Neural Information Retrieval 2022 Report

Zi Wang ***zrw348***
University of Copenhagen

Abstract

This is report for the NIR2022 project. In this project, I implemented conventional information retrieval pipeline based on various models. Specifically, I explored how different indexing approaches, retrieval models, and relevant strategies impact the final performance of IR system through experiments. I used Pyterrier as the main framework for implementation and the TREC 2004 Robust(Voorhees, 2005) dataset as the benchmark for experiments.

1 Introduction

In this project, I implemented the pipeline of information retrieval system with Pyterrier. This report is structured basically following this pipeline, involving data pre-processing and indexing, ranking and re-ranking model construction and evaluation. The relevant details are also discussed and analyzed in this report. Furthermore, I involved review and future work regarding how the methods can be improved.

2 Pre-processing & Indexing

2.1 Dataset

Table 1 shows the statistics of the dataset used in this project. The TREC 2004 Robost dataset mainly contains three parts: Query, qrel and document.

Query is the input of retrieval system, consisting of *qid* and *query* text. Since the whole dataset is split into training and test set based on queries, we have 200 queries for training and 50 unseen queries for test. Most query texts are short phrases with an average length of 2.75.

Document is a collection of newswire documents involving *title* and *text*. One field of *docno* is used as the ID of documents. From the statistics, we notice that there are 528155 documents with an average of around 275 words. The large scale of document collection and relatively long text make the dataset more realistic and challenging.

Qrel contains the ground truth of each query. Each qrel has 3 fields (for this project) indicating query *qid*, document *docno* and the assessed relevance or label between them. The statistics in training set show that there are 247569 assessed query-document pairs, and for each query, there are averagely 199 documents assessed.

TREC 2004 Robust		
Query (train)	Number	200
	Average Length	2.75
Qrel (train)	Number	247569
	Number per Topic	199
Document	Number	528155
	Average Length	275.6

Table 1: Dataset Statistics

2.2 Indexing Approach

Pyterrier provides an efficient pipeline to complete basic text pre-processing and indexing. Within the framework of Pyterrier, I primarily compared different stemming methods and how they impact final indexes. Specifically, I set one indexing pipeline *NNS* without any pre-processing except tokenization, and three pipelines with stop words removal and different stemming methods including no stemming (*NS*), Porter stemming (*PS*) and snowball stemming (*SS*). The final indexes are then generated based on the processed documents. The statistics of 4 resulting indexes are shown in Table 2.

We notice that all the indexing pipelines process all 528K documents. However, different strategies largely influenced the efficiency both in memory and time. The NNS which introduced the stop words as indexes occupies significantly larger storage than the other 3 index sets and consumes more time. NS is also less efficient because the index terms involves many variants without stemming or lemmatization. For the other two index sets, they are close in both term and posting counts although

		NNS	NS	PS	SS
Count	Document	528K	528K	528K	528K
	Term	622K	621K	520K	521K
	Posting	117.8M	90.4M	83.7M	83.6M
	Token	257M	145M	145M	145M
Efficiency	Storage	2.46 GB	1.97 GB	1.90 GB	1.90 GB
	Time	13m 14s	12m 21s	10m 37s	11m 50s

Table 2: Index Statistics. The processing strategies: NNS-Without stop words removal and stemming; NS-Without stemming; PS-With Porter stemming; SS-With snowball stemming. The count numbers are approximated, where M refers to million and K refers to thousand.

the stemming strategies cause slight difference, and the snowball stemming appears to consume more time.

Further evaluation was also conducted to select the optimal index, and the results are integrated into the following ranking model tuning section.

3 Ranking Model

This section integrates the preliminary evaluation on all training topics, involving comparison of different indexes mentioned above. For specific ranking model selection, BM25 and Dirichlet Language Model were selected to be optimized after simple comparison with similar models. The tuning process largely depends on the *KFoldgridsearch* provided by Pyterrier, which applies cross validation over the split K-fold subset based on mean average precision (MAP). In this project, the training topics are split into 3 folds thus the resulting size of single fold is close to test set.

3.1 BM25

For BM25, tunable parameters include b and $k1$, where b normalize the length of documents and $k1$ penalize terms with high term frequency. While b and $k1$ were set as 0.75 and 1.2 in previous work empirically, the k-fold search turned out the combination of b=0.3 and $k1$=0.8 achieved best performance on this dataset. *GridScan* was also applied to measure BM25 models with different parameter combinations on all training topics, and the results was further visualized in Figure 1, with which we can obtain the same optimal parameters as *KFoldGridSearch*.

3.2 DLM

Dirichlet Language Model was selected as another model to tune. Typical language models built for information retrieval rank the documents based on

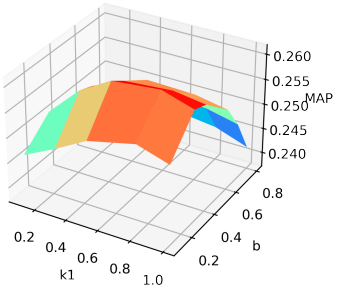

Figure 1: BM25 MAP versus parameters

their probability of generating the target query. For DLM, a Bayesian smoothing with Dirichlet Prior was applied on LM, and μ in this model is tunable regarding specific collection. As stated in the paper(Zhai and Lafferty, 2004), the μ value is thought to vary from collection to collection, and an empirical setting is μ =average document length .

Similarly with BM25 tuning, cross validation was conducted on DLM to tune the μ. It turned out that μ around 275 performed well.

3.3 Index Comparison

Model	Index	MAP	nDCG@10
BM25	NNS	0.230	0.422
	NS	0.231	0.424
	PS	0.260	0.448
	SS	**0.261**	**0.450**
DLM	NNS	0.223	0.418
	NS	0.218	0.410
	PS	0.238	0.422
	SS	0.240	0.424

Table 3: Index performance comparison.

The generated indexes mentioned in section 2

are compared based on the two optimal models. As shown in Table 3, the combination of BM25+SS achieved the best performance while DLM+NS is the least effective combination. It is worth to notice that NNS involving stop words outperformed NS on DLM, suggesting that the stop words may slightly contribute to the language model prediction.

Comparing indexes with and without stemming, we can see that stemming significantly improved retrieval performance. As for the two stemming strategies, PS appears to slightly outperformed SS, indicating that snowball stemmer reduced semantically relevant word variants to the same base more precisely than Porter stemmer.

To compare the impact of ranking models and indexing approaches, I further evaluate several models including *BM25F*, which obtained **0.212** for MAP with SS. Thus we can see the performance gap between the best and worst models is significantly larger than that between indexes. Although indexing is as important as ranking since it is the foundation of IR, the ranking models appear to affect the overall performance more because they affect the measure scores in a larger range scale.

4 Reranking

In this part, several methods are applied to re-rank the retrieved documents, mainly including query rewriting and text embedding.

4.1 Query Rewriting

4.1.1 Word2vec

Pre-trained word embedding enables us expand the query with semantically similar terms. The *word2vec-google-news-300* provided by *gensim* was used to extend the query by adding *K* most similar words to the query. The word embedding pre-trained on news corpus was chosen to guarantee the domain gap is as small as possible. The expanded queries were then tested on BM25+SS. However, our results show that this simple strategy can lead to a fairly bad performance with MAPs around **0.13** (K=10) and **0.17** (K=3) . It appears that model tend to retrieve irrelavent documents with the newly introduced terms. A vital reason for this is that a pre-trained word embedding is used here, while the ideal setting(Kuzi et al., 2016) trains word embedding on the corresponding corpus, thus the expanded terms tend to be relevant with the documents and indexes.

To modify this method, I apply a simple weighting fusion strategy to guarantee that the wrongly retrieved documents have low scores while the expanded terms still providing meaningful information, which can be formulated as follows.

$$score_{final} = score_r + \frac{score_{bm25}}{5} \quad (1)$$

where $score_r$ is the scores obtained with expanded queries, $score_{bm25}$ is obtained with original queries. The documents are then ranked based on final score $score_{final}$. The results in Table 4 show that the fusion model W2V Fusion slightly improved BM25 performance.

4.1.2 Pseudo Relevance Feedback

PRF-based methods can better rewrite the queries based on retrieval feedback. While Pyterrier provides several PRF-based query rewriting methods, I choose two conventional methods involving *RM3* and *Bo1*. *RM3* is a prototypical model which defines language model generating new terms for the query based on its retrieved documents, and *Bo1*(Amati, 2003) further introduced Bose-Einstein distribution to expand terms by measuring their information content. Moreover for RM3, a tunable parameter α controls the importance of expanded terms and is set as **0.6** by default, which were suggested to be an optimal value on this dataset by our tuning experiment.

Table 4 shows that the two models achieved fairly close performance which obtained significant improvement compared with BM25 baseline. It is also worth to notice that the rewriting methods mentioned above are all tested with DLM, and similar with *Bo1* shown in the table, they all have negative impact on effectiveness of DLM because they destroyed the structured original query which is suitable for language model.

5 Text Similarity

Another advanced approach to handle re-ranking problem introduces contextualized embedding such as BERT to evaluate the similarity between query and document text, and re-ranks documents based on their similarity scores. In this project, *sentence-BERT* was used to compute the similarity. While the open source package provides pre-trained *bi-encoder* and *cross-encoder*, I choose the latter one since it is believed to perform better on similarity prediction(Reimers et al., 2019).

The process basically follows *retrieve_rerank_simple_wikipedia.ipynb*[1], which divides the long document into short text segments and compute similarities between query-segment pairs. Specifically, segments of 10 words are extracted at a stride of 5, thus to retain information as much as possible. As for the final scores for each document, I attempt three simple strategies, i.e. taking the maximum, mean and sum score of its corresponding segments to re-rank. As expected, the maximum strategy achieved best performance, since it preserves the most relevant segment from impact of others which are not as representative. We see the performance improvement of cross-encoder re-ranking top 1000 documents in Table 4, indicating the effectiveness of this method.

Ranking	Re-ranking	MAP	nDCG@10
DLM	-	0.240	0.424
	Bo1	0.213	0.389
BM25	-	0.261	0.450
	W2V Fusion	0.263	0.453
	Bo1	0.295	0.461
	RM3	0.295	0.460
	Cross-encoder	**0.315**	**0.497**

Table 4: Re-ranking performance comparison.

6 Evaluation

Although preliminary evaluation has been conducted on part of the models in tuning section, limited metrics are measured. In this section, I select several model combinations to evaluate with more metrics and on both training data and unseen test data.

The metrics contain *MAP* and *nDCG* at top K. The main difference between the two kind of metrics is that MAP considers only if relevant, while nDCG takes into account the relevance scores. Furthermore, different cut-offs also suggest how model will perform in realistic scenes where only top K documents are presented to users.

6.1 Training Query

As shown in Table 5, the evaluation on training queries shows that all models combinations involving re-ranking achieved better overall performance than simple models, and cross-encoder achieved the best performance in all metrics. While the performance comparisons are as discussed in previous sections, we further measure the response time of processing 200 training queries.

For the indexes comparison, we notice that both stemmed indexes are more efficient than NS, and the SS index appears to help retrieve faster than PS. It is also intuitive that all rewriting methods of RM3 and Bo1 consume more time to respond, and cross-encoder takes more than 2 hours for 200 queries, since neural-based encoder is time consuming and should predict similarity for each segment.

6.2 Test Query

Each model is evaluated based on the unseen test queries. Overall, all models performed worse than on training queries. To quantify this difference, I compute the increment of performance metrics as shown in Table 6. We notice that while all models significantly decreased in MAP, certain models especially BM25-based models relatively preserved their ability of retrieving top-5 relevant documents, and cross-encoder even gain an improvement on nDCG@5.

Although the overall decreasing in MAP indicates certain "systemic" problem, the two PRF-based methods suffer more than other models, especially compared with the other re-ranking method cross-encoder. Furthermore, the difference in Δ_{MAP} of different models suggest how well specific model resist this systemic problem.

7 Discussion

As some discussion are involved in previous sections, the following section focuses more on the test and training performance gap, especially analyzing how query and the overall query set influence the performance. Before detailing the discussion, I argue that the performance gap in this project is not simply due to over-fitting, which is a usual reason for training-test performance gap. Instead, the problem appears to be caused by the bias of distribution between training and test set.

7.1 Query Set Analysis

I pick basic BM25 and 3 re-ranking models RM3, Bo1 and cross-encoder to investigate query difference. The AP performance for each query are evaluated for each model, then the APs are sorted in descending order as shown in Figure 2. Note that the X axis does not refer to specific query, but

[1]https://github.com/UKPLab/sentence-transformers/tree/master/examples/applications/retrieve_rerank

Query	Index	Model	MAP	nDCG@5	nDCG@10	nDCG@20	Response Time
Train	NS	BM25	0.231	0.448	0.424	0.404	19.3s
	PS	BM25	0.260	0.467	0.448	0.423	17.7s
	SS	BM25	0.261	0.473	0.450	0.425	16.5s
		BM25+RM3	0.290	0.459	0.449	0.434	37.6s
		BM25+Bo1	0.295	0.472	0.461	0.445	39.3s
		DLM	0.240	0.447	0.424	0.403	**13.4s**
		Cross-encoder	**0.315**	**0.499**	**0.497**	**0.478**	2h 41m(CUDA)
Test	NS	BM25	0.217	0.456	0.429	0.401	-
	PS	BM25	0.230	0.470	0.438	0.401	-
	SS	BM25	0.230	0.470	0.438	0.401	-
		BM25+RM3	0.241	0.452	0.416	0.379	-
		BM25+Bo1	0.246	0.460	0.407	0.384	-
		DLM	0.207	0.408	0.384	0.363	-
		Cross-encoder	**0.299**	**0.518**	**0.491**	**0.469**	-

Table 5: Comprehensive evaluation results involving training queries and test queries. For response time, cross-encoder was specially tested on *google colab* with CUDA.

Index	Model	$\Delta_{\mathbf{MAP}}$	$\Delta_{\mathbf{nDCG@5}}$
NS	BM25	**-0.014**	+0.008
PS	BM25	-0.030	+0.003
SS	BM25	-0.031	-0.003
	BM25+RM3	-0.049	-0.007
	BM25+Bo1	-0.049	-0.012
	DLM	-0.033	-0.039
	Cross-encoder	-0.016	**+0.019**

Table 6: Performance difference on test query compared with on training query.

the order based on AP.

We observe that AP varies largely from query to query for all models, suggesting that queries do have different degree of difficulty, as stated in previous research(Voorhees, 2005). While RM3 and Bo1 almost overlap, the cross-encoder performed better on the tail part, i.e. the harder queries, which explains its better overall performance.

Furthermore, the hardest 10 queries for each model are picked out and shown in venn diagram as Figure 3. It is intuitive that hardest queries for RM3 and Bo1 still largely overlap since they apply similar strategy. Although we notice that hardest queries for cross-encoder are not as similar as the 2 PRF methods, five out of ten queries are still hard for all the 3 models. Intuitively, this large overlap among all models could account for the overall performance gap.

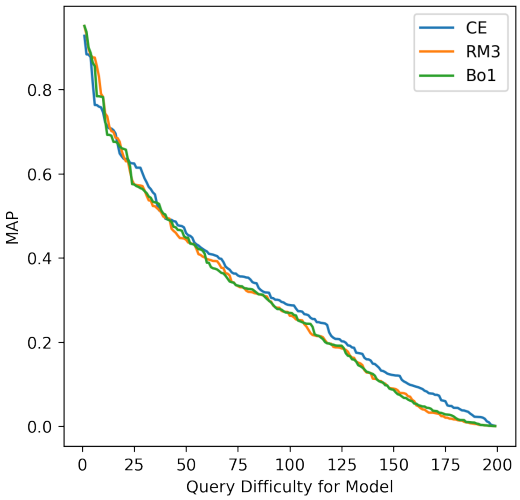

Figure 2: Average precision versus query difficulty.

7.2 Query Specific Analysis

Inspired by Figure 3, I pick up several representative queries for further analysis, specifically, queries which is not as hard for PRF model as for cross-encoder and which is hard for both models. Then it is worth to compare them with easiest queries. While several queries were analyzed, the most representative queries are selected as listed in Table 7.

For the easiest queries such as *"agoraphobia"* and *"leaning tower of pisa"*, we notice that they are more likely to be proper noun or similar phrases which lead to little ambiguity, and relevant docu-

qid	Query	MAP_{RM3}	MAP_{CE}
309	rap and crime	0.0015	0.0008
322	international art crime	0.0010	0.0114
348	agoraphobia	0.8767	0.8833
677	leaning tower of pisa	0.9432	0.9280
344	abuses of e mail	0.0095	0.3560

Table 7: Query examples. Poor performance scores are underlined. From top to bottom: 309,322: Hard query for both models; 348,677: Easy query for both models; 344: Hard query for PRF model while not for cross-encoder.

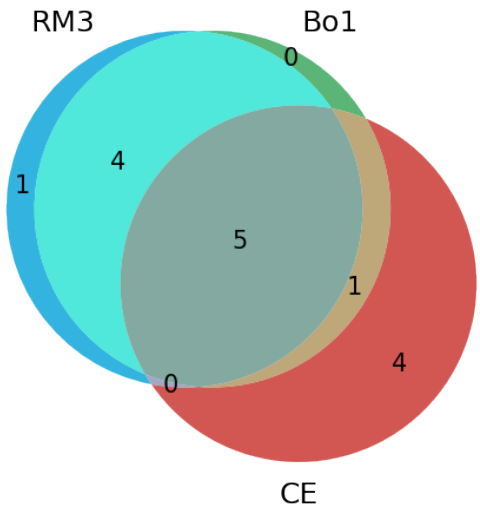

Figure 3: Caption

ments tend to mention the involved terms. As for queries poorly performed by PRF methods but not cross-encoder such as *"abuses of e mail"*, they tend to contain contextualized information which can be captured by cross-encoder. This example also suggests that text pre-processing may change or destroy original intention such as from "e-mail" to "e" and "mail", which cause ambiguity for retrieval system.

Finally, the hardest queries for both models appear to be more implicit in semantic, such as "rap and crime" and "international art crime". Take one relevant document of "rap and crime" for example, the title writes *"THE MUSIC DIDN'T MAKE THEM DO IT..."*, which requires high-level semantic cognition to understand and further build relevance with the query. Although it is believed that embedding model such as cross-encoder can handle this problem to some extend, for example, by building semantic relevance between "music" and "rap" or "rob" and "crime", they are still largely limited by the retrieval results completed by basic model at the first stage. The bad retrieval results of first stage could further influence the effectiveness of PRF-based rewriting, since they will receive more irrelevant feedback.

From all analysis above, we can see how hard and easy queries influence model performance, while hard query is not necessarily identically hard for all models. The performance gap can thus be explained that test query set involves a larger proportion of hard queries, and PRF rewriting methods deal with these queries worse, since they largely depend on feedback from the first stage, while embedding-based methods resist it better due to semantic understanding.

8 Conclusion

In this project, I implemented the pipelines of retrieval system and evaluated them on training and test queries. The results suggested performance gap between models and query set, which led to further analysis and discussion especially regarding queries. The conclusion claims that there is distribution bias of difficulty between training and test query.

However, the limitation is also obvious that MAP is focused more than other metrics such as nDCG, both in model tuning and analysis. I believe it is worth extent the analysis to more metrics to discover other latent factor. Furthermore, limited kinds of models are discussed in the final analysis, which could be taken into account following similar scheme in discussion section.

As for future work, it is worth to explore ranking methods for better understanding hard query by introducing novel NLP techniques as well as pre-processing methods for better retaining query information.

References

Giambattista Amati. 2003. *Probability models for information retrieval based on divergence from ran-*

domness Ph. D. Ph.D. thesis, thesis. University of Glasgow.

Saar Kuzi, Anna Shtok, and Oren Kurland. 2016. Query expansion using word embeddings. In *Proceedings of the 25th ACM international on conference on information and knowledge management*, pages 1929–1932.

Nils Reimers, Iryna Gurevych, Nils Reimers, Iryna Gurevych, Nandan Thakur, Nils Reimers, Johannes Daxenberger, Iryna Gurevych, Nils Reimers, Iryna Gurevych, et al. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, pages 671–688. Association for Computational Linguistics.

Ellen Voorhees. 2005. Overview of the trec 2004 robust retrieval track.

Chengxiang Zhai and John Lafferty. 2004. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems (TOIS)*, 22(2):179–214.