SIB Text Mining at TREC 2019 Deep Learning Track: Working Note

Julien Knafou[a,c], Matt Jeffryes[a,c], Luc Mottin[a,c], Douglas Teodoro[b,c], Patrick Ruch[a,c]

[a] *HES-SO / HEG Geneva, Information Sciences, Geneva, Switzerland*
[b] *HES-SO / HEG Geneva, Business Information Systems, Geneva, Switzerland*
[c] *SIB Text Mining, Swiss Institute of Bioinformatics, Geneva, Switzerland*
contact: {julien.knafou}@hesge.ch

Abstract

The TREC 2019 Deep Learning task aims at studying information retrieval in a large training data regime. It includes two tasks: the document ranking task (1) and the passage ranking task (2). Both of these tasks had a full ranking (a) and reranking (b) subtasks. The SIB Text Mining group participated at the full document ranking subtask (1a). In order to retrieve pertinent documents in the 3.2 million documents corpus, our strategy was two-fold. At first, we used a BM25 model to retrieve a subset of documents relevant to a query. We also tried to improve recall by using query expansion. The second step consisted in reranking the retrieved subset using an original model, so-called query2doc. This model, which has been designed to predict if a query-document pair was a good candidate to be ranked in position #1, was trained using the training dataset provided for the task. Our baseline, which is basically a BM25 ranking performed the best and achieve a MAP of 0.2892. Results of the query2doc run clearly indicates that the query2doc model could not learn any meaningful relationship. More precisely, to explain such a failure, we hypothesize that using documents returned by our baseline model as negative items confused our model. As future steps, it will be interesting to take into account features such as the document's BM25 score as well as the number of times a document's URL is mentioned in the corpus and use them along with learning to rank algorithms.

Introduction

The SIB Text Mining group [1], at the Swiss Institute of Bioinformatics in Geneva, has a long history of participation in TREC campaigns, including TREC Genomics [2], TREC Medical Records [3], TREC Chemical IR [4] and TREC Clinical Decision Support [5,6] tracks. The first iteration of the Deep Learning track was an opportunity for us to evaluate some machine learning tools on a real-world scenario.

Since the turn of the millennium, Information Retrieval (IR) systems had to face an influx of available information. The question arises of how to efficiently and accurately find the right answer through billions of documents when the users' queries may contain only a few words. In very specific domains, the context may be used to support the search engine (e.g. by

implementing related features in the IR component or by adding constraints to the initial query). However, with no context definition, machine learning strategies appears as a good alternative to the traditional approaches.

This year, the TREC 2019 Deep Learning track proposed an extensive amount of query-document mappings for the training of search engines. Such a dataset was a challenge for the development of various models from scratch, and this year we focused on the document ranking task with a preeminent strategy based on recurrent neural networks (RNN).

1. Data

The full ranking (retrieval) subtask of the deep learning track's document ranking task provides the MS-MARCO dataset which gathers about 3.2 million documents [7]. Each document contains a URL, a title and a body. Along with these documents, three sets of queries were provided. A training and development set which map 367,013 and 5,193 queries respectively to a MS-MARCO document and a testing set of 200 queries (without mapping), which was provided later in the campaign.

2. Strategies

To deal with the large number of documents, our strategy was two-fold. We first used traditional methods to create subsets of relevant documents (1) which allowed us to use computationally intensive methods to refine the retrieved documents selection (2). In other words, in (1) our idea is to return a thousand documents while maximizing the recall score, while in (2), we use a document reranking model to improve the precision of our predictions on the subset returned in (1).

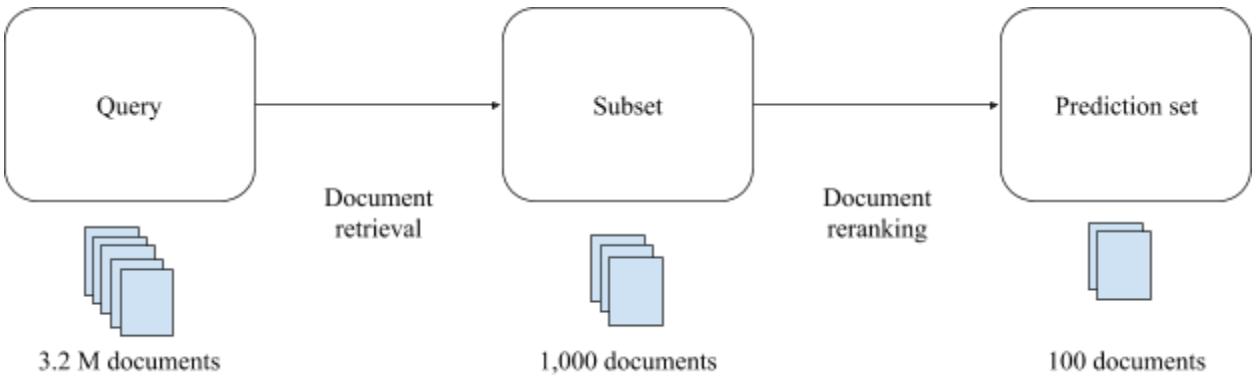

Figure 1 Strategies graphical representation

Methods used to create the first subset (1) are detailed in the next section under Document Retrieval, while the methods used to refine subsets (2) will be described later in the Document Reranking section.

2.1 Document Retrieval

2.1.1 BM25 retrieval

For the initial retrieval, the documents were inserted into an Elasticsearch cluster [8]. The document ID, URL, title and body text of the documents were indexed. Retrieval from Elasticsearch can be adjusted with various options and parameters. We tested the performance BM25 retrieval module across a range of parameters (see figure 2).

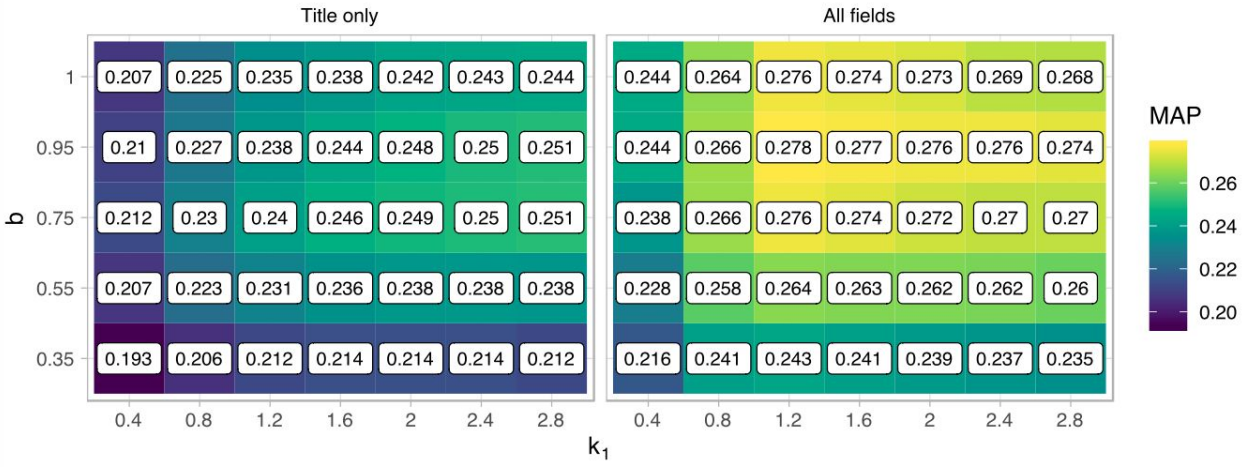

Figure 2 BM25 fine tuning using MAP as metric. In the left panel are the results for scoring based on the title only and in the right panel are the results using all fields. Along the *x* axis, the value for k_1 is varied, and along the *y* axis the value for *b* is varied.

The k_1 variable affects how much a single query term can contribute to the score [9]. That is, how many occurrences in the document maximises the possible score for a term. The *b* variable affects how much the document length is penalised, with *0* not penalising long documents at all, and *1* being the maximum. In addition, we tested whether it was beneficial in this task to score documents using only the title text or to use all fields. Using this performance tuning, we identified the optimal parameters for the BM25 module to be $b=0.95$ and $k_1=1.2$. With these parameters, recall at 1000 was 0.913.

2.1.2 Query Expansion

As the preselection was an important starting point for the second part of our strategy, we tried to boost the BM25 performances by using query expansion [10]. Unfortunately, query expansion did not improve the recall score in our development (at least for the size of the subset we were returning) and was not used for the first step for our two runs. We tried two ways of expanding

queries. The first method replaces words in a query with other words, which are related in a language model, the second method translates queries in a foreign language and translate it back to the original language.

2.1.2.1 Word2Vec similarity replacement

We used different pre-trained Word2Vec models [11] found in [12] and replace words in a query that where exceeding a certain threshold of similarity using the cosine distances as a metric. This method allows us to retrieve documents [13] which does not necessarily contain the query's words.

2.1.2.1 Back-Translation

Back-translation is a way to paraphrase entire sentences without having to look for monolingual parallel corpora [14]. Unlike the previous method using Word2Vec similarities, where words are replaced one by one, this method generates a paraphrase from a translation of the original sentence. We used the pretrained model described in [15] to generate translations in both German and Russian and translate them back to English.

2.1.3 Query to documents distances

We used doc2vec [16] and computed the vector representation for both the queries and each of the 3.2 million documents. We then computed, for each query, the cosine distances between the query vector and the document vector. The 1,000 closest documents were used as candidates.

2.2 Document reranking

This part details the two methods we used in order to narrow the 1,000 document selection down to 100 documents.

2.2.1 Classification (query2doc RNN)

We developed a model which takes a query followed by a document's content (see figure 3) as an input and trained it over the 367,013 queries in the training set. We thought it would be more appropriate to encode the queries and documents in two different recurrent neural networks (RNN) because of the amount of words which is quite different in average between documents and queries. Both of these RNNs output a vector that is concatenated in order to feed a neural network (NN). Thanks to gradient optimization methods, this architecture allows us to train the model at once instead of creating three different models (i.e. one for encoding queries, one for encoding documents and one for predicting if the query-document pair encoding is a good candidate) and training them separately. We used Nestrov Momentum in our optimization process, more precisely an approximation of [17] which is implemented in TensorFlow [18].

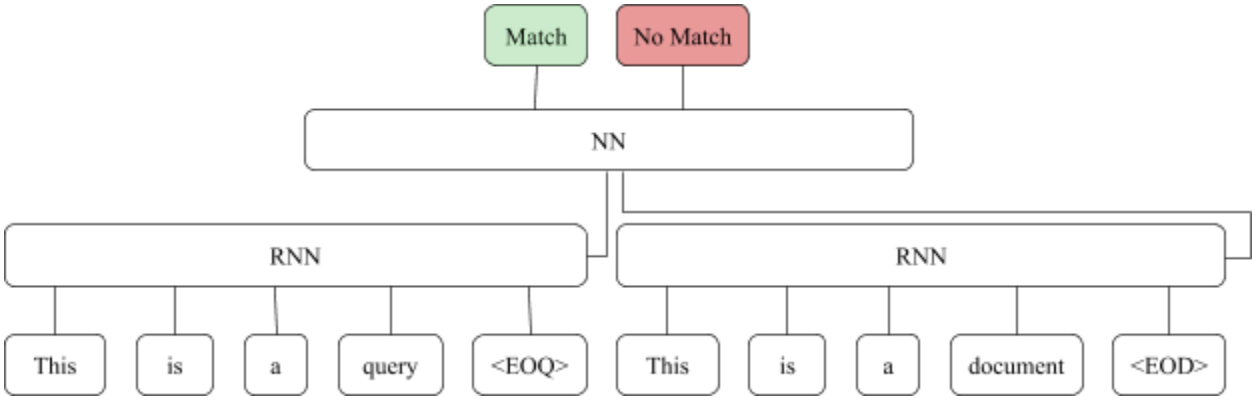

Figure 3 Model graphical representation

Documents that were retrieved according to our initial retrieval methods were used as negative observations when they were not labelled as positive in the dataset. For the inference, we took documents that were returned in the initial retrieval part and take the score that our model would allocate to each query-document pair. For each query, the 100 documents with the highest scores were then returned. We called this model the query2doc RNN which corresponds to our first submitted run.

2.2.2 Word Mover Distance (WMD)

Even if we submit only the previous method in our runs (apart from the baseline), we also used WMD [19] to refine the selection. We were unable to complete our testing of this method by the deadline and will aim to test the results of this method once the qrel of the testing set is released.

3. Results & Discussion

In table 1, we can see results for our two runs. The baseline model used the BM25 retrieval method returning 1,000 documents (2.1.1) whereas the query2doc run was a baseline subset of 100 documents using query2doc method (2.2.1).

Run name	**MAP**	**recall_1000**	**NDCG**
Baseline	0.2892	0.6383	0.5563
query2doc	0.0111	0.0697	0.0643

Table 1 Principal metrics for our two runs

It is obvious that the query2doc method worked poorly. Our hypothesis for this poor performance lies in the fact that we were using documents returned by our baseline model that were not labeled in the dataset as negative observations which confused the discriminative power of our model. Indeed, some of the documents labelled as negative could have been as pertinent as a positive one content-wise. In other words, the relationship between the negative observation and the query-document pair was ambiguous and thus, our model could not make sense of it.

This is why it would be interesting to see if the WMD (2.2.2) method, which is unsupervised, would have worked better. We also think that the aggregation of other features would have significantly improved the reranking (e.g. the document's BM25 score, the number of times a document's URL is mentioned in the corpus, etc.) and in future editions, we will test the efficiency of learning to rank algorithms in such a task.

Finally, we did not try to change the size of our subsets, which could have had an impact on our metrics. In our next experiments, we will use our best performing methods of document retrieval on larger sets (e.g. 2,000; 5,000; 10,000) and use learning to rank methods to refine the selection. As query expansion has not been used in our runs, it would also be interesting to see how it performs for retrieving a larger set.

Conclusion

Our experiment allowed us to step into the first iteration of the TREC Deep Learning track. Although the results of our main run are quite poor, it has given us insights about how to approach these problems in the future. Some aspects of our approach were more successful than the final results would imply. Our initial retrieval using Elasticsearch was very simple but

effective, and drastically reduced the number of candidates which needed to be processed using the RNN. As noted in the previous section, we think that reranking should have been tackled differently and we will certainly rethink our strategy for the next iteration of this track.

References

[1] http://bitem.hesge.ch/
[2] Gobeill J., Tbahriti I., Ehrler F. and Ruch P., (2007). Vocabulary-Driven Passage Retrieval for Question-Answering in Genomics. In TREC. 2007.
[3] Gobeill J., Gaudinat A., Pasche E., Teodoro D., Vishnyakova D. and Ruch P., (2011). BiTeM Group Report for TREC Medical Records Track 2011. In TREC. 2011.
[4] Gobeill J., Gaudinat A., Pasche E., Teodoro D., Vishnyakova D. and Ruch P., (2011) BiTeM group report for TREC Chemical IR Track 2011. In TREC. 2011.
[5] Gobeill J., Gaudinat A., Pasche E. and Ruch P., (2014) Full-texts representation with Medical Subject Headings and co-citations network reranking strategies for TREC 2014 Clinical Decision Support Track. In TREC. 2014.
[6] Gobeill J., Gaudinat A., P Ruch., (2015) Exploiting incoming and outgoing citations for improving Information Retrieval in the TREC 2015 Clinical Decision Support Track. In TREC. 2015.
[7] Bajaj P., Campos D., Craswell N. et al. (2016) MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. arXiv:1611.09268.
[8] Elasticsearch 6.2.4 https://www.elastic.co/downloads/past-releases/elasticsearch-6-2-4
[9] Robertson S., Walker S., Jones S., Hancock-Beaulieu M. M. and Gatford M., (1995) Okapi at TREC-3. In Overview of the Third Text REtrieval Conference (TREC-3).
[10] He B. and Iadh O. (2009) Studying Query Expansion Effectiveness. Proceedings of the 31th European Conference on IR Research on Advances in Information Retrieval, in Advances in Information Retrieval, 611-619. DOI: 10.1007/978-3-642-00958-7_57.
[11] Mikolov T. et al. (2013) Efficient Estimation of Word Representations in Vector Space
[12] https://github.com/RaRe-Technologies/gensim-data
[13] Aklouche B., Bounhas I. and Slimani Y., (2018) Query Expansion based on NLP and Word Embeddings. In TREC 2018.
[14] Bannard C. and Callison-Burch C. (2005) Paraphrasing with bilingual Parallel Corpora. In ACL 2005, 43rd Annual Meeting of the Association for Computational Linguistics.
[15] https://github.com/pytorch/fairseq/tree/master/examples/wmt19
[16] Le Q. V. and Mikolov T., (2014) Distributed Representations of Sentences and Documents
[17] Sutskever I., Martens J., Dahl G. and Hinton G. (2013) On the importance of initialization and momentum in deep learning. Proceedings of the 30th International Conference on Machine Learning, PMLR 28(3):1139-1147, 2013.
[18] https://www.tensorflow.org/api_docs/python/tf/compat/v1/train/MomentumOptimizer
[19] Kusner M.J., Sun Y., Kolkin N.I. and Weinberger K.Q. (2015). From word embeddings to document distances. ICML'15 Proceedings of the 32nd International Conference on International Conference on Machine Learning, 37, 957-966.