# MS Marco Document re-ranking

Team-002

Geoffrey Brenne
270421@uis.no

Célian Debethune
270412@uis.no

Paul Duffaut
270399@uis.no

## ABSTRACT

Knowing how to classify documents by a criterion of relevance to a query is a mechanism that is becoming increasingly important today through the many search engines. Today, whether through neural networks or algorithms, many methods have been tested with different results, more or less effective. In our case, we will first try to rank our documents with a BM-25 algorithm and then re-rank these documents with the BERT method. Finally, we obtain convincing results that are close to those previously achieved.

## KEYWORDS

information retrieval, machine learning,relevant document

## 1 INTRODUCTION

Today, when we are curious about something and we are looking for an answer to our question, we use a search engine. This one will return by order of relevance, the various sites in connection with our question. In our case, we will keep this principle but use it a little differently. Indeed, given a set of documents as well as a set of words placed in a query, we try to determine the ranking of the 1000 most relevant documents according to our query. Thus, in order to obtain these 1000 documents, our method consists in using the BM25 algorithm to associate a score for each document. BM25 is an algorithm that scans a set of documents and returns a score according to the frequency of the terms present in this document.

. In order to obtain an order on these documents with respect to a set of terms, this method seems perfectly adequate in view of the various reports already established. For example, Ivan Sekulic[6] or even Liana Ermakova[4] also rely on BM25 to obtain a ranking on their documents. The ease of implementation of this method according to its very convincing results makes it a suitable method that we have chosen to use. In addition, other studies have been done on other models that are more expensive or more difficult to implement, such as that of Simon Jaillet[3], who uses the SYGMART method.

## 2 PROBLEM STATEMENT

As explained in the introduction, the problem consists in ranking the documents of a database. We therefore want the documents at the top of the ranking to be the most relevant for the user. Our work focuses mainly on ranking and then reranking a set of documents according to a given query. Thus, our main program receives as input a query in string form and produces a text file containing the ranking of the 1000 most relevant documents according to our model. This document is of the form:

| qid | Q0 | docno | rank | score | tag |
|---|---|---|---|---|---|
| 1 | Q0 | nhslo3844_12_012186 | 1 | 1.733152 | mySystem |
| 1 | Q0 | nhslo1393_12_003292 | 2 | 1.725810 | mySystem |
| 1 | Q0 | nhslo3844_12_002212 | 3 | 1.725227 | mySystem |
| 1 | Q0 | nhslo3844_12_012182 | 4 | 1.725227 | mySystem |
| 1 | Q0 | nhslo1393_12_003296 | 5 | 1.713744 | mySystem |

**Table 1: Example of results in TrecRun format (from Trec-Tools documentation)**

The doc_id (or docno) are the id of the documents in MS-MARCO: a large dataset containing queries and documents in text form. This dataset allow the creation and training of ranking systems, based or not on machine learning. As said before, our work will not focus on the preprocessing of the documents and on the creation of the evaluation methods of our model, requiring a lot of computing power, energy and time:

- For the preprocess, we use the Pyterrier module which contains an inverted index of the documents in the MS-MARCO dataset that have been preprocessed. We also use this module to obtain information about the dataset such as the average size of the documents, necessary for the BM-25 function.
- We will also use the BM-25 algorithm of the Pyterrier module for the production of the top 1000 after having made sure that our version developed from our knowledge obtains the same results. We thus make sure that we have understood the algorithm while having optimal performance.
- To evaluate our model, we use the Trec-Tools module, which allows us to evaluate rankings according to several criteria such as Precision@k, Recall@k, NDCG, which allows us to know how our model performs.

Our model consists of a first ranking (baseline method) which retrieves the first 1000 documents and a reranking (advanced method), a more expensive method but allowing to obtain better results. The reranking of the documents is based on BERT only on the previous 1000 pre-ranked documents.

## 3 BASELINE METHOD

. The baseline method retrieves the 1000 most relevant documents according to the BM-25 model score from the index provided by pyterier.

Let remember the BM-25 formula:
**BM25 Formula:**

$$score(d,q) = \sum_{t \in q} \frac{c_{t,d} \times (1+k_1)}{c_{t,d} + k_1(1 - b + b\frac{|d|}{avgdl})} \times idf_t$$

**Parameters:**

- $k_1$: calibrating term frequency scaling
- $b$: document length normalization

This model uses often used statistics such as the frequency of appearance of a term in a document and the inverse document frequency (idf) which allows to modulate the action of the most common words and thus probably the least discriminating in terms of relevance of the document with respect to a query. As indicated on the score formula given above, BM-25 is based on the terms present in the query to calculate the score of the latter in relation to a document. The more frequently a document contains the terms of the query, the better its score will be. Similarly, if a document is rarely used, it will have a high idf, and documents containing this term will have a higher score. b and k1 are called smoothing parameters, commonly k=1.2 and b=0.75. This algorithm is based on the assumption that a document containing the terms of the query (and thus similar to this query) is probably a document containing the answer to this question, or at least deals with the same subject and is thus able to provide some information to the user. We have implemented this algorithm using the inverted index of pyterrier.

Our program generates the ranking of the 1000 runs with the best scores in TrecRun format, and then gives this document and the qrels in TrecQrel format to the Trec Tools package, allowing us to obtain different evaluations such as P@k, MRR, MAP... Two versions of the program exist: the first one is a python implementation of our part of BM-25 using the classic formula (3min per query to get the top 1000 on our PCs). The second one uses directly the BM-25 algorithm of the pyterrier module, much faster (10 seconds for the 200 queries of the test dataset). After having made sure that we obtained the same results with these two versions, we used the pyterrier version to evaluate the baseline method on the dataset. We performed the measurements on the test dataset of the TREC-2019 reranking competition.

We can see that the results are very close to those obtained by the MS Marco Team who implemented the baseline method. The slight differences are probably due to the use of a different stemmer or a different list of stopwords. These results ensure that we have a good base, however they could be greatly improved by using a second method, by reranking.

The raw performance of the baseline method BM-25 is more than once out of three very low, and therefore making statistics on our results was relatively difficult, as the ranking regularly contains no relevant documents. Nevertheless we obtain results of the same order of magnitude as those obtained by other teams and are therefore confident about the implementation of the algorithm, as well as the functioning and the good use of the pyterrier and Trec Tools packages. We can now try to improve the performance of our program by implementing an advanced method that will rerank the documents obtained by the baseline method.

## 4 ADVANCED METHOD

. This advanced method is algorithmically heavier and therefore cannot be applied to the 8.8 million documents. That's why we have previously performed a ranking by keeping the 1000 best documents. Thus we can then apply a heavier algorithm to better rank the documents kept, that is to say that from a set of documents already pre-processed, we will rank with BM25 in a first step these documents to obtain the 1000 most relevant of them given a query.

In a second step, from these 1000 documents, we will re-rank this selection to order them from the most relevant to the least relevant. The reranking part will be done with BERT Cross-Encoder. So in this case we input simultaneously the query and the document, allowing to have as output a number representing the relevance of the query and the document between them. (In our case the numbers are not put between 0 and 1 with a sigmoid function in order to save computing time).

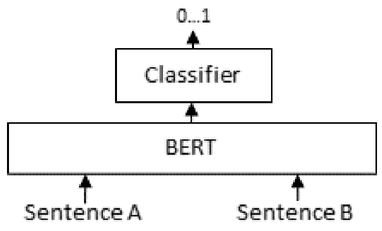

**Figure 1: Cross Encoder**

**Table 2: Example for the query "is goldfish grow"**

| doc | result |
|---|---|
| doc 1 | 5,25 |
| doc 2 | 5,25 |
| doc 3 | 5,25 |

Moreover, the use of BERT, which is a model much used in the literature, allows to have a BERT already trained on MS-Marco values and thus to have better results. This is the reason why we decided to use the sBert (ie Cross-Encoder) library of python which proposes a fast BERT model already pre-trained for MS-Marco.

So to summarize our advanced method we apply the baseline method as previously explained then we take the 1000 best ranked documents to reclassify them via BERT which is used to give a value to the fact that the query and the document are correlated. And from this new value we re-sort the 1000 documents.

## 5 RESULTS

**Table 3: BM-25 and BERT evaluation on test dataset**

| Method | Aveg | P@5 | P@30 | P@100 | R-Prec | Rcl |
|---|---|---|---|---|---|---|
| BM-25 | 0.3587 | 0.6259 | 0.4315 | 0.2444 | 0.3819 | 0.7512 |
| BERT | 0.4629 | 0.7889 | 0.5333 | 0.2913 | 0.4589 | 0.7481 |

When we compare our different values to those present in the different previous publications using the BM-25 method, we realize that our values obtained are very close to those we have. For example, in Robertson's paper [5], he obtains an AveP of 0.35 while we also have an AveP of 0.35. Their P@5, P@30 and P@100 are respectively 0.72; 0.59 and 0.44. On our side, our values are slightly lower but remain in the same range (0.63 instead of 0.72; 0.43 instead of 0.59 and 0.24 instead of 0.44). Finally, our R-Prec is similar to theirs with a value of 0.38 while our Rcl is slightly higher than theirs (0.75 instead of 0.70). Our values seem consistent as they are close to those obtained in previous work. On the side of our BERT, our values are better than the ones we had with BM-25 which seems correct because we re-Rank our documents classified by BM-25. Moreover, when we take the values obtained by previous studies, such as Aliannejadi's [1], we obtain values for precision very similar to theirs. Indeed, they have a value for P@5 of 0.66 while we have a value for P@5 of 0.78. The value of our Rcl is slightly lower than that of our BM-25 because one of our documents was not ranked during the re-Rank.

## 6 DISCUSSION AND CONCLUSIONS

. During the implementation of our re-rank, the difficulties encountered were to manage to manipulate different libraries, some of which had documentation that was difficult to understand. Then, we had to link the output of our BM-25 to the input of our BERT and the fact that the two are not compatible required us to manipulate our results in order to put them in a different format acceptable for the BERT method.

Finally, our results are very satisfactory because we notice, as expected, a clear improvement of the performance during the re-rank. Moreover, when we compare with previous works in the field, we have very similar results which justify the coherence of the work done.

The complexity of the BERT method compared to that of the BM-25 method, obviously makes the results obtained after our BERT re-rank more relevant than those after our BM-25 rank. We could have taken another better trained BERT model, which would have allowed us to have other, perhaps more efficient results. On the other hand, we might have had to use another algorithm than BM-25, a bit more complex to get a more relevant first ranking.

## REFERENCES

[1] Mohammad Aliannejadi, Manajit Chakraborty, Esteban Andrés Ríssola, and Fabio Crestani. 2020. Harnessing evolution of multi-turn conversations for effective answer retrieval. In *Proceedings of the 2020 Conference on Human Information Interaction and Retrieval*. 33–42.

[2] Negar Arabzadeh, Bhaskar Mitra, and Ebrahim Bagheri. 2021. MS MARCO Chameleons: Challenging the MS MARCO Leaderboard with Extremely Obstinate Queries. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 4426–4435.

[3] Jacques Chauché, Violaine Prince, Simon Jaillet, and Maguelonne Teisseire. 2003. Classification automatique de textes à partir de leur analyse syntaxico-sémantique. In *Actes de la 10ème conférence sur le Traitement Automatique des Langues Naturelles. Articles longs*. 55–64.

[4] Liana Ermakova and Josiane Mothe. 2016. Document re-ranking based on topic-comment structure. In *2016 IEEE Tenth International Conference on Research Challenges in Information Science (RCIS)*. IEEE, 1–10.

[5] Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. 1995. Okapi at TREC-3. *Nist Special Publication Sp* 109 (1995), 109.

[6] Ivan Sekulić, Amir Soleimani, Mohammad Aliannejadi, and Fabio Crestani. 2020. Longformer for MS MARCO document re-ranking task. *arXiv preprint arXiv:2009.09392* (2020).

## A DIVISION OF WORK DURING THE PROJECT

We met during each workshop to work in groups. In addition, we often made appointments with our supervisors and took into account the modifications and advice given. In the end we met more regularly to finalise the project.