we tried to improve them, by having a better pre-processing for example. Likewise, we found out that we could use N-grams in the pre-processing, which are combinations of n adjacent words in the text. For example, in the sentence "I love Python programming", some 2-grams would be "I love" and "Python programming". However, using that, we got a result of 0.79 with these combinations, which was worse than with our previous pre-processing, so we removed it.

Also, as we saw during the course, we can think that the first words in a document are the most important. Thus, we decided to only take the first 150 words of each document . By doing that, our score dropped by 6 points, to 0.85, which led us to believe that the first words of the documents weren't the most important in our medical corpus.

In another context, we tested the summarization of the documents, using a model specialised in medical summarization. However,this process was too resource-intensive and took hours to run. We therefore decided to abandon the idea of using summarization. Maybe, if we had more compute resources, we could have followed this path.

Explorations

We could also have used other models, such as Biobert or ClinicalBERT to improve our results, but we didn't have the time to implement it. As an extension of this, we could have tried to normalise dates, find synonyms of words or find a model that could handle medical acronyms.

Our result

As a final result, based on the 150 first documents of the corpus, we had a ncdg score of 0.912. WIthout using any solutions that we found, just by implementing a better pre-processing and computing the BM25 score, we managed to improve the basic score from 0.75 to 0.91 which still indicates a great improvement of the original solution.

Although surprised by the fact that the solutions we had implemented didn't improve the ndcg score at all, we made assumptions to explain this event. Indeed, and as explained previously, we believe that this is because we’ve only used the vectors of the word2vec models, and not the models themselves - which was forbidden.

The vectors are not as efficient as the models directly and this is explained by our inability to alter its pre-processing values and its corpus. We’ve managed to deduce this by looking at the word vector similarity scores. For some model vectors like the ones from the bioWordVec, we’ve encountered occasional slashes or hyphens here and there, which wouldn’t be done in our own word2vec model. Hence, we believe that this is the reason why keeping the BM25 score was the better decision at the end.

After carrying this work, we believe that we have managed to get a satisfactory score that can be used in practical cases, although we also believe that the explorations described above will be the next step to take along with using the model described in this document instead of their keyed word vectors.