RMIT at TREC Deep Learning Track 2020

J. Shane Culpepper
RMIT University
Melbourne, Australia

Binsheng Liu
RMIT University
Melbourne, Australia

Abstract

RMIT University submitted multiple baseline runs to improve the diversity of system types in the judgment pool for the TREC Deep Learning Track in 2020. All of these runs used the publicly available Terrier and Indri search engines and no machine learning. In addition, we submitted a single non-baseline run which applied a custom pairwise ranker to a Bart transformer to rerank passages for the passage ranking task. The RMIT Bart run as well as several of the more basic baseline runs performed well overall in the document and passage ranking tasks based on the preliminary assessment provided by NIST.

1 Introduction

The TREC Deep Learning Track in in its second year in 2020. The track used the MS-MARCO [Bajaj et al., 2016] document and passage reranking collection created by Microsoft just as it did in 2019. NIST provided 100 new queries to participants, who were asked to provide the top-100 documents and passages from their respective MS-MARCO collections. This year, the track organizers provided an additional training collection created at Microsoft, called ORCAS [Craswell et al., 2020], which provides an additional 18 million training queries.

One of the key goals of the track is to explore the performance of Deep Learning models for information retrieval and to compare these new approaches with more traditional IR ranking algorithms. The results were promising in 2019, but relatively few submissions were received which did not use Deep Learning. So, in 2020, the organizers reached out to several IR research groups and asked them to submit additional baseline runs that used more traditional IR techniques such as query expansion, sequential dependency models, and feature-based learning-to-rank.

In response to the request, RMIT submitted 12 baseline runs to the document ranking track and 8 baseline runs to the passage ranking task. None of the runs use LTR as this remains difficult to do with the current collection as only plain text is available for passages and documents in the collection, and many of the most effective features require semi-structured data such as HTML. We did however submit a single Deep Learning run using BART for the passage ranking task for comparison.

In the following sections we will describe the collection curation and processing, baseline configurations, and a few initial observations.

2 Collection Preparation

While the TREC organizers provide a version of the document collection in TRECTEXT format, we opted to perform our own pre-cleaning of the collection to remove various errors in the text provided by Microsoft. All document text was processed using python along with the `clean-text[gpl]`, `spacy`, and `lemminflect` packages. The same transforms were applied to all documents, passages, and queries before constructing TRECTEXT files which were indexed using Indri or Terrier as described in the next section.

3 System Descriptions

Document Ranking Runs. The 12 runs submitted for the document ranking task are described in Table 1. We used two different publicly available search engines – Indri and Terrier. Note that we have a publicly available fork of Terrier which includes implementations of several stemmers which are not available in the official version. We also have made modifications to a few different ranking algorithms such as BM25 in order to allow us to set both k_1 and b parameters as we have found that the default values for these two parameters

Run Name	Description
RMIT_DFRee	Terrier v5.1, Krovetz stemming, DFRee ranker, DRF sequential dependency type SD with $\mu = 4000$ and n-gram length $= 5$, Bo1 query expansion with 5 documents and 50 terms.
RMIT_DPH	Terrier v5.1, Krovetz stemming, DPH ranker, DRF sequential dependency type SD with $\mu = 4000$ and n-gram length $= 5$, Bo1 query expansion with 2 documents and 50 terms.
indri-sdmf	Indri v5.17, Sequential Dependencies with fields, title $w = 0.35$, url $w = 0.35$, body unigram $w = 0.99$, coocurrence $w = 0.1$, unordered window of size 8 and $w = 0.2$
rindri-bm25	Indri v5.17, Krovetz stemming, Okapi ranker, $k_1 = 1.6$, $b = 0.7$
rmit_indri-fdm	Indri v5.17, Full Dependency Model with default parameters
rmit_indri-sdm	Indri v5.17, Sequential Dependency Model with default parameters
rterrier-dph	Terrier v5.1, Krovetz stemming, DPH ranker with default parameters.
rterrier-dph_sd	Terrier v5.1, Krovetz stemming, DPH ranker, DRF sequential dependency type SD with $\mu = 4000$ and n-gram length $= 5$. All other parameters were the default.
rterrier-expC2	Terrier v5.1, Krovetz stemming, In_expC2 ranker with $c = 2$.
rterrier-tfidf	Terrier v5.1, Krovetz stemming, TF_IDF ranker with default parameters.
rterrier-tfidf2	Terrier v5.1, Krovetz stemming, LemurTF_IDF ranker with default parameters.
terrier-jskls	Terrier v5.1, Krovetz stemming, Js_KLs ranker, DRF sequential dependency type SD with $\mu = 4000$ and n-gram length $= 5$, KL query expansion with 1 documents and 10 terms.

Table 1: System Descriptions for document ranking runs.

are rarely the best choice. All of the modifications can be found in a GitHub repository located at https://github.com/jsc/terrier-5.1.1.1. We performed a very basic parameter sweep of various system configurations using the QREL judgments from the 2019 track queries as the judgments are too shallow to perform parameter tuning using the MS MARCO test set, which contain a single judgment per query. The original MS MARCO test queries work reasonably well if you only wish to tune for reciprocal rank, but not for deeper measures used in the final TREC assessments such as AP@100 and NDCG@10. Therefore, the best parameter configurations were drawn from tuning runs with the 2019 queries, or the default parameters were used as described in the table.

Passage Ranking Runs. The 8 runs submitted for the document ranking task are summarized in Table 2. All of the baseline runs followed the same process as described for the document ranking runs. The one major exception was the RMIT-Bart which is the only non-standard baseline produced by RMIT in 2020. The details of this run a along with a basic analysis is provided in the next section.

4 RMIT Bart Run

4.1 Ranking Model

BART [Lewis et al., 2019] is a pretrained transformer model with an encoder-decoder architecture. We convert the BART model into a neural ranking model by adding a fine-tuning layer on top of the decoder layer. Figure 1 shows how the model works. Below the last feed-forward layer is the pretrained BART model which takes the concatenation of a query and a document as input for both the encoders and the decoders and produces contextualized embeddings for every token. To get a relevance prediction, we project the embeddings of the last token into a scalar score as only the last token can attend the whole sentence. This is in contrast with typical BERT ranking models [Nogueira and Cho, 2019, Nogueira et al., 2019] where the embeddings of the first token is used for fine-tuning.

The BartForSequenceClassification API from Huggingface is used to implement our ranker, but any

Run Name	Description
indri-fdm	Indri v5.17, Full Dependency Model with default parameters
indri-sdm	Indri v5.17, Sequential Dependency Model with default parameters
indri-lmds	Indri v5.17, Query Likelihood Model with $\mu = 460$
terrier-BM25	Terrier v5.1, Krovetz stemming, Okapi BM25 ranker modified to use $k_1 = 1.6$, $b = 0.7$.
terrier-dph	Terrier v5.1, Krovetz stemming, DPH ranker. All parameters were the default.
terrier-InL2	Terrier v5.1, Krovetz stemming, InL2 ranker. All parameters were the default.
TF_IDF_d_2_t_50	Terrier v5.1, Krovetz stemming, TF_IDF ranker, Bo1 query expansion with 2 documents and 50 terms.
DLF_d_5_t_25	Terrier v5.1, Krovetz stemming, DLF ranker, Bo1 query expansion with 5 documents and 25 terms.
RMIT-Bart	Anserini v0.9.3 indexing+retrieval, BART reranking (Details in Section 4).

Table 2: System Descriptions for passage ranking runs.

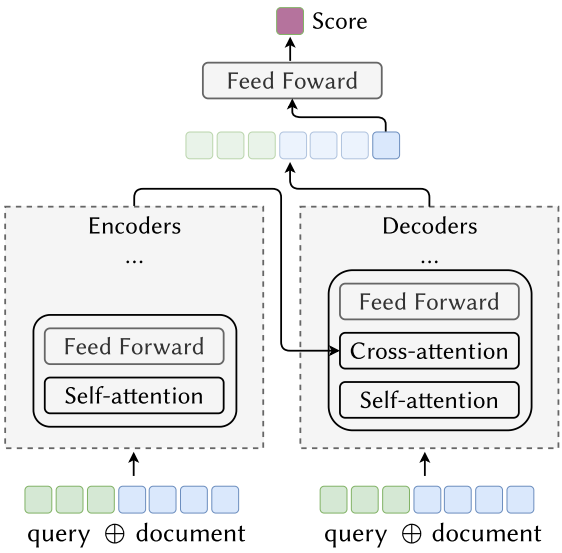


Figure 1: RMIT-BART ranking model takes as input the concatenation of a query and a document and produces a scalar score based on the last token's contextualized embeddings.

of the XXXForSequenceClassification APIs could be used to create a ranking model with few modifications, making it a very compelling tool for IR researchers to use in the future.

4.2 Training

We construct our own training data instead of using the tuples released by Microsoft. We first index the DeepCT [Dai and Callan, 2019] enriched collection and tune BM25 for recall using 1000 randomly selected training queries. Then we retrieve 1000 passages for each training query to use as our training data. The re-construction of the training data ensures that the passages in the training set, the validation set, and the testing set are consistent with our first-stage run.

Each training instance contains a query Q with one positive document D^+ and one pseudo-negative document D^- (there are no true negative judgments provided by the organizers). The loss for a training instance is calculated as

$$Loss = \max(0, 1 - (S(Q, D^+) - S(Q, D^-))) \quad (1)$$

where $S(\cdot)$ is the score produced by the ranking model.

We train the model for 10 epochs and sample new negative documents in each epoch. We set the learning rate to $5e^{-6}$ with a 1 epoch warmup, a training batch size of 32, gradient accumulation is set to 4 batches, and the maximum sequence length is set to 256 tokens. We use the AdamW [Loshchilov and Hutter, 2019] optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and weight_decay $= 5e^{-5}$.

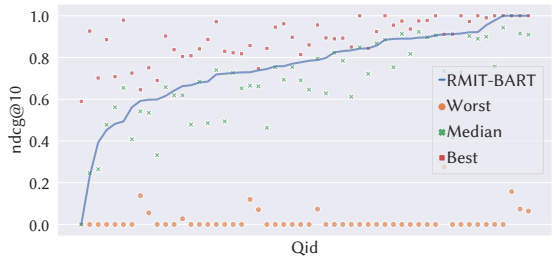
Using strong negative documents — which the model fails to rank lower than positive documents during training — to reinforce the model results when the effectiveness is low for the instance. We visually inspected a few strong negative documents and found that they could easily be labeled as positive if assessed by a human. This is not surprising

as the MS-MARCO judgments are not guaranteed to be complete. So care should be taken when researchers use this data augmentation technique as it may simply be an over-optimization that is highly collection-specific. Nevertheless, if the goal is to improve effectiveness on this collection it appears to work quite well.

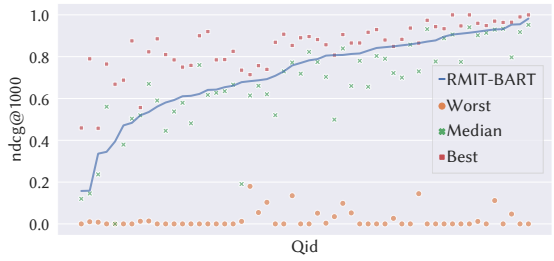
4.3 Results

Table 3: RMIT-BART effectiveness.

Metric	Value
NDCG@10	0.754
NDCG@20	0.721
NDCG@1000	0.719
MAP	0.512
Recall	0.809
RBP(0.50)	0.722 +0.000
RBP(0.80)	0.621 +0.018
RBP(0.95)	0.375 +0.186



(a) NDCG@10.



(b) NDCG@1000.

Figure 2: Per-query result.

Table 3 summarizes the effectiveness results for the RMIT-BART run in 2020. Figure 2 shows the RMIT-BART breakdown at the query level w.r.t. the worst, the median and the best results in terms of NDCG@10 and NDCG@1000 as provided by the organizers. It is clear RMIT-BART was more effectiveness than the median for the majority of assessed topics this round, but there is still room for improvement in the future.

Duplicates	Topic ID
157	640502
122	583468
99	555530
69	673670
29	1133579
15	169208
10	768208, 1132532
9	1108651
8	701453
6	1136043
5	258062
4	1037496
3	121171, 1127540, 1051399
2	877809, 390360, 330975, 135802, 1131069, 1110678, 1103153
1	730539, 174463, 141630, 1136047, 1115210, 1113256, 1106979, 1064670, 1049519, 1043135

Table 4: The 33 assessed topics which had one or more duplicate / near-duplicate query in the training sets.

RMIT-BART performed the worst for topic 673670 "what is a alm" and 405163 "is caffeine an narcotic" (leftmost of Figure 2a). The rank position 1 documents for theses two topics appear to be relevant in our opinion, but were labeled as non-relevant by the assessors. For example, document 8632360 is "ALM is a set of pre-defined processes that start somewhere in the business as an idea, a need ...". The relevant documents actually have different definitions of the acronym "ALM" which is ambiguous.

So it is quite possible one might expect some level of assessor disagreement on the current judgments, but this has not been studied for the collection to our knowledge. Since the topics do not originate with the NIST assessors which is often the case in TREC exercises, the assessors would not be classified as "gold", which can lead to lower assessor agreement [Damessie et al., 2017, 2018]. This is always a possibility when assessing results with only a query and not a full description of the true information need. Relevance labels become more likely to interpreted differently when assessed independently by multiple assessors. This is a classic and lingering problem is IR evaluation worth exploring further in future work.

5 Other Observations

One interesting observation related to the TREC DL 2020 Task – there appear to be a significant number of duplicate and near duplicate queries in the training data available. Table 4 summarizes the duplicate or near duplicate queries contained in one of the training set. Of the 60 assessed topics, roughly half of them had one or more duplicate in the training sets. We intend to do a more detailed analysis of this phenomenon in future work.

6 Conclusion

We have described all of the runs submitted for the TREC DL 2020 reranking tasks in this report. Our baseline runs used basic Terrier and Indri configurations. In addition, we have performed a preliminary analysis of the RMIT-Bart run which applied a more common Deep Learning approach similar to many of the other participating teams. The RMIT Bart run as several of the more basic baseline runs performed well overall in the document and passage ranking tasks based on the preliminary assessment provided by NIST, which we will analyze further once the TREC conference concludes in November of 2020.

Acknowledgments

This work was supported by Australian Research Council Grant DP190101113.

References

P. Bajaj, D. Campos, N. Craswell, L. Deng, J. Gao, X. Liu, R. Majumder, A. McNamara, B. Mitra, T. Nguyen, M. Rosenberg, X. Song, A. Stoica, S. Tiwary, and T. Wang. MS MARCO: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*, 2016.

N. Craswell, D. Campos, B. Mitra, E. Yilmaz, and B. Billerbeck. ORCAS: 18 million clicked query-document pairs for analyzing search. *arXiv preprint arXiv:2006.05324*, 2020.

Z. Dai and J. Callan. Context-Aware Sentence/Passage Term Importance Estimation For First Stage Retrieval. *arXiv:1910.10687 [cs]*, Nov. 2019.

T. T. Damessie, T. P. Nghiem, F. Scholer, and J. S. Culpepper. Gauging the quality of relevance assessments using inter-rater agreement. In *Proc. SIGIR*, pages 1089–1092, 2017.

T. T. Damessie, J. S. Culpepper, J. Kim, and F. Scholer. Presentation ordering effects on assessor agreement. In *Proc. CIKM*, pages 723–732, 2018.

M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. *arXiv:1910.13461 [cs, stat]*, Oct. 2019.

I. Loshchilov and F. Hutter. Decoupled Weight Decay Regularization. *arXiv:1711.05101 [cs, math]*, Jan. 2019.

R. Nogueira and K. Cho. Passage Re-ranking with BERT. *arXiv:1901.04085 [cs]*, jan 2019.

R. Nogueira, W. Yang, K. Cho, and J. Lin. Multi-Stage Document Ranking with BERT. *arXiv:1910.14424 [cs]*, oct 2019.