Experiments with Adaptive ReRanking and ColBERT-PRF: University of Glasgow Terrier Team at TREC DL 2022

Xiao Wang
University of Glasgow, UK
x.wang.8@research.gla.ac.uk

Sean MacAvaney, Craig Macdonald, Iadh Ounis
University of Glasgow, UK
firstname.lastname@glasgow.ac.uk

ABSTRACT

This paper describes our participation in the TREC 2022 Deep Learning Track. In our participation, we applied the Adaptive ReRanking technique on the constructed corpus graph from various first-stage retrieval models, namely the BM25 and SPLADE retrieval models, before applying the reranker, namely the ELECTRA reranking model. In addition, we employed the ColBERT-PRF technique on various first stage retrieval models. Finally, we experimented with ensemble retrieval for implementing both the Adaptive ReRanking and the ColBERT-PRF techniques. We submitted 14 passage ranking runs (including six baseline runs). Among the submitted runs, the run where the Adaptive ReRanking technique is applied on the ensemble of BM25 and SPLADE retrieval, namely `uogtr_e_gb`, is the most effective in terms of nDCG@10.

1 INTRODUCTION

The University of Glasgow Terrier team participated in the TREC 2022 Deep Learning track to evaluate the effectiveness of a variety of newly-proposed retrieval and ranking approaches, including ColBERT-PRF [16] and Adaptive ReRanking [8], on the large MS MARCO v2 corpus. These approaches are both designed to identify new potentially-relevant documents from the corpus, however they accomplish the task in different ways. ColBERT-PRF refines the query representation using clusters of term embeddings from the top-retrieved documents. Meanwhile, Adaptive ReRanking is applied at the re-ranking stage and iteratively retrieves documents that are close to the top-scored ones discovered by system so far.

We sought to explore the following research questions: (1) whether ColBERT-PRF can be effectively applied over a variety of initial ranking models; (2) whether Adaptive ReRanking is affected by the large number of duplicate documents in the corpus; and (3) whether it is possible to achieve competitive results without using a learned first-stage retrieval function (since they are expensive to run over a large corpus).

To answer these questions, we conducted our experiments using our PyTerrier [10, 11] information retrieval (IR) toolkit, allowing us to easily define and execute flexible retrieval pipelines. In particular, we applied our ColBERT-PRF [16] technique on several learned first-stage retrieval models, including single-representation models, such as TCT-ColBERT [6, 7] and SPLADE [3, 4], and multiple-representation dense retrieval, specifically ColBERT, as well as an ensemble of different dense retrieval models as the first-stage retrieval. We also constructed several Adaptive ReRanking pipelines over a variety of first-stage models, using both nearest neighbour information from BM25 and TCT-ColBERT.

The structure of the remainder of this paper is structured as follows: Section 2 introduces the notations of retrieval pipelines in PyTerrier. Section 3 describes the models and the frameworks we used as well as the experimental setup used in this work. Both the baseline and the group's submitted runs are detailed in Section 4. the results and analysis are presented in Section 5. Concluding remarks follow in Section 6.

2 PYTERRIER RETRIEVAL PIPELINES

All of our experiments and submitted runs for the TREC 2022 Deep Learning track are built upon PyTerrier, the expressive Python bindings for Terrier [10, 11]. In particular, in PyTerrier, all retrieval components (rankers or rerankers) to take the form of *transformer* objects, which transform one dataframe to another. To create flexible *pipelines* composed of multiple transformers, PyTerrier overloads standard Python operators for transformer objects as follows:

- » (then): Passes the output of one transformer into another.
- | (result set union): Combines the results of two transformers by selecting query-document pairs that appear in either result set.

All ranking approaches described in the rest of this paper were expressed as pipelines of transformers using these operators. We refer the reader to [11] for more information about the PyTerrier platform, the flexibility of its operators, and and its wider ecosystem of plugins for a variety of retrieval techniques implemented as transformers.

3 METHODS

In this section, we introduce the background knowledge of the graph-based Adaptive ReRanking technique in Section 3.1 and the ColBERT-PRF dense retrieval technique in Section 3.2, respectively. This is followed by the experimental setup details in Section 3.3.

3.1 Graph-based Adaptive ReRanking (GAR)

In our participation, we implemented the graph-based Adaptive ReRanking [8, 9] technique. The Adaptive ReRanking algorithm works within a pipelined cascading retrieval architecture. The Adaptive ReRanking technique leverages a pre-computed corpus graph, where the nodes represent the documents and an edge between two nodes represents a high degree of similarity between the pair of documents. More specifically, different techniques are used to construct corpus graphs, namely a graph of BM25 similarities and a TCT-ColBERT built from an Hierarchical Navigable Small World (HNSW) [12] graph. Based on such graphs, given an initial ranking pool of documents, the Adaptive ReRanking algorithm extracts the documents from the corpus that are most similar to the highest-ranked documents to improve the retrieval effectiveness.

In the following, we present one experimental pipeline using BM25 retrieval, which is then adaptively re-ranked using ELECTRA and a BM25 corpus graph:

$$BM25 \quad » \quad GAR_{BM25}(ELECTRA), \tag{1}$$

where $GAR_{BM25}(ELECTRA)$ represents the Adaptive ReRanking procedure over a BM25 graph using an *ELECTRA* scorer [14].

3.2 ColBERT-PRF

ColBERT-PRF is a pseudo-relevant feedback (PRF) mechanism, which operates entirely in the embedding space of the ColBERT model. More specifically, the stages of ColBERT-PRF [16, 17] can be summarised as follows:

(a) After obtaining the top ranked documents and their corresponding document embeddings from a first stage that deploys Approximate Nearest Neighbour (ANN) retrieval, ColBERT-PRF employs KMeans clustering to obtain the representative (centroid) embeddings.

(b) Among the representative (centroid) embeddings, to identify the most discriminative embeddings, ColBERT-PRF resorts to an approximate nearest neighbour index to obtain the nearest tokens as the most likely tokens for a given representative embedding. By doing so, the Inverse Document Frequency (IDF) of the most likely token is used as the weight of the given centroid embedding. Thus, the representative embeddings with high discriminative power are selected as the expansion embeddings to be appended to the original query representation.

(c) Finally, to control the emphasis of the expansion embeddings in the final exact scoring stage, a hyperparameter β is used. Thus the weight of an expansion embedding is influenced by both IDF and β.

Following our notations in Section 2, the full ColBERT-PRF pipeline, which performs query embedding expansion on the retrieved documents from an earlier stage, applied to a passage ranking can be formulated as:

$$ColBERT\text{-}PRF \quad » \quad ColBERT\text{-}MaxSim, \tag{2}$$

where *ColBERT-MaxSim* denotes the Max-Sim reranking stage of ColBERT. ColBERT-PRF has been shown to be effective upon the reranking of a ColBERT dense retrieval first stage [16], i.e.:

$$ColBERT \quad » \quad ColBERT\text{-}PRF \quad » \quad ColBERT\text{-}MaxSim. \tag{3}$$

This year in our participation, we varied the first stage retrieval for ColBERT-PRF, namely applying ColBERT-PRF on top of ColBERT, TCT-ColBERT as well as SPLADE. The retrieval pipelines are all similar, but vary the source of the the first stage retrieval.

3.3 Experimental Setup Details

We use the following pipeline components, grouped into methods that perform retrieval, document re-scoring, and pseudo-relevance feedback (PRF):

Retrievers:

- **DPH** [2] and **BM25** [15]: Lexical retrieval from a Terrier inverted index over the `msmarco-passage-v2` corpus.
- **SPLADE** [3, 4]: A distilled SPLADE++ retrieval model.[1]
- **ColBERT** [5]: A late-interaction end-to-end dense retrieval model (E2E), which consists of approximate nearest neighbour search, denoted as ColBERT-ANN and maximum similarity search, denoted as ColBERT-MaxSim in this paper.
- **TCT** [6]: A TCT-ColBERT single-representation retrieval model.[2]

Scorers:

- **ELECTRA** [14]: A version of the monoELECTRA scoring model trained with hard negatives.[3]
- **monoT5** [13]: A monoT5 scoring function.[4]

Pseudo-Relevance Feedback:

- **Bo1** [1]: Pseudo-relevance feedback using the DFR Bo1 model over a Terrier index.
- **ColBERT-PRF** [16]: A dense pseudo-relevance feedback model. The default ColBERT-PRF is implemented on top of ColBERT.
- **GAR**$_G(S)$ [8]: Graph-based Adaptive ReRanking using corpus graph G and scoring function S. We use both a BM25-based corpus graph and a TCT-based corpus graph, and ELECTRA as our scoring function. We use a re-ranking budget of 5000 and corpus graphs with 8 nearest neighbours.

All our experiments were conducted on the PyTerrier [10, 11] IR experimentation platform. PyTerrier is available from https://github.com/terrier-org/pyterrier.

4 SUBMITTED RUNS

We submitted eight group runs to the passage ranking task. We also submitted six baseline runs. We did not participate in the document ranking task.

4.1 Baseline Runs

The baselines that we submitted to the 2022 Deep Learning passage ranking track, including two traditional sparse retrieval runs with and without query expansion, two SPLADE runs with and without ELECTRA reranking, one BM25 run reranking using ELECTRA and one ColBERT end-to-end run. All the baseline runs are summarised as follows:

- `uogtr_dph`: Conducts DPH on our passage sparse index.
- `uogtr_dph_bo1`: Conducts DPH and Bo1 query expansion on our passage sparse index.
- `uogtr_s`: Conducts SPLADE retrieval on an `msmarco-passage-v2` SPLADE learned sparse index.
- `uogtr_se`: Conducts SPLADE retrieval, followed by re-ranking with ELECTRA.
- `uogtr_be`: Conducts BM25 retrieval then re-ranked using ELECTRA.
- `uogtr_c`: Conducts ColBERT end-to-end (E2E) retrieval on the ColBERT dense index.

4.2 Group Runs

For the 2022 Deep Learning passage ranking track, we submitted the following 8 runs:

- `uogtr_be_gb`: Conducts BM25 retrieval, adaptively re-ranked using the ELECTRA and a BM25 graph.
- `uogtr_se_gb`: Conducts SPLADE retrieval, adaptively re-ranked using the ELECTRA with a BM25 graph.

[1] `naver/splade-cocondenser-ensembledistil`

[2] `castorini/tct_colbertv2-msmarco` [3] `crystina-z/monoELECTRA_LCE_nneg31`

[4] `castorini/monot5-base-msmarco`

Table 1: Results on the TREC Deep Learning track 2022 Passage Ranking track. The best performing run for each measure is emphasised. The symbol ⋄ indicates that the corresponding metric improves over the TREC Median value.

Run ID	Pipeline	nDCG@10	MAP@100	RR@100	P@10	Recall@100	Judged@10
TREC Best (per-topic)		0.8283	0.3591	1.000	0.8316	-	-
TREC Median (per-topic)		0.5435	0.1364	0.7378	0.4434	-	-
		baseline runs					
uogtr_dph	DPH	0.2905	0.0410	0.3065	0.1789	0.1460	1.0000
uogtr_dph_bo1	DPH » Bo1 » DPH	0.2905	0.0410	0.3065	0.1789	0.1460	1.0000
uogtr_se	SPLADE » ELECTRA	**0.6510**⋄	0.2252⋄	**0.8023**⋄	0.6000⋄	0.4137	1.0000
uogtr_be	BM25 » ELECTRA	0.6235⋄	0.1896⋄	0.7782⋄	0.5684⋄	0.3349	1.0000
uogtr_s	SPLADE	0.5697⋄	0.1831⋄	0.7015	0.4947⋄	0.3735	1.0000
uogtr_c	ColBERT (e2e)	0.5217	0.1319	0.7153	0.4053	0.2409	1.0000
		group runs					
uogtr_be_gb	BM25 » GAR_{BM25}(ELECTRA)	0.6480⋄	0.2113⋄	0.7907⋄	**0.6053**⋄	0.3802	0.9908
uogtr_se_gb	SPLADE » GAR_{BM25}(ELECTRA)	0.6508⋄	0.2252⋄	**0.8023**⋄	0.6000⋄	0.4133	1.0000
uogtr_e_gb	(BM25 | SPLADE) » GAR_{BM25}(ELECTRA)	0.6501⋄	**0.2257**⋄	**0.8023**⋄	0.5987⋄	**0.4149**	0.9987
uogtr_se_gt	SPLADE » GAR_{TCT}(ELECTRA)	0.6508⋄	0.2256⋄	**0.8023**⋄	0.6000⋄	0.4140	1.0000
uogtr_t_cprf	TCT » ColBERT-PRF » ColBERT-MaxSim	0.5078	0.1646⋄	0.6269	0.4329	0.3410	0.8513
uogtr_s_cprf	SPLADE » ColBERT-PRF » ColBERT-MaxSim	0.5682⋄	0.1866⋄	0.6743	0.5013⋄	0.3665	1.0000
uogtr_c_cprf	ColBERT » ColBERT-PRF » ColBERT-MaxSim	0.5075	0.1355	0.6390	0.4184	0.2480	0.8618
uogtr_e_cprf_t5	(uogtr_t_cprf |uogtr_c_cprf) » monoT5	0.6182⋄	0.2061⋄	0.7541⋄	0.5539⋄	0.3739	1.0000

- `uogtr_e_gb`: Conducts the ensemble of BM25 and SPLADE retrieval, adaptively re-ranked using ELECTRA with a BM25 graph.
- `uogtr_se_gt`: Conducts SPLADE retrieval, adaptively re-ranked using ELECTRA with a TCT-ColBERT graph constructed using HNSW data.
- `uogtr_t_cprf`: Conducts ColBERT-PRF on top of the TCT-ColBERT retrieval, then reranks using ColBERT.
- `uogtr_s_cprf`: Conducts ColBERT-PRF on top of the SPLADE retrieval, then reranks using ColBERT.
- `uogtr_c_cprf`: Conducts the ColBERT-PRF reranker on top of ColBERT end-to-end (E2E), then reranks using ColBERT.
- `upgtr_e_cprf_t5`: Conducts the ensemble of the retrieved documents from the TCT_ColBERT reranking using ColBERT-PRF and ColBERT E2E reranking using ColBERT-PRF, then reranking using the monoT5 reranker.

5 RESULTS & ANALYSIS

Table 1 lists the obtained effectiveness results for all our passage ranking task runs, including six baseline runs and eight submitted group runs, as well as the TREC per-topic best and median scores across all participating systems, in terms of nDCG@10, MAP@100, MRR and P@10. In addition, we also report the Recall@100 and the Judged@10 for each model.

Firstly, we analyse the performance of our submitted baseline runs in Table 1. The top part shows the baseline runs, which consist of two sparse retrieval – namely `uogtr_dph` and `uogtr_dph_bo1` with the Bo1 query expansion model – while the other four are the neural retrieval models. Comparing between the sparse and neural retrieval baselines, we find that all four neural retrieval models outperform both the sparse retrieval models across all metrics. In addition, comparing between `uogtr_dph` and `uogtr_dph_bo1`, we observe that sparse query expansion shows slight improvements over sparse retrieval on all reported metrics. Comparing between `uogtr_s` and `uogtr_se` runs, where we conduct SPLADE with and without the ELECTRA reranker, we find that the ELECTRA reranker run improves effectiveness. Besides, comparing between the `uogtr_se` and `uogtr_be` runs, we find that SPLADE is better than the BM25 first stage retrieval. In addition, we find that the ColBERT model exhibits lower performance than SPLADE. Overall, SPLADE and the two ELECTRA reranking baseline runs exhibit higher performances than the TREC Median performance, while two sparse retrieval and ColBERT baseline runs underperform the TREC Median performance.

Next, we turn our attention to our submitted group runs, in the lower part of Table 1. The submitted group runs can be categorised into two families: the Adaptive ReRanking runs, namely the `uogtr_be_gb`, `uogtr_be_gb` and `uogtr_be_gb` as well as `uogtr_be_gt`, and the ColBERT-PRF reranking runs, namely the `uogtr_t_cprf`, `uogtr_s_cprf` and `uogtr_c_cprf` as well as `uogtr_e_cprf_t5`. From Table 1, we can see that when comparing between the `uogtr_c` and `uogtr_c_cprf` runs, we find that on the TREC 2022 query set, ColBERT-PRF shows a slightly lower performance than the ColBERT model. On the other hand, all four of the Adaptive ReRanking models outperform the ColBERT-PRF models across all the metrics. Comparing among the Adaptive ReRanking models, we find that for nDCG@10, the run `uogtr_e_gb`, where we perform the ensemble of BM25 and SPLADE retrieval, adaptively re-ranked using ELECTRA with a BM25 graph, exhibits the highest performance. In addition, the run `upgtr_se_gt`, where we apply a SPLADE first stage retrieval, adaptively re-ranked using ELECTRA with a TCT-ColBERT graph constructed using HNSW data, shows the highest MAP@100 performance. This indicates the superiority of SPLADE retrieval over BM25 retrieval, particularly in terms of Recall@100. Similarly, applying Adaptive ReRanking upon SPLADE using a

Table 2: Results on the TREC Deep Learning track 2022 Passage Ranking track with duplicate documents removed. The best performing run for each measure is emphasised. The symbols ↑ and ↓ indicate that the corresponding metric improves or degrades compared to the model performance for the same metric presented in Table 1.

Run ID	Pipeline	nDCG@10	MAP@100	RR@100	P@10	Recall@100	Judged@10
		baseline runs (post-deduped)					
uogtr_dph	DPH	0.2774$^{\downarrow}$	0.0367$^{\downarrow}$	0.3129$^{\uparrow}$	0.1553$^{\downarrow}$	0.1400$^{\downarrow}$	1.0000
uogtr_dph_bo1	DPH » Bo1 » DPH	0.2989$^{\uparrow}$	0.0397$^{\downarrow}$	0.3264$^{\downarrow}$	0.1776$^{\downarrow}$	0.1452$^{\downarrow}$	1.0000
uogtr_se	SPLADE » ELECTRA	0.6410$^{\downarrow}$	0.2028$^{\downarrow}$	**0.8045**$^{\uparrow}$	0.5882$^{\downarrow}$	0.3810$^{\downarrow}$	1.0000
uogtr_be	BM25 » ELECTRA	0.6094$^{\uparrow}$	0.1694$^{\downarrow}$	0.7793$^{\uparrow}$	0.5500$^{\downarrow}$	0.3080$^{\downarrow}$	1.0000
uogtr_s	SPLADE	0.5596$^{\downarrow}$	0.1650$^{\downarrow}$	0.7114$^{\uparrow}$	0.4750$^{\downarrow}$	0.3431$^{\downarrow}$	1.0000
uogtr_c	ColBERT (e2e)	0.5084$^{\downarrow}$	0.1179$^{\downarrow}$	0.7141$^{\downarrow}$	0.3908$^{\downarrow}$	0.2191$^{\downarrow}$	0.9974$^{\downarrow}$
		group runs (post-deduped)					
uogtr_be_gb	BM25 » GAR_{BM25}(ELECTRA)	0.6332$^{\downarrow}$	0.1888$^{\downarrow}$	0.7947$^{\uparrow}$	0.5855$^{\downarrow}$	0.3516$^{\downarrow}$	0.9816$^{\downarrow}$
uogtr_se_gb	SPLADE » GAR_{BM25}(ELECTRA)	0.6419$^{\downarrow}$	0.2032$^{\downarrow}$	**0.8045**$^{\uparrow}$	**0.5895**$^{\downarrow}$	**0.3836**$^{\downarrow}$	1.0000
uogtr_e_gb	(BM25 | SPLADE) » GAR_{BM25}(ELECTRA)	**0.6423**$^{\downarrow}$	0.2032$^{\downarrow}$	0.8041$^{\uparrow}$	**0.5895**$^{\downarrow}$	0.3829$^{\downarrow}$	0.9974$^{\downarrow}$
uogtr_se_gt	SPLADE » GAR_{TCT}(ELECTRA)	0.6411$^{\downarrow}$	**0.2035**$^{\downarrow}$	**0.8045**$^{\downarrow}$	0.5882$^{\downarrow}$	0.3835$^{\downarrow}$	1.0000
uogtr_t_cprf	TCT » ColBERT-PRF » ColBERT-MaxSim	0.4919$^{\downarrow}$	0.1483$^{\downarrow}$	0.6357$^{\uparrow}$	0.4171$^{\uparrow}$	0.3157$^{\downarrow}$	0.8447$^{\downarrow}$
uogtr_s_cprf	SPLADE » ColBERT-PRF » ColBERT-MaxSim	0.5566$^{\downarrow}$	0.1657$^{\downarrow}$	0.6818$^{\uparrow}$	0.4895$^{\downarrow}$	0.3362$^{\downarrow}$	1.0000
uogtr_c_cprf	ColBERT » ColBERT-PRF » ColBERT-MaxSim	0.4916$^{\downarrow}$	0.1187$^{\downarrow}$	0.6446$^{\downarrow}$	0.3974$^{\downarrow}$	0.2206$^{\downarrow}$	0.8408$^{\downarrow}$
uogtr_e_cprf_t5	(uogtr_t_cprf |uogtr_c_cprf) » monoT5	0.6002$^{\downarrow}$	0.1829$^{\downarrow}$	0.7598$^{\uparrow}$	0.5289$^{\downarrow}$	0.3350$^{\downarrow}$	0.9974$^{\downarrow}$

BM25 corpus graph shows the highest RR@100, P@10, Recall@100 performance, as well as a higher Judged@10. Overall, all the four Adaptive ReRanking runs markedly outperform the TREC Median performance.

We now explore the specific research questions posed in Section 1.

5.1 ColBERT-PRF over Various Initial Retrieval Models

Among the different first stage retrieval models implemented with the ColBERT-PRF approach, we find that the SPLADE model has the best performance, namely the `uogtr_s_cprf`. In addition, comparing among the four ColBERT-PRF models, we find that the run `uogtr_e_cprf_t5`, which conducts the monoT5 reranker upon the ensemble of the results from the SPLADE reranking with ColBERT-PRF and the ColBERT E2E reranking with the ColBERT-PRF approach, exhibits the highest performance. In summary, we observe that applying ColBERT-PRF using the SPLADE initial retrieval and with monoT5 as reranker runs exhibit a higher performance than the TREC Median performance. Hence, in answer to the question about the performance of ColBERT-PRF over various initial retrieval models, we find that ColBERT-PRF can be extended to be implemented upon various initial retrieval result sets and that the ensemble run gives the highest effectiveness.

5.2 Impact of the Duplicates in the Corpus

Next, we turn our attention to the heavily-duplicated nature of the MS MARCO version 2 corpus, and the effects that this duplication might have on the retrieval models. The task organisers identified 19M duplicates (around 14% of the corpus). However, the duplicated items are included in the official evaluation setting (the results of which are presented in Table 1). To test the effect of the duplicates, in both post-hoc duplicate removal and pre-hoc duplicate removal settings, on the performance of our submitted runs, we construct a version of the relevance assessments where the duplicated assessments are removed. The results using these assessments are presented in Table 2.

We find that for all runs, the nDCG@10, MAP@100, RR@100 and Recall@100 performance decrease. More importantly, however, we find that the order of the systems is highly correlated; in particular, the Spearman ρ correlation coefficient is 0.96 for nDCG@10, indicating a very strong correlation (correlation computed over 14 runs in Table 1 and Table 2). This suggests that the official evaluation setting is unlikely to change the conclusions when evaluated w/o duplication removal.[5]

5.3 Lexical-only First-stage Retrieval

Finally, we examined the effect of the lexical-only first-stage retrieval, namely the traditional BM25 retrieval and the learned sparse retrieval, namely SPLADE. Comparing the runs where we apply the ELECTRA reranker on top of BM25 and SPLADE, we find that the SPLADE »ELECTRA run can result in a 4.4% improvement in terms of nDCG@10 (from 0.6235 to 0.6510 in Table 1). Furthermore, we observe that GAR_{BM25}(ELECTRA) on top of BM25 and SPLADE can both result in improvements. We note that the difference between the BM25 and SPLADE runs is very close (nDCG@10: 0.6480 vs. 0.6508 in Table 1), indicating that the traditional lexical-only first-stage retrieval is effective enough to replace the expensive learned-sparse retrieval under the advanced Adaptive ReRanking approach.

[5] The earlier notebook version of this paper contained an analysis of the impact of the duplicates on Adaptive ReRanking. As the NIST organisers have since changed how duplicates are applied in the evaluation methodology, we have removed this analysis.

6 CONCLUSIONS

Overall, our participation in the TREC 2022 Deep Learning track was a useful activity to explore a number of recently-proposed deep learning retrieval pipelines. We found that: (i) ColBERT-PRF can be extended to various first stage retrieval approaches; (ii) in terms of the retrieval effectiveness, the ensemble run with Apdative ReRanking gives the highest retrieval effectiveness; (iii) the large number of duplicates in the corpus has little impact on the order of the systems to be compared; and (iv) lexical-only retrieval is effective enough compared to the learned-sparse retrieval for Adaptive ReRanking.

REFERENCES

[1] Giambattista Amati. 2003. Probability models for information retrieval based on divergence from randomness Ph.D. thesis. *University of Glasgow* (2003).

[2] Giambattista Amati. 2006. Frequentist and bayesian approach to information retrieval. In *Proceedings of ECIR*. 13–24.

[3] Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 2022. From Distillation to Hard Negative Sampling: Making Sparse Neural IR Models More Effective. In *Proceedings of SIGIR*. 2353–2359.

[4] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking. In *Proceedings of SIGIR*. 2288–2292.

[5] Omar Khattab and Matei Zaharia. 2020. ColBERT: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of SIGIR*. 39–48.

[6] Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. 2021. In-batch Negatives for Knowledge Distillation with Tightly-coupled Teachers for Dense Retrieval. In *RepL4NLP-2021 Workshop*. 163–173.

[7] Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy J. Lin. 2020. Distilling Dense Representations for Ranking using Tightly-Coupled Teachers. *ArXiv* abs/2010.11386 (2020).

[8] Sean MacAvaney, Nicola Tonellotto, and Craig Macdonald. 2022. Adaptive Re-Ranking as an Information-Seeking Agent. (2022).

[9] Sean MacAvaney, Nicola Tonellotto, and Craig Macdonald. 2022. Adaptive Re-Ranking with a Corpus Graph. *arXiv preprint arXiv:2208.08942*.

[10] Craig Macdonald and Nicola Tonellotto. 2020. Declarative Experimentation in Information Retrieval Using PyTerrier. In *Proceedings of ICTIR*. 4526–4533.

[11] Craig Macdonald, Nicola Tonellotto, Sean MacAvaney, and Iadh Ounis. 2021. PyTerrier: Declarative experimentation in Python from BM25 to dense retrieval. In *Proceedings of CIKM*. 4526–4533.

[12] Yu A Malkov and Dmitry A Yashunin. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence* 42, 4 (2018), 824–836.

[13] Rodrigo Nogueira, Zhiying Jiang, and Jimmy Lin. 2020. Document ranking with a pretrained sequence-to-sequence model. *arXiv preprint arXiv:2003.06713* (2020).

[14] Ronak Pradeep, Yuqi Liu, Xinyu Zhang, Yilin Li, Andrew Yates, and Jimmy Lin. 2022. Squeezing Water from a Stone: A Bag of Tricks for Further Improving Cross-Encoder Effectiveness for Reranking. In *Proceedings of ECIR*. Springer, 655–670.

[15] Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. 1995. Okapi at TREC-3. *Nist Special Publication Sp* 109 (1995), 109.

[16] Xiao Wang, Craig Macdonald, Nicola Tonellotto, and Iadh Ounis. 2021. Pseudo-Relevance Feedback for Multiple Representation Dense Retrieval. In *Proceedings of ICTIR*. 297–306.

[17] Xiao Wang, Craig Macdonald, Nicola Tonellotto, and Iadh Ounis. 2022. ColBERT-PRF: Semantic Pseudo-Relevance Feedback for Dense Passage and Document Retrieval. *ACM Transactions on the Web* (2022).