Team-013: MS MARCO Document re-ranking task

Sondre Tennø
University of Stavanger
s.tenno@stud.uis.no

Anthony Ibeme
University of Stavanger
a.ibeme@stud.uis.no

Einar M. Langholm
University of Stavanger
e.langholm@stud.uis.no

ABSTRACT

Machine Reading Comprehension dataset, also known as MS MARCO, is a dataset gathered from a little over one million anonymized questions. The questions are retrieved from Bing's search query logs and have been answered by real humans [9]. This paper introduces our implementation of re-ranking the top 100 documents retrieved by the Indri model [3] by relevance. Both a baseline and an advanced method is introduced to show how an advanced method performs in comparison to the baseline method. Our baseline model is a learning to rank model based on a few features that takes query word sequencing into account. The advanced method builds upon the baseline model adding the score from the initial retrieving function in addition to a few more features . Both our baseline and advanced model assumes no prior knowledge about the document collection, only information from the top 100 documents are used for feature extraction. The ranking models implemented in this project shows minimal improvement over the official baseline language model Indri [3]. Feature extraction and subsequent ranking using our models are significantly slower than established methods.

KEYWORDS

Ranking, Document Re-ranking, Information Retrieval, MS MARCO, Indri

1 INTRODUCTION

MS MARCO is a dataset gathered from a little over one million anonymized questions. The questions are retrieved from Bing's search query logs and have been answered by real humans [9]. In addition, the dataset contains more than 8,8 million passages extracted from a little over 3,5 million web documents [3]. The top 100 documents for each query have been retrieved by using Indri language model [9]. In this project we want to do a re-ranking of these documents to try to improve the mean reciprocal rank. We will implement both a baseline and an advanced ranking function based on the Indri ranking of these documents.

The three main approaches to tackle NLP(Natural processing language) tasks are rule-based, traditional machine learning, and neural networks.

Rule-based are the oldest of the common NLP approaches. Since it is the oldest, it is very well tested. Rule-based approaches are low precision, high recall, meaning they can have high performance in specific use cases, but often suffer performance degradation when generalized [8].

Traditional Machine Learning approaches include probabilistic modeling, likelihood maximization, and linear classifiers. Traditional Machine Learning is using a set of algorithms that parse data, learn from them, and then apply what they have learned to make intelligent decisions [8].

In Neural Networks feature engineering is generally skipped, as networks will "learn" important features (this is generally one of the claimed big benefits of using neural networks for NLP). Instead of feature engineering, it will use streams of raw parameters without engineered features, and are fed into neural networks [8].

While a Machine Learning model makes decisions according to what it has learned from the data, a Neural Network arranges algorithms in a fashion that it can make accurate decisions by itself. Thus, although Machine Learning models can learn from data, in the initial stages, they may require some human intervention [7].

Neural networks do not require human intervention as the nested layers within pass the data through hierarchies of various concepts, which eventually makes them capable of learning through their own errors [7].

In this project we have decided to go with a traditional machine learning approach.

Related Work. On the official MS MARCO website, we can see the official leaderboard re-ranking score of the 30 top implementations. Here anyone can submit their work and be evaluated by the MS MARCO team to show others your score and implementation.

Taking a look at the leaderboard for document re-ranking, we can see that most of the top implementations are implemented using different variations of BERT. BERT stands for Bidirectional Encoder Representations from Transformers and is a language representation model by Google. It uses two steps, pre-training and fine-tuning, to create state-of-the-art models for a wide range of tasks. The number one implementation on the leaderboard as the competition has been concluded, are an implementation by Zhiyuan Liu, Chenyan Xiong and Maosong Sun and their Openmatch approach [11]. Also most of the leaderboard implementations are using a neural network approach. This is something we won't implement in this project, since we are going for a feature based traditional machine learning approach.

1.1 Problem Statement

Given a set of 100 documents ranked in relevance to a query, re-rank the documents in terms of their relevance to the question

2 BASELINE MODEL

Our baseline model is a feature based learning to rank model that takes query word sequences into account. The idea is that words that appear in the right sequence or in the same sentence are more likely to be related to the given query than when the same number of matches appear ordered in a text. Naturally, this might not improve ranking for single word queries given the characteristics of the initial ranking function. Each query in this dataset has been

DAT640, Fall 2020, Stavanger, Norway
2020.

evaluated to be an actual question and therefore single word queries are not present [1] , visualised in figure 3. Here we can see clearly that there is not a single one word query. Query length ranges from 2 to 15 word queries. The query is tokenized and split in to sequences of two and three words (bigram,trigram). The same tokenization is done to the body and title of each document. The number of times these short sequences appear in the document is counted. Number of sentences containing two or more query terms is counted. This feature is called unordered bigram in this paper.

Query document features .

- Unigram TF score field. Max,min and mean score for each term in the query
- Ordered bigram. Max,min and mean score for each bi-gram in the query. (word bi-grams)
- Ordered trigram. Max,min and mean score for each tri-gram in the query. (word tri-grams)
- Unordered bigram

Document features :

- Length of title
- Length of body

Query features :

- Length of query

For our baseline model, we're using a random forest based learning to rank algorithm to solve our problem in a pointwise manner. We've used the sklearn library for Random Forest Regressor combined with the multiple different features we have generated. The Random Forest Regressor are of version 3.2.4.3.2.sklearn.ensemble. Random Forest Regressor with parameter max depth = 5, and the rest being standard parameters.

Some of the functions we have used to create features and model ranking, are given to us by Krisztian Balog from the course DAT640-1 20H Information Retrieval and Text Mining in Assignment A5 [2].

2.1 Data exploration and preprocessing

The datasets are downloaded from the official MS MARCO Microsoft website [9]. Details on the dataset created for bench-marking machine reading comprehension and question-answering models is specified in [9]. Queries in this dataset are based on genuine Bing searches. Only queries that are believed to be genuine questions are present in the dataset.

Table 1: Example query .

Query id: 645590.
Query: what does physical medicine do

Doc	Rank	Score*	Title	url	Body
D877577	1	-4.97612	Sports Medicine	answers.com	....
D216558	2	-4.97992	Research Maniacs..	researchmaniacs	...
D1035880	3	-5.00558			
.....					
D359258	100	-5.74551			

Example query showing the information available for each document query pair and information from the initial ranking.
*Indri Query Likelihood

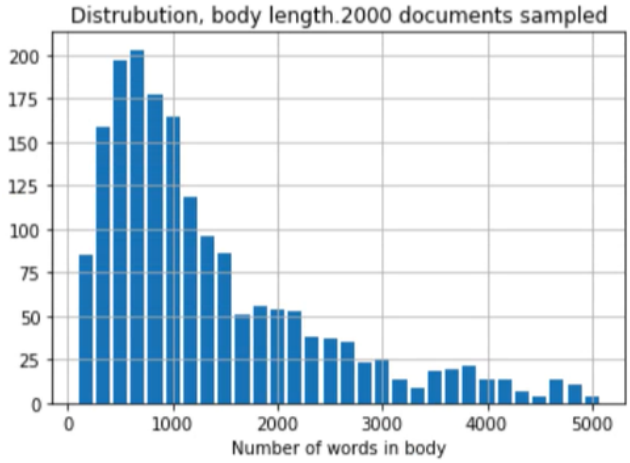

Figure 1: This plot shows how body length is distributed among a sample of the training set. The numbers are based on the queries as is, without stopwords removed.

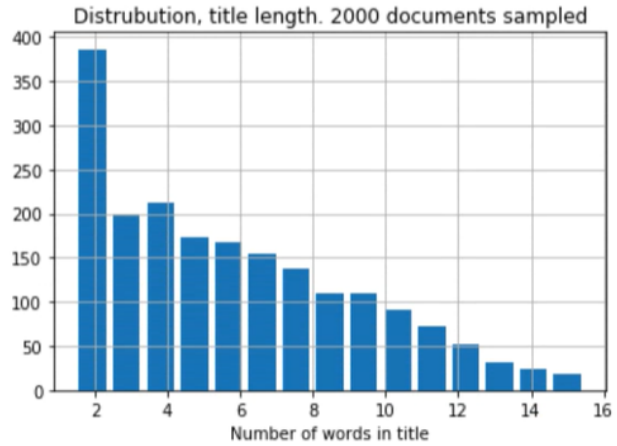

Figure 2: This plot shows how title length is distributed among a sample of the training set. The numbers are based on the queries as is, without stopwords removed.

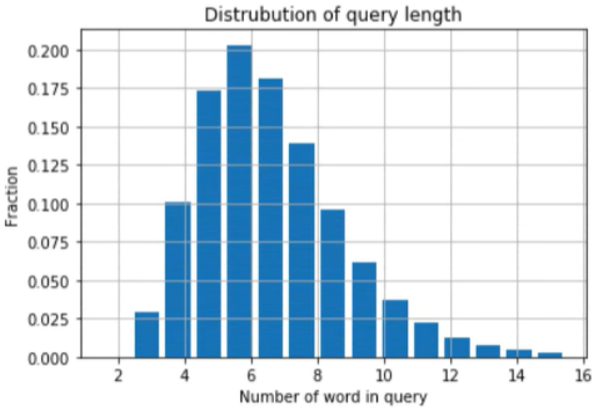

Figure 3: This plot shows how query length is distributed in the training set. The numbers are based on the queries as is, without stopwords removed.

2.2 Preprocessing

The dataset is indexed using Elasticsearch and prepossessing is done specifying filters used in Elasticsearch standard analyzer. The filters we have added to the analyzer, are a standard stop word filter [6], lowercase filter [5] and an algorithmic and a dictionary based stemmer based on Krovitz [4]. The urls are prepossessed with a simple function to split them into words before being tokenized.

2.2.1 Feature extraction. Features used for re-ranking are based solely on the query and the documents retrieved by the initial retrieval function. Apart from the scores given by this retrieval function, we assume no prior knowledge about the collection the documents were retrieved from. This allows for more computationally expensive features to be extracted. Some documents are significantly larger than the norm 1. We are limiting the length of the document bodies used for calculating the bigram and trigram features.

2.3 Generating training data

The MS MARCO ranking dataset is highly unbalanced, with the number of relevant documents being only a small fraction of the retrieved documents for each query. There are no true negative samples in the training and development data, this leads to the possibility for mislabelled samples. To tackle this problem, we generated a more balanced dataset using only a few documents for each query in our training set. All documents relevant to the given query and two documents retrieved in the initial ranking, but not considered relevant. The two documents considered not relevant are taken from the lowest scoring documents for the given query. The idea is that these documents have a lower possibility for being true positives, even though they contain terms relevant to the query.

3 ADVANCED MODEL

Our advanced model builds on our baseline, adding additional features to improve the quality of the ranking. That means that we're still using a random forest regression based learning to rank approach in a pointwise manner, but with more features gathered from the data. Since the advanced model is built on extending the baseline, the advanced model is using the same Random Forest Regressor as the baseline, with the same version and parameters.

Query document features .

- Retrieval score of the entire document, Indri
- Unigram TF score field. Max,min and mean score for each term in the query
- Ordered bigram. Max,min and mean score for each bi-gram in the query
- Ordered trigram. Max,min and mean score for each bi-gram in the query.
- Unordered bigram
- Exact matches
- Email token matches

Document features :

- Length of title
- Length of body

Table 2: Files retrieved from MS MARCO .

Type	Filename	File size	Number of records
Corpus	msmarco-docs	22 GB	3213835
Train	msmarco-doctrain-queries	15 MB	367013
Train	msmarco-doctrain-qrels	7,6 MB	384597
Dev	docleaderboard-top100	27 MB	519300

Table 2: Table above shows the statistics summary of the documents we've used in this project.

Table 3: Files for rank testing .

Type	Filename	File size	Number of records
Dev	msmarco-docdev-queries	216 KB	5193
Dev	msmarco-docdev-top100	27 MB	5193
Dev	msmarco-docdev-qrels	112 KB	5478

Table 3: Table above shows the statistics summary of the documents used for ranking test files.

4 RESULTS AND DISCUSSION

Evaluations are done using the first 500 queries from the development set. The development set can be found here[10]. MRR@100(Mean reciprocal rank) are used as the official evaluation metric.

Testing on 260 random queries. For the ranking we're using the evaluation script given by the MS MARCO team for ranking MRR@100. In figure 4, we have done an experiment, ranking our models on 260 random generated queries. Here we can see that our baseline is slightly ranking higher than the Indri model, and our advanced model slightly ranking higher than our baseline model. Giving scores of 0,248 MRR@100 for Indri, 0,256 MR@100 for baseline and 0,275 MRR@100 for our advanced model.

Model	260 Random queries MRR@100
Indri	0,248
Baseline	0,256
Advanced	0,275

Table 4: Showing results from our model's performance on 260 random queries.

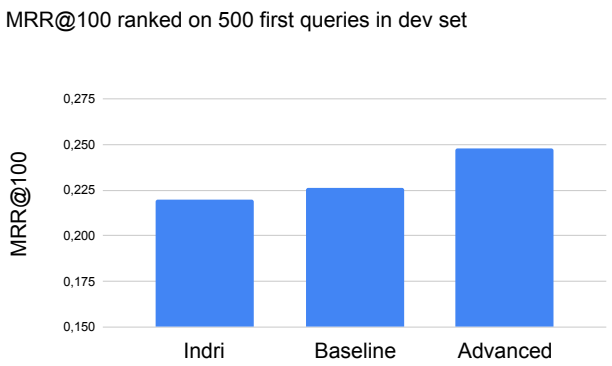

Figure 5: Showing the MRR@100 ranking score for Indri, our baseline model and our advanced model on 500 first queries from the dev set.

Model	Dev set 500 MRR@100
Indri	0,220
Baseline	0,226
Advanced	0,248

Table 5: Showing results from our model's performance on 500 first queries from the dev set.

4.0.1 Baseline. We can see that our baseline model marginally outperforms the Indri model on the training sets we tested on, both the 260 random queries set and 500 first queries from the dev set. Testing is done on a relatively small set of queries compared to the official evaluation set containing 5,793 query's, our result is therefore not directly comparable with the official score. We do not have enough data to prove that our baseline actually has a better accuracy then the Indri model. The official baseline for the evaluation set is 0.192 MRR@100 using the Indri model, the best performing models in this set has a score of 0.329 [9].

4.0.2 Advanced model. Our advanced model also consistently slightly outperforms both the Indri model and the baseline model on the training sets we tested on, both the 260 random queries set and 500 first queries from the dev set.

In addition to adding features we experimented with normalizing the feature vectors based on min-max for each query. The min max scaling is done to make the feature vectors less query dependent. Experimentation with this led to terrible performance(MRR@100<0.05) and was therefore not used in the final advanced model. We suspect the low performance obtained with min max scaling is due to the way we construct our training-set. For each query we had to calculate feature vectors for all 100 retrieved documents, even though we only use a few documents for each query.

Parameter tuning of the random forest regression model and experimentation with other learning to rank models was minimal. This was in part because our inefficient feature extraction made it very time consuming to train and test models.

There is a lot that could be done to improve the efficiency of the feature extraction, but it will most likely be significantly slower than feature extraction used on existing language models (like Indri) and vector space models. The result only shows marginal improvement over significantly faster and 'simpler' methods and are not even close to the performance of more advanced models. For these reasons, the models implemented in this project will not be considered useful in any practical application.

5 CONCLUSION

In this project we have implemented a model for re-ranking the top 100 most relevant documents from the MS MARCO dataset. We have done so by implementing a baseline and an advanced model to tackle this problem. Both our models slightly perform better on the evaluation metrics when it comes to the tests we've done, as you can see in the results section. These results however are made on a very small sample of the dataset. The reason behind our small sample sizes is because our models, and especially our advanced model, takes a very long time to run compared to the Indri model we're re-ranking. Therefore we can say that our models don't have a very practical use for bigger datasets like the MS MARCO one, since the run time would be too long.

REFERENCES

[1] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. 2018. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. arXiv:cs.CL/1611.09268
[2] Krisztian Balog. 2020. Assignment 5: Learning to rank. https://github.com/uis-dat640-fall2020/
[3] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. 2020. Overview of the TREC 2019 deep learning track. arXiv:cs.IR/2003.07820
[4] elastic elastic. 2020. Kstem token filter: Elasticsearch. https://www.elastic.co/guide/en/elasticsearch/reference/current/analysis-kstem-tokenfilter.html
[5] elastic elastic. 2020. Lowercase token filter: Elasticsearch. https:https://www.elastic.co/guide/en/elasticsearch/reference/current/analysis-lowercase-tokenfilter.html
[6] elastic elastic. 2020. Stop token filter: Elasticsearch Reference [7.10]. https://www.elastic.co/guide/en/elasticsearch/reference/current/analysis-stop-tokenfilter.html
[7] Kechit Goyal. 2020. Machine Learning vs Neural Networks: What is the Difference? https://www.upgrad.com/blog/machine-learning-vs-neural-networks/
[8] Matthew Mayo. 2018. The Main Approaches to Natural Language Processing Tasks. https://www.kdnuggets.com/2018/10/main-approaches-natural-language-processing-tasks.html
[9] MS MS. 2019. MS MARCO git. https://microsoft.github.io/msmarco/.
[10] MS MS. 2020. MSMARCO-Document-Ranking git. https://github.com/microsoft/MSMARCO-Document-Ranking
[11] Chenyan Xiong Zhiyuan Liu and Maosong Sun. 2020. Openmatch. https://github.com/thunlp/OpenMatch