MS MARCO Document re-ranking

Asahi Cantu
asahicantu@outlook.com
University of Stavanger
Stavanger, Norway

Shaon Rahman
sha_on@yahoo.com
University of Stavanger
Stavanger, Norway

1 Introduction

Information Retrieval and text mining are very important branches in the field of Computer Science and Artificial Intelligence. In the latest decades and with the continuous digestion of big datasets, important discoveries and models have been created to have better and more accurate systems capable of delivering the required information among millions of sources.

This report aims to contribute to the analysis of big data sets and the information retrieval models in the field of "document-re-ranking" and "Learning To Rank (LTR)", provide a detailed analysis of the retrieved data and how by employing machine learning and deep learning algorithms for document re-ranking it is possible to be more accurate towards the intended information to be retrieved. The main purpose of such task is to create systems intelligent enough by helping in real world tasks to deliver the right information by a previous analysis of human written questions and return the most relevant information.

The analysis takes place in two different tasks:

- **Document re-ranking with a base method**. Common machine learning regression models are applied to known retrieved documents, and then compared against original ranking BM25 method to prove the efficiency of such models towards expected values.
- **Document re-ranking with an advanced method**. A deep learning method (BERT) is used to re-rank the given documents and compared against the developed models.

In each task the results are displayed, interpreted and compared one against the other. Based on those results discussion and conclusions are presented towards the efficiency and usability of such models for real life applications.

2 Problem Statement

2.1 Scope of work

Implement a baseline and advance methods for document re-ranking for a corpus dataset called MS-MARCO[1] provided by Microsoft where a set of 100,000 real questions were processed by Microsoft's search engine Bing™[1]. The dataset brings already a top-100 retrieved document set per question, being each set the most relevant documents found by the search engine and a set of labeled documents, where each query contains only one most relevant document. The accuracy of the algorithms will be translated by the rank in which such relevant document is shown once the information is retrieved[2].

Once a traditional algorithm BM25 is implemented for document retrieval, two additional methods will be implemented to evaluate the results for document re-ranking and their accuracy by employing:

- Baseline Method. A general experiment run over different common Machine Learning models:
  - Linear Regression
  - XGBoost
  - Random Forest
- Advanced Method. An deep learning model implementation using BERT[3]

Both methodologies will be compared against the traditional and shown in the Results section.

2.2 Dataset Specifications

2.2.1 Name. MSMARCO Document Ranking (Microsoft Machine Reading Comprehension).

2.2.2 Decription: Large scale dataset focused on machine reading comprehension, question answering, and passage ranking, Keyphrase Extraction, and Conversational Search Studies. Is a dataset focused in data search related problems.The document ranking dataset is based on source documents.

2.2.3 Corpus size:

- 3.2 million documents
- 367,013 queries.

For each training query there is a positive passage ID to the corresponding document ID in the corpus. It is assumed that a document that produced a relevant passage is usually a relevant document. **Document Source (URL:)**

- https://github.com/microsoft/MSMARCO-Document-Ranking
- https://microsoft.github.io/msmarco/

2.3 Software tools and packages

Employed software tools for the development of this project were:

[1]Microsoft Bing™is a web search engine owned and operated by Microsoft. The service has its origins in Microsoft's previous search engines: MSN Search, Windows Live Search and later Live Search. Bing™provides a variety of search services, including web, video, image and map search product

Table 1. Dataset Details

#	Type	Name	Size
1	Corpus	msmarco-docs.tsv	22 GB
2	Train	msmarco-doctrain-queries.tsv	15 MB
3	Dev	msmarco-docdev-queries.tsv	216 KB
4	Dev	msmarco-docdev-top100	27 MB
5	Dev	msmarco-docdev-qrels.tsv	112 KB

Table 2. Dataset Details - Records

#	Records	Format
1	3,213,835	tsv: docid, url, title, body
2	3,213,835	tsv: docid, offset_trec, offset _tsv
3	5,193	tsv: qid, query
4	5,193	TREC submission: qid, "Q0", docid, rank, score, runstring
5	5,478	TREC qrels format

2.3.1 Elastic Search™. Search engine based on the Lucene library. Provides a distributed, multitenant-capable full-text search engine with an HTTP web interface and schema-free JSON documents. This search engine retrieves documents by a provided query using BM25 algorithm[2]. All the dataset was document by document into an Elastic Search instance and later on used for the implementation of baseline and advance methodologies.

2.3.2 Python. Used for code development, and Jupyter Notebooks for experimentation and implementation of the methods. Used Packages:

- **Scikit Learn** and **XGBoost** for the baseline method
- **Pytorch** and **Bert Sentence Transformers** for the Advanced method[4][5]

2.4 Data Ingestion Process

A 10% sample of the corpus and queries was taken to perform the experiments once data was indexed,prepared,and ready to be processed in *ElasticSearch*.

Since the dataset already contained the top-100 documents retrieved by queries, after a data analysis was found that just 490,601 out of 3,213,835 documents were required to index from the corpus, diminishing with this the amount of time required to load the whole corpus from up to 12 hours to just 3.5 hours.

[2]BM25 (Best Matching 25) is a ranking function used by search engines to estimate the relevance of documents to a given search query. It is based on the probabilistic retrieval framework developed in the 1970s and 1980s by Stephen E. Robertson, Karen Spärck Jones, and others. Represent state-of-the-art TF-IDF-like retrieval functions used in document retrieval.

2.4.1 Data preparation. Extracted data from the corpus was indexed using ElasticSearch software for later usage on the baseline and advanced methods.

To implement the baseline method sample data was divided in training data and test data.

Table 3. Dataset Sample 10%

#	Data type	Records
1	queries	519
2	qrels(query-relevan document)	519
3	documents	49601

Table 4. Sample data preparation

Data type	Percentage
Training data	80%
Test data	10%

3 Baseline Method

The following steps were employed:

3.1 Feature extraction

Features are extracted from a query and documents stored in *ElasticSearch*. Relevant features are:

- Query
  - Query length
  - Sum of query term's IDF[3] in the corpus
  - Maximum among query terms IDF in the corpus
  - Average of query terms IDF in the corpus
- Document
  - Length of body
  - Length of title
- Document-Query pair
  - Number of unique query terms in document title
  - Number of unique query terms in document body
  - Sum, Max, Average, value for document title and body's term frequency in the corpus

The given query and body of documents are used to construct a feature vector.

3.2 Training

From given 10% sample data, 519 query-document pair elements were taken from the file *'msmarco-docdev-qrels.tsv'*. This document query pairs are considered our training set. Since this file is the ground truth and every document is relevant to the corresponding query, documents in the mentioned file was labeled for its target vector value as 1, for the

[3]Inverse Document Frequency

feature vectors generated from the document query pair. Non relevant documents were labeled as *'0'* and extracted from *ElasticSearch* by getting top 100 documents for each query and generating feature vector from the document-query pair. If *ElasticSearch* returns the a relevant document, it is ignored. Therefore there are 519 document query pairs. For each relevant document 99 irrelevant documents are retrieved. Every one of them has its own feature vector generated to make a total of 51900 feature vectors.

3.3 Document re-ranking

Document re-ranking was performed by taking document-query pairs from *'msmarco-docdev-top100.tsv'*. In this file there is top-100 pre- ranked documents for every query. Queries that are used for training in the machine learning models are excluded. After exclusion, features for every query-document pair are extracted and used on each ML model to predict the relevancy (get a value between 0 and 1 for every query document pair). Finally documents are sorted from high to low predicted values to get re-ranked documents.

Among several available regression models *XGBoost* and *RandomForest* proved to be optimal approaches for document re-ranking problems[9]. The *Random Forest* model was also chosen based on the results obtained from previous experiments with IR[4] and document re-ranking.

3.3.1 Random Forest. Parameters: max_depth=7, random _state=0, n _jobs=4, n_estimators=19

3.3.2 XGBoost. Parameters: max_depth=3, random_state=0, objective='reg:linear', verbosity=0, n_estimators=11

3.3.3 Linear Regression. Parameters: normalize=True

4 Advanced Method

The advanced method was performed using BERT[3][5] with the sentence-level retrieval approach, performing a semantic search and S-BERT[6] algorithms. This is achieved by creating word embeddings (pre-computed document representation) for the test corpus and queries. Then BERT model compares the query to all existing sentences (for a dataset of 'n' sentences would require 'n' comparisons). Finally the measurement of document 'closeness' to specific query is performed by finding the cosine similarity.[7]. The base ranking dataset was passed to a BERT Sentence transformer model with a train language model called DistilBERT[8]. The training process for text summarizing and word embedding took 23 hours with 1 GPU[6] running in a cloud based system Kaggle[7].

4.1 Feature Extraction

Test Data was preprocessed using the following algorithms:

1. Bert-extractive-summarizer[4]. Is a python package used to embed the sentences, and run a clustering algorithm to finding the sentences that are closest to the cluster's centroids. With this approach it is possible to reduce the document size (more than 2000 words per document) allowing to reduce the time and work over more consistent documents.
2. Sentence Transformers[5]. Python framework which computes dense vector representations for sentences and paragraphs (sentence embeddings). The models are based on transformer networks (BERT among others). Having parsed the text trough summarazing method, each document is tuned by specificially meaningul sentence embeddings such that sentences with similar meanings are close in vector space.

4.2 Model Evaluation

Feature vectors are evaluated by extracting their cosine similarity a the metric used to measure how similar the documents relative to the given query. Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space. The smaller the angle, higher the cosine similarity.

5 Results

Results for each model were taken by using Mean Reciprocal Rank Score Metric over top 100 occurrences *MRR@100*.

$$MRR = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{rank_i}$$

Where $rank_i$ refers to the rank position of the first relevant document for the i-th query. The reciprocal value of the mean reciprocal rank corresponds to the harmonic mean of the ranks.

This results were found by using the testing queries over several iteration through the different models. The parameters for those models were adjusted and tuned accordingly as the observations showed better ranking scores over the iterations.

Tthe base ranking queries were used with their related documents per query. After performing *BERT* summarizing and and Sentence Tokenizer-Transformer, the results were scored by a cosine similarity from a given query. Tuning the advanced model was performed by adjusting the length

[4]Information Retrieval

[5]BERT stands for Bidirectional Encoder Representations from Transformers. Is a state of the art language representation model built, developed and pre-trained by Google.

[6]Graphics Processing Unit

[7]Kaggle provides computational resources to perform machine learning operations with mid to high computational resources to facilitate any user research and model ML and deep learning algorithms for free

of summarized text, observing under which conditions the ranking score was the best. The documents are ranked by the similarity to the query and then the result scored using *MRR@100*.

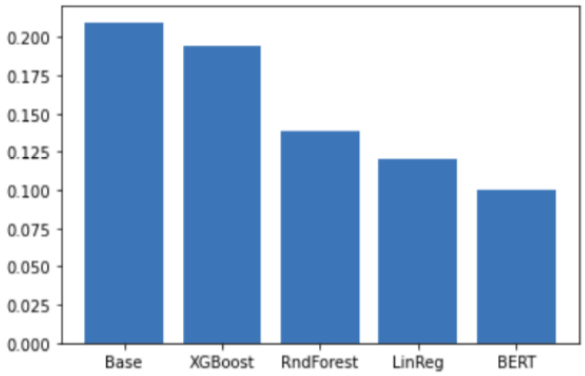

Figure 1. MRR@100*results*.

Table 5. MRR@100 Results

Model	MRR@100
Base	0.2097
XGBoost	0.1938
RndForest	0.1382
LinReg	0.1201
BERT	0.0996

6 Discussion and Conclusions

During the experiments, observed results fell below the standard BM25 algorithm due to the size of the sample dataset and the resources available for the training experiments. Different literature for similar works over different algorithms have demonstrates to perform better as the training and testing datasets increase in size or take the whole corpus for evaluation. Text mining and information retrieval require several experiments to fine tune and check the best ML algorithm for such problem. Document re-ranking can be improved at the expense of time and high computational resources.

On the other hand and according to [6], the best performance for the advanced model in document re-ranking is achieved once BERT model is pre-trained with the whole corpus. This approach was not possible to perform with the current computational resources available in considerable time, since the corpus is massive.

Considering that only 10% of the entire set was taken to perform all the analyzes, tt is expected that by increasing the sample size the results tend to be higher than BM25. Further research in other ML algorithms (such as Support Vector Machines, or Naive Bayes approach) could be developed for the baseline method in order to find a more powerful re-ranking algorithm.

As for the advanced method, the usage of a large BERT model and the possibility of eliminating the text summarizing could help to increase the ranking level at the expense of higher computing power and time to make it happen. It was observed also from [1] that the leader-board in research and experiments for this dataset re-ranking, similar approaches to the one is used in this document presented higher scores when the full dataset is considered.

Performing re-ranking tasks over big datasets is not a trivial. In many cases requires advanced understanding on the deep learning methodologies to make tasks more efficient than with conventional ML algorithms. BERT is in this context a game changer for information retrieval and text mining. Without a doubt these new models take over in the near future for the numerous developments over information retrieval. Massive datasets and the disposition of technologies allow to experiment, tune and create new models which can help the industry and companies to improve search engines and text mining technologies.

A full implementation of the work is available in a Kaggle Notebook https://www.kaggle.com/asahicantu/dat640-final-project-ms-marco.

To test generated results and get the developed models the GitHub repository is available. https://github.com/asahicantu/DAT640-FINAL-PROJECT

References

[1] Microsoft Corporation. 2020. *MS MARCO.* Retrieved Nov 3, 2020 from https://microsoft.github.io/msmarco/

[2] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M Voorhees. 2020. Overview of the trec 2019 deep learning track. *arXiv preprint arXiv:2003.07820* (2020).

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018). https://arxiv.org/abs/1810.04805

[4] Derek Miller. 2020. *Bert Extractive Summarizer.* Retrieved Nov 3, 2020 from https://pypi.org/project/bert-extractive-summarizer/

[5] Derek Miller. 2020. *Sentence Transformers: Multilingual Sentence Embeddings using BERT / RoBERTa / XLM-RoBERTa Co. with PyTorch.* Retrieved Nov 3, 2020 from https://pypi.org/project/sentence-transformers/

[6] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing.* Association for Computational Linguistics. https://arxiv.org/abs/1908.10084

[7] Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084* (2019). https://arxiv.org/abs/1908.10084

[8] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108* (2019). https://arxiv.org/abs/1910.01108

[9] Libin Shen and Aravind K Joshi. 2005. Ranking and reranking with perceptron. *Machine Learning* 60, 1-3 (2005), 73–96.