# MS Marco Document Re-ranking

Team 003

Álvaro Esteban Muñoz
264555@uis.no

Rainui Ly
264553@uis.no

Maria Eriksen
228891@uis.no

## ABSTRACT

The following report summarizes and discusses the work done during the group project in DAT640 - Information Retrieval and Text Mining. The main purpose of the project is to use obtained knowledge solving an open information retrieval problem. This will be done by first defining a baseline method then re-ranking the documents, and finally comparing the advanced methods to the baseline by using a standard test collection. Besides, participants will practice their skills of solving problems as a team.

## KEYWORDS

Information retrieval, Text retrieval, Language Model, Machine learning

## 1 INTRODUCTION

Re-ranking is a major task of modern Information retrieval. A lot of research is done on the field because of its complexity, featuring problems like vocabulary mismatch and data representation. Thus, many people take part in research on this topic and, to raise the interest for this research among even more people, there are several competitions related to text retrieving.

Text REtrieval Conference (TREC) 2019 Deep Learning Track is one of these competitions. The goal of the Track is to study the which methods work better at information retrieval of a large dataset. Track had two tasks containing large human-labeled training sets. 15 groups of participants were competing against each other during TREC 2019 Deep Learning Track [5]. Participants try different approaches, such as Machine Learning or classical information retrieval methods, in order to find out which is the best option for achieving the goal of this task.

The dateset used in TREC 2019 is the MS MARCO (Microsoft MAchine Reading COmprehension) dataset, which is made available for the researchers to promote advancement in fields like AI, etc. Initially, The MS MARCO contained one hundred thousand real-life Bing questions, but soon evolved into becoming a collection of data, containing among others: one million questions and answers, a natural language generation and a passage ranking datasets, etc. [3]

Taking a look at the results of TREC 2019 Deep Learning Track, all the participating groups that are ranked the highest are using BERT pre-trained transformer for re-ranking, which is a neural approach. For the purpose of research, we are choosing to study the same problem from the perspective of classical Machine Learning only.

## 2 PROBLEM STATEMENT

As stated in the official presentation document of the MS MARCO dataset, the data collection contains more than one million questions that were extracted from the logs of Bing's search query. Each of these questions has a corresponding human-generated answer, in addition to approximately 180 thousands completely human rewritten answers. As mentioned in Introduction, the dataset is not limited to questions and answers - among others, the collection consists of **8,841,823** passages that were extracted from Bing's **3,563,535** web documents; this information is important when working with the natural language answers. [3]

To generate more than one million questions with corresponding unique answers, queries from Bing's search logs were sampled and non-question queries were filtered out from this set. The resulting **3,563,535** retrieved relevant documents for each question were extracted from Bing's index [8]. Our actual dataset consists of **3,213,835** documents, structured using three fields: (i) the URL, (ii) the body text, and (iii) the title.

It is worth mentioning that our actual dataset is missing around 300,000 documents; this is due to the fact that they were no longer in the index. Moreover, for the rest of the documents, it is possible that the content is not the same as when they were first retrieved. [3]

In addition, we are given a set of 377,999 queries split in train, dev and test datasets (367,013, 5,193 and 5,793 respectively) and a top-100 documents retrieved by Indri Query Likelihood model per query. Thus, we are already given a baseline.

Table below shows the summary of the MS MARCO Document Ranking dataset [8].

| File | Records | Size |
|---|---|---|
| Corpus | 3,213,835 | 22GB |
| Train queries | 367,013 | 15MB |
| Train top-100 documents | 36,701,116 | 1.8GB |
| Train qrels | 384,597 | 7.6MB |
| Validation queries | 5,193 | 216KB |
| Validation top-100 documents | 519,300 | 27MB |
| Validation qrels | 5,478 | 112KB |
| Test queries | 5,793 | 124KB |
| Test top-100 documents | 579,300 | 2.9MB |

The aim of the project is to study different model's ability to outperform the baseline retrieved by the Indri Query Likelihood model [10]. First, all documents will be pre-processed and indexed in order to retrieve information. Then, several more sophisticated retrieval models will be used to re-rank the documents depending on the agreed criteria. Finally, the new ranking will be compared with the old one according to the baseline defined.

## 3 BASELINE METHOD

A **baseline** is a plain model that is used to calculate predictions for a given dataset, using simple metrics [7]. These values will be

used later to compare the performances of the baseline and the **advanced method** to see if there is any improvement.

In the particular project, the initial set of data is already ranked, and it is therefore reasonable to use this ranking as the baseline for the future comparison with the re-ranked models.

Our initial dataset is top-100 documents per query, which were retrieved by the **Indri Query Likelihood model** with Krovetz stemming and stopword removal. This model is a combination of **Inference Network models** and **Language models**.[10] This is very convenient because it allows the usage of language modeling estimates (while) performing the evaluation of the structured queries within the network, instead of using tf-idf.

The main difference of this model compared to classical language models is the fact that documents are represented as binary features. This makes it possible to model not only word occurrences, but also other features like capitalization of the words in the text. Due to binarity, a multiple-Bernoulli model is estimated for both full text and subsections (paragraphs) of each document. This leads to the weakness of the model in the context of text retrieval, as the multiple-Bernoulli model assumes independence of the features.[10]

The second component of an Indri Query Likelihood model, Inference Networks approach, is a method of combining different evidence to determine the relevance of the document. Such evidence is occurrence of phrases or terms from the query within a document.[10]

Mathematically, Indri Query Likelihood model is equivalent to the Dirichlet smoothing estimate of the multinomial model, and may be expressed as follows: [10]

$$P(r_i|D,\alpha,\beta) = \frac{\#(r_i,D) + \mu P(r_i|C)_i}{|D| + \mu} \tag{1}$$

where $r_i$ are the document features, $\#(r_i, D)$ is the number of times a feature $r_i$ is set to 1 in document $D$'s multiset of feature vectors, $\alpha$ and $\beta$ are hyperparameters, $\mu$ is a tunable smoothing parameter and $C$ is a constant.

Measuring of the baseline will be performed using the same measures as in the official TREC 2019 Deep Learning Track, which are Normalized Discounted Cumulative Gain (NDCG@10, NDCG@100), Mean Reciprocal Rank (MRR@10, MRR@100) and Mean Average Precision (MAP).

## 4 ADVANCED METHOD

The main goal of the task is to compare different ways of re-ranking. We have chosen different approaches, such as classical retrieval and learning-to-rank, and will review their performance on this task. When talking about classical information retrieval, we focus specifically on probabilistic models, since they are the dominant paradigm today [6].

### 4.1 Probabilistic models

Probabilistic models are well-known for their effectiveness, due to being based on a strong foundation of mathematical theory - **the probabilistic theory**. Under this category, there are different probabilistic approaches; the most common ones will be used for re-ranking in this task.[6] We will be using PyTerrier's default parameters, unless the models are parameter-free.

*4.1.1 BM25 Re-ranking.* One of our advanced methods will be BM25 ranking algorithm. It extends the scoring function for the binary independence model to include document and query term weights. This model is dependent on two parameters; $k_1$ (calibrating term frequency scaling) and $b$ (document length normalization).

$$score(d,q) = \sum_{t \in q} \frac{c_{t,d} \times (1 + k_1)}{c_{t,d} + k_1 \times (1 - b + b\frac{|d|}{avgdl})} \tag{2}$$

PyTerrier's version of BM25 uses an implementation with three parameters. However, it only allows you to modify the document length normalization: $k_1 = 1.2$, $k_3 = 8$ and $b = 0.75$.

Moreover, we will also try the fielded version of this algorithm. It replaces the term frequencies with pseudo term frequencies that depend on the field.

$$score(d,q) = \sum_{t \in q} \frac{\tilde{c}_{t,d}}{k_1 + \tilde{c}_{t,d}} \times idf_t \tag{3}$$

For this version, PyTerrier doesn't let you modify any BM25's specific parameters, nevertheless you can choose weight fields.

*4.1.2 Language models.* Language Models is another family of approaches under the probabilistic models. Every word in the language has an associated probability of occurrence; Language Models are based on the probability distribution of words in a language. [6]

$$score(d,q) = logP(q|d) = \sum_{t \in q} logP(t|\theta_d) \times c_{t,q} \tag{4}$$

The value of $P(t|\theta_d)$ estimate will depend on the model used. We will be running an experiment with the Dirichlet smoothing using $\mu = 2500$ and Hiemstra's smoothing with $\lambda = 0.15$.

*4.1.3 Divergence From Randomness models.* Divergence From Randomness models (DFR) are term-document matching functions which are calculated by multiplying two divergence functions. [2]

$$\sum_i I_1(\hat{p}_i^+||\hat{p}_i) \cdot I_2(\hat{p}_i||p_i) \tag{5}$$

These models are very easy to implement with PyTerrier's DFR framework. This framework is based on the combination of three components:

(1) Randomness Model **RM** (Poisson distribution);
(2) Information Gain Model **GM** (Laplace after-effect); and
(3) Term Frequency Normalization Model (Normalization 2).

We decided to study PL2 model, and it's fielded version.

The only parameter that needs to be defined is the term frequency normalization parameter; $c = 1.0$.

*4.1.4 Multinomial From Randomness model:* Multinomial from randomness models are designed to make the fields of the documents an integral part of the randomness model. In other models, such as PL2F, the term frequencies are supposed to follow the same distribution in each field. Having a multinomial from randomness model allows us to use multinomial distribution to integrate the document fields of the model.

$$P_M(t \in d|D) = \binom{TF}{tf_1 tf_2 ... tf_k tf'} p_1^{tf_1} p_2^{tf_2} ... p_k^{tf_k} p'^{tf'} \quad (6)$$

From this family, we chose to look at ML2 and MDL2 models. The only difference between them is that for the second one we use an approximation of the multinomial distribution. [9]

Both models need two parameters in PyTerrier:

- $c_i$: Normalization parameter for each field (1.0 as default).
- $p_i$ Prior weight adjustments for each field (1.0 as default).

*4.1.5 Hyper-geometric from Randomness Models.* Hyper-geometric models are mainly concerned of two probabilities: the relative term frequency withing the document and the term frequency within the entire collection. Keeping things simple, these models assume a high divergence maximum likelihood estimate and the prior probability; $P(tf|d, p = P(t))$. [1]

$$P(tf|d) = \frac{\binom{TF}{tf} \cdot \binom{TFC-TF}{l(d)-tf}}{\binom{TFC}{TF}} \quad (7)$$

In PyTerrier, these models take advantage of the DFR framework. Notice these models are parameter free, so we do not need to specify anything when we construct them in using PyTerrier.

## 4.2 Learning-to-rank approach

Having plenty of training data makes it possible to learn a **discriminative model**. Discriminative models associate a probability of belonging to a class from observed features of the document found in training data. These models are expected to be superior when enough data is given during the training, due to the huge amount of features that can be considered for web ranking [6]. We consider this approach worth trying since we dispose of a large quantity of data to train the model as well as relevance judgments.

PyTerrier framework will be used to implement the Learning-To-Rank pipeline. There are three stages to be performed. First, rank the topics with a simple model; this is already done by the TREC, so the results will be directly add to the LTR pipeline. Then, re-rank the topics with the features selected. We decided to use the two best simple models as extra features (DFRee and DPH). Finally, apply a regressor on the data, where **LambdaMART** is the one we have chosen. In fact, LambdaMART outperforms the rest of the pairwise LTR approaches. [4]

LambdaMART is a pairwise ranking algorithm, in other words, given two documents, it classifies which document should have a better rank than the other. Difference between the pair is estimate as a probability based on the **sigmoid function**. Then, **cross entropy** is computed to measure how close are the two probability distributions. **Backpropagation** technique allow us to compute the weights in the learner that minimize the cross entropy loss. [4]

# 5 RESULTS

The following results are computed with Python using a library called PyTerrier. It is a Python 3.6 wrapper of the Java-based Terrier information retrieval platform that support indexing and retrieval operations.

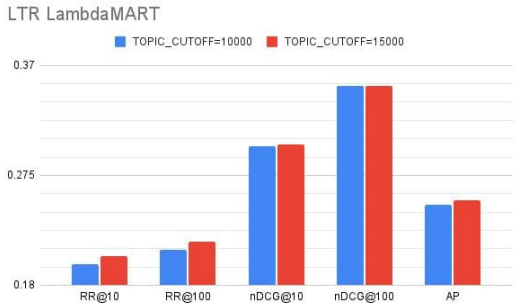
The chosen index structure is an inverted index based on the fields provided by the dataset which allows fast full-text searches at the cost of 5 hours of documents processing.

| Description | Value |
|---|---|
| Number of documents | 3213835 |
| Number of documents | 3213835 |
| Number of terms | 16779685 |
| Number of postings | 925509293 |
| Number of fields | 3 |
| Number of tokens | 2281485681 |
| Field names | [url, title, body] |
| Positions | false |

The following table shows the calculated metrics for each approach.

| | RR@10 | RR@100 | NDCG@10 | NDCG | AP |
|---|---|---|---|---|---|
| DFRee | 0.265 | 0.276 | 0.325 | 0.377 | 0.276 |
| DPH | 0.264 | 0.275 | 0.325 | 0.376 | 0.275 |
| BM25 | 0.260 | 0.271 | 0.320 | 0.373 | 0.271 |
| PL2 | 0.257 | 0.268 | 0.319 | 0.371 | 0.268 |
| PL2F | 0.253 | 0.264 | 0.314 | 0.368 | 0.265 |
| BM25F | 0.252 | 0.263 | 0.311 | 0.366 | 0.264 |
| HiemstraLM | 0.250 | 0.261 | 0.310 | 0.365 | 0.261 |
| ML2 | 0.241 | 0.253 | 0.299 | 0.357 | 0.252 |
| LambdaMART | 0.228 | 0.240 | 0.290 | 0.346 | 0.245 |
| **Baseline** | **0.209** | **0.222** | **0.263** | **0.329** | **0.222** |
| MDL2 | 0.203 | 0.216 | 0.246 | 0.317 | 0.209 |
| DirichletLM | 0.191 | 0.205 | 0.245 | 0.315 | 0.205 |

The chart shows the relationship between the amount of training data and the metrics scores for our LTR model.




**Comparison of LambdaMART with two different load of training topics**

# 6 DISCUSSION AND CONCLUSIONS

## 6.1 General

MS MARCO collection provides a large corpus of document. The first problem encountered was the corpus' indexing. The indexing was performed using PyTerrier as it is suitable for TREC format files.

We indexed the pre-processed documents instead of pre-processing documents after retrieving from index in order to save time. We performed tokenization, stopwords removal and stemming with PorterStemmer. This pre-processing will help us to partially solve the vocabulary mismatch problem, thanks to stemming process, but we could have addressed this problem by using query expansion. Nevertheless, the creation of the inverted index of all the documents took about 6 hours.

As mentioned in Introduction, the Deep Learning is the common approach used for Track. Therefore, we wanted to try different methods that we saw during lectures. We chose to work on Probabilistic Models and Learning-To-Rank instead of Deep Learning. To make it possible, it was necessary to work on 'irds:msmarco-document/dev', instead of 'irds:msmarco-document/train' dataset, which is a smaller dataset because of our hardware limitations.

Indeed, many Machine Learning libraries are designed for CPU. For the purpose of research, the framework selected for the Learning-To-Rank approach is LightGBM. It is primary designed for CPU computation. After experiencing memory errors due to lack of power, we opted for Google Colab Pro as our development environment to acquire the necessary RAM resources needed for this method.

## 6.2 Baseline

In this project, Indri Query Likelihood model was a natural choice when deciding on the baseline; this model has a weak performance compared to many other simple models that we have tried during our work on the project. This may be explained by the fact that this model is not a sheer Language Model, but also contains an Inference Network nature. Therefore, this model would probably not be a clear choice for us if we had taken the decision ourselves. Additionally, since we were not performing the original ranking, we did not have a chance to neither tune the baseline nor try a different approach. Potentially, this could have affected the project's outcome.

If we could choose freely, we would probably use BM25 as the baseline; although it is widely used, many researchers consider BM25 a weak baseline [7]. However, keeping in mind test results, BM25 could have been a good measure in our research. Indeed, we have found several probabilistic models that outperformed BM25, but also several models that were weaker than the potential baseline.

## 6.3 Discussion of the results

Divergence from Randomness Models are based on the measure of divergence and are used to test the amount of information carried in the documents. These models are therefore perfect for our task, since we want to compare the amount of relevant information to decide which document has the most of it. Hyper-geometric-based versions of Divergence from Randomness Models perform better since they are more accurate at large data scales - Poisson and Binomial approximations are good too, but when we focus on a big population, we start to lose some information.

BM25 performed well compared to other models; however, since it is based on binary independence model, it's still a poor model when it comes to computing the amount of information at large scales of data.

The Language Models showed relatively poor performance; these models provides context to distinguish between words and phrases that are phonetically similar. In other words, this is a model that performs very good at voice recognition tasks, but is probably less efficient when it comes to retrieving from the internet.

Multinomial from Randomness Models are very good for specific tasks related to fields, such us topic distillation, named page finding and home page finding, which is not the case in this task. Further, we notice that all fielded versions perform more poorly than regular versions. [9]

An interesting observation was made while studying the results - parameter free models tend to have similar measures, while models that have parameters have some spread within the results. For example, both Divergence from Randomness Models have almost the same score, while for the two Language Models the score is different. This may be explained by the fact that we didn't tune the models, and only used default parameters when performing the re-ranking.

Finally, the LTR approach performed nice, perhaps not as well as it is expected for an ML model. However, we proved that the effectiveness of this model is strongly related to the amount of data given to the model; giving more a larger amount of data would result in a completely outperforming of all the other models. The extra features computed for the model were DFRee and DPH, both belonging to Hyper-geometric Divergence from Randomness models. We could have computed features from diverse approaches in order to get information that approaches from the same family does not retrieve.

## 6.4 Conclusion

A detailed result of the performance of different probabilistic models is presented in table **Calculated Metrics** in chapter 5. The difference in the score among the models is not dramatic; however, the method which has the best performance is the Hypergeometric Divergence From Randomness model DFRee. Nevertheless, LambdaMART could have been a better alternative to DFRee depending on the amount of data given for training the model. Indeed, we have experienced that, the more the data we provide to the model, the more the metrics' scores are increasing, but the more the computation cost is increasing too.

## REFERENCES


[1] G. Amati. 2017. Frequentist and Bayesian Approach to Information Retrieval. *Fondazione Ugo Bordoni, Rome, Italy* (2017).
[2] G. Amati and U. Bordoni, F. 2009. Divergence From Randomness Models. (2009).
[3] P. Bajaj, D. Campos, C. Craswell, L. Deng, J. Gao, X. Liu, R. Majumder, A. McNamara, B. Mitra, T. Nguyen, M. Rosenberg, X. Song, A. Stoica, S. Tiwary, and T. Wang. 2016. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. (2016).
[4] K. Chung. 2019. Introduction to Learning to Rank. (2019).
[5] N. Craswell, B. Mitra, E. Yilmaz, D. Campos, and M. Voorhees, E. 2020. Overview of the TREC 2019 Deep Learning Track. (2020).
[6] Metzler D. Croft, W. B. and T. Strohman. 2015. *Search Engines. Information Retrieval in Practice.*
[7] D. Li, E. Hasanaj, and S. Li. 2020. Baselines. *Carnegie Mellon University.* (2020).
[8] B. Mitra and Microsoft. 2016. MSMARCO Document Ranking. (2016).
[9] Ounis I. Plachouras, V. 2007. Multinomial Randomness Models for Retrieval with Document Fields. *University of Glasgow, UK* (2007).
[10] Metzler D. Turtle H. Strohman, T. and W.B. Croft. 2005. Indri: A language model-based search engine for complex queries. *International Conference on Intelligence Analysis.* (2005).

# Appendices

## A DIVISION OF WORK

The division of work was naturally done according to our strengths and weaknesses, but also according to the interest that each of us had in the subject.

Álvaro was interested in the topic of this subject because of his interest in Machine Learning. In preparation for his Bachelor's thesis on this field, this subject is a step in the right direction. Therefore, he undertook the project as a leader and focused on the problem-solving and knowledge required to complete the project.

Rainui was involved in the implementation of the project and preferred to focus on the technical challenges of information retrieval. From the choice of the framework (PyTerrier) to the choice of the code platform (Google Colab), he took care of implementing what the team decided to do.

Maria has experience of writing scientific papers and has followed Machine Learning course before. Hence, she helped Alvaro in the search for information about the re-ranking, sorting out what was useful in the report and what was useful for the implementation of the code.

In addition to that, the team has opted for a flexible working method adapted to the rhythm of each individual. As a result, the use of sharing and communication tools such as Telegram, Google Colab, Google Drive and Github was of great assistance. Also, the team met four times a week to ensure that the project was completed on time. Our workload can be quantified in the following way.

**Workload**

| | Code | Research | Report |
|---|---|---|---|
| Alvaro | 10% | 60% | 30% |
| Maria | 10% | 30% | 60% |
| Rainui | 80% | 10% | 10% |