H_2oloo at TREC 2020: When all you got is a hammer... Deep Learning, Health Misinformation, and Precision Medicine

Ronak Pradeep, Xueguang Ma, Xinyu Zhang, Hang Cui, Ruizhou Xu, Rodrigo Nogueira, and Jimmy Lin

David R. Cheriton School of Computer Science
University of Waterloo

Abstract

The $h_2$oloo team from the University of Waterloo participated in the TREC 2020 Deep Learning, Health Misinformation, and Precision Medicine Tracks. Our primary goal was to validate sequence-to-sequence based retrieval techniques that we have been working on in the context of multi-stage retrieval dubbed "Expando-Mono-Duo" [6, 10] comprising a candidate document generation stage (driven by "bag of words" techniques) followed by a pointwise and then a pairwise reranking stage built around T5 [11], a powerful sequence-to-sequence transformer language model. For the Health Misinformation task, we also employ learnings from our fact verification system, VerT5erini [9].

All of our experiments employed the open-source Anserini IR toolkit [14, 16], which is based on the popular open-source Lucene search library, for initial retrieval that feeds the T5-based rerankers. Besides being the state of the art in various other collections (e.g., Robust04 and TREC-COVID), we found our models achieved much better effectiveness compared to the BM25 baselines as well as the median scores in all three tracks, demonstrating the versatility and the zero-shot transfer capabilities of our multi-stage ranking system.

1 Introduction

The $h_2$oloo team from the University of Waterloo participated in multiple tracks at TREC 2020. This notebook paper describes our approach in the Deep Learning, Health Misinformation, and Precision Medicine Tracks.

We use a two-stage ranking architecture coupled with pre-indexing document expansion, all of which use the T5 sequence-to-sequence transformer [11]. This earns it the name, "Expando-Mono-Duo T5" [10]. The general strategy involves an initial BM25-based keyword retrieval that is refined by a pointwise and then pairwise ranking. Document expansion is used when feasible to enrich keyword representations in the inverted index.

2 Multi-Stage Ranking with T5

In our formulation, a multi-stage ranking architecture comprises a number of stages, denoted H_0 to H_N. Except for H_0, which retrieves k_0 candidates from an inverted index, each stage H_n receives a ranked list R_{n-1} comprising k_{n-1} candidates from the previous stage. Each stage, in turn, provides a ranked list R_n comprising k_n candidates to the subsequent stage, with the obvious requirement that $k_n \leq k_{n-1}$. The ranked list generated by the final stage H_N is designated for consumption by the (human) searcher. However, prior to building the inverted index that feeds H_0, we first perform

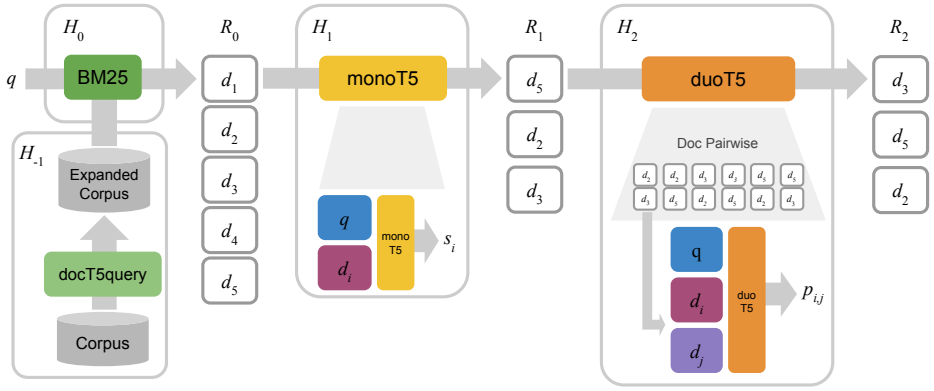

Figure 1: Illustration of our multi-stage ranking architecture. Prior to indexing, we (optionally) perform document expansion, denoted H_{-1}. In stage H_0, given a query q, the top-k_0 ($= 5$ in the figure) candidate documents R_0 are retrieved using BM25. In stage H_1, monoT5 produces a relevance score s_i for each pair of query q and candidate $d_i \in R_0$. The top-k_1 ($= 3$ in the figure) candidates with respect to these relevance scores are passed to stage H_2, in which duoT5 computes a relevance score $p_{i,j}$ for each triple (q, d_i, d_j). The final list of candidates R_2 is formed by reranking the candidates according to these scores.

document expansion on the input corpus to enrich its representation (we denote this as the H_{-1} stage). We describe each component of the overall architecture (see Figure 1) in detail below.

2.1 H_{-1}: Doc Expansion with docT5query

The idea behind document expansion is to enrich each document with additional terms that are representative of its content. In our particular implementation, we take a corpus of (question, relevant document) pairs and train a sequence-to-sequence model to predict, given a document, questions that it can potentially answer. These questions are then directly appended to the document; once this expansion has been performed for every document, the collection is indexed, as before.

As the first stage of our pipeline, we expand all documents in the MS MARCO corpus with queries predicted with docT5query [7]. The model was trained with a constant learning rate of 10^{-3} for 4k iterations with batches of 256, which corresponds to 2 epochs with the MS MARCO passage ranking training set. We use a maximum of 512 input tokens and 64 output tokens. In the MS MARCO passage ranking dataset, none of the inputs or outputs have to be truncated when using these lengths. Similar to Nogueira et al. [8], we find that the top-k sampling decoder [2] produces more effective queries than beam search. We use $k = 10$ and sample 40 queries per document.

We use T5-base as we did not notice any improvement in retrieval effectiveness with the large model. We did not experiment with T5-3B and T5-11B due to their computational cost. We use Google's TPU v3s to train and run inference. Training takes less than 1.5 hours on a single TPU. For inference, sampling 5 queries per document for 8.8M documents requires approximately 40 hours on a single TPU, costing $96 USD (40 hours $\times$ $2.40 USD/hour) using preemptible TPUs. Note that inference is trivially parallelizable and linear with respect to the number of samples. All expanded documents are then indexed with the Anserini IR toolkit [15] (post-v0.9.2); the expanded queries are appended to the original documents, but not specially delimited.

2.2 H_0: "Bag of Words" BM25

The stage H_0 receives as input the user query q and produces top-k_0 candidates R_0. In our implementation, the query is treated as a "bag of words" for ranking documents from the corpus using a standard inverted index based on the BM25 scoring function [12]. We use the Anserini IR toolkit [14, 16],[1] which is built on the popular open-source Lucene search engine.

[1] `http://anserini.io/`

2.3 H_1: Pointwise Reranking with monoT5

In general, the task of a reranking stage H_n is to estimate a score s_i quantifying how relevant a candidate $d_i \in R_{n-1}$ is to a query q. Naturally, we expect that the ranking induced by these scores yields a higher metric (e.g., MAP or MRR) than the scores from the previous stage.

In stage H_1, the documents retrieved in H_0 are reranked by a pointwise reranker, which we call monoT5. Our reranking method is based on Nogueira et al. [6], which uses T5 [11], a sequence-to-sequence model that uses a similar masked language modeling objective as BERT to pretrain its encoder–decoder architecture. In this model, all target tasks are cast as sequence-to-sequence tasks. We adapt the approach to document ranking by using the following input sequence:

$$\text{Query: } q \quad \text{Document: } d \quad \text{Relevant:}$$

where q and d are the query and document texts, respectively. The model is fine-tuned to produce the words "true" or "false" depending on whether the document is relevant or not to the query. That is, "true" and "false" are the "target words" (i.e., ground truth predictions in the sequence-to-sequence transformation).

At inference time, to compute probabilities for each query–document pair (in a reranking setting), we apply a softmax only on the logits of the "true" and "false" tokens. Hence, we rerank the documents according to the probabilities assigned to the "true" token. We arrived at this particular approach after some trial and error. Other approaches, for example, reranking documents according to the logit of the "true" token or using logits of all tokens to compute the softmax, were not effective, i.e., the retrieval metrics were close to zero.

We note that while H_0 uses a corpus enriched by document expansion, documents in R_0 consist of original texts that are not expanded; this is due to the input length restrictions of T5.

We train our models on MS MARCO passage [1], which is a passage ranking dataset with 8.8M passages obtained from the top 10 results retrieved from the Bing search engine. The training set contains approximately 500K pairs of query and relevant documents. Each query has one relevant passage, on average. Non-relevant documents for training are also provided as part of the training dataset.

We fine-tuned our monoT5-3B model on the MS MARCO passage ranking training set with a constant learning rate of 10^{-3} for 100K iterations with class-balanced batches of size 128. We use a maximum of 512 input tokens and one output token. In the MS MARCO passage ranking dataset, none of the inputs have to be truncated when using this length. Training monoT5-3B take approximately 160 hours overall on a single Google's TPU v3-8.

At inference time in all but the Deep Learning passage ranking task, the document length is usually much longer than the length restrictions of the model. Hence, it is not possible to feed the *entire* text of the document into our model at once. To address this issue, we first segmented each document into passages by applying a sliding window of n_{length} sentences with a stride of n_{stride}. We obtained a probability of relevance for each segment by performing inference on it independently, and then selecting the highest probability among the segments as the relevance score of the document.

2.4 H_2: Pairwise Reranking with duoT5

The output R_1 from the previous stage is used as input to the pairwise reranker we call duoT5. Within the framework of "learning to rank", duoT5 can be characterized as a "pairwise" approach, while monoT5 can be characterized as a "pointwise" approach [4]. In this pairwise approach, the reranker estimates the probability $p_{i,j}$ of the candidate d_i being more relevant than d_j for query q, where $i \neq j$.

This reranker, also using T5, instead takes as input the sequence:

$$\text{Query: } q \quad \text{Document0: } d_i \quad \text{Document1: } d_j \quad \text{Relevant:}$$

The pairwise sequence-to-sequence model is fine-tuned to produce the words "true" or "false" depending on whether the document d_i has higher or lower relevance than d_j to the question q.

At inference time, we aggregate the pairwise scores $p_{i,j}$ so that each document receives a single score s_i. We ablate over the number of candidates k_1 that is reranked by the pairwise ranker. The number

of inference calls made per query is given by the number of candidate pairs i.e. $k_1(k_1 - 1)$. We only run experiments where $k_1 = 50$. We use the following aggregation technique:

$$\text{SYM-SUM} : s_i = \sum_{j \in J_i} (p_{i,j} + (1 - p_{j,i})) \tag{1}$$

where $J_i = \{0 \leq j < k_1, j \neq i\}$.

The candidates in R_1 are reranked according to their scores s_i to get the final list of candidates R_2. The output R_2, in our current framework, is provided to the end-user, and serves as the input to computing the final evaluation metrics.

We fine-tuned duoT5 from the monoT5 model trained on MS MARCO passage ranking dataset as it serves as a good initial point having learnt the task of pointwise ranking. We use the same hyperparameters as those used for training monoT5. We initially experiment with duoT5-base and find that model performance converges at about 50K iterations. Hence, we train duoT5-3B for 50K iterations which corresponds to 80 hours overall on a single Google TPU v3-8.

At inference time, in all but the Deep Learning Passage Ranking task, we run duoT5 using the highest monoT5 scoring segment as the representative of the document. We increased the maximum input tokens for duoT5 from 512 to 1024 to account for pairs of passages that were longer than the default limit of 512 tokens. We were able to do so in T5 since the models were trained with relative positional encodings [13] and thus can (hopefully) generalize to contexts larger than those seen during training. This modification, however, imposed additional computational costs that come from the model needing to attend to twice the number of tokens; transformers exhibit quadratic complexity in both time and space with respect to input length [3].[2]

Note that if more than k_1 hits are requested as the output of duoT5, we simply take additional ranked output from monoT5. For example, if we requires 1000 hits, then the first 50 will come from duoT5 (assuming $k_1 = 50$), while the remaining results (rank positions 51–1000) will be the unaltered rankings from monoT5. Hence, the goal of the pairwise reranker is to improve the quality of results high in the ranked list.

3 Deep Learning Track

All inference experiments in the document ranking task are run with $n_{length} = 10$ and $n_{stride} = 5$. Based on the multi-stage pipeline described in Section 2, we submitted a total of four bag-of-words baseline runs and three neural runs to the Deep Learning Track for both passage and document ranking tasks. We first describe each of the baseline runs below.

BM25 (bm25): This is our baseline bag-of-words retrieval, i.e., H_0 is the only stage of ranking. We adopt all the default settings in Anserini's BM25 implementation.

BM25 + RM3 (bm25rm3): To examine the effects of query expansion, we employ the strong BM25 + RM3 baseline described in [17].

docT5query + BM25 (d2q_bm25): To examine the effects of document expansion, we apply the document expansion stage H_{-1}, followed by BM25 retrieval stage, H_0.

docT5query + BM25 + rm3 (d2q_bm25rm3): To examine the effects of both document expansion and query expansion, we apply the document expansion stage H_{-1}, followed by the BM25 + RM3 query expansion baseline in H_0.

We now describe each of the multi-stage neural reranking runs (which we call "Expando-Mono-Duo T5" runs) below.

BM25 + RM3 + monoT5 + duoT5 (bm25rm3_duo): We first rerank the top-k_0 (=1000) documents retrieved by BM25 + RM3 using our pointwise reranker, monoT5-3B. Then, we rerank the top-k_1 (=50) documents retreived by H_1 using our pairwise reranker, duoT5-3B.

[2]Note that increasing the length to 1024 tokens was sufficient in this case. However, for monoT5, such an increase would still not have been sufficient to perform inference on a complete document.

Run	MAP	nDCG@10	nDCG@1K	RR	R@1K
(0) Median	0.4413	0.6810	0.6631	0.8443	-
(1) p_bm25	0.2856	0.4796	0.5830	0.6585	0.7863
(2) p_bm25rm3	0.3019	0.4821	0.6046	0.6360	0.8217
(3) p_d2q_bm25	0.4074	0.6187	0.6840	0.7326	0.8452
(4) p_d2q_bm25rm3	0.4295	0.6172	0.7041	0.7424	0.8699
(5) p_bm25rm3_duo	0.5355	0.7583	0.7387	0.8759	-
(6) p_d2q_bm25_duo	0.5609	0.7837	0.7539	0.8798	-
(7) p_d2q_rm3_duo	0.5643	0.7821	0.7732	0.8798	-

Table 1: Results on TREC 2020 Deep Learning Track Passage Ranking Task.

Run	MAP	nDCG@10	nDCG@1K	RR	R@1K
(0) Median	0.3902	0.5733	0.5859	0.9444	-
(1) d_bm25	0.3791	0.5271	0.5647	0.8521	0.8085
(2) d_bm25rm3	0.4006	0.5248	0.5726	0.8541	0.8260
(3) d_d2q_bm25	0.4230	0.5885	0.6115	0.9369	0.8403
(4) d_d2q_bm25rm3	0.4228	0.5407	0.5902	0.8147	0.8596
(5) d_bm25rm3_duo	0.5270	0.6794	0.6929	0.9476	-
(6) d_d2q_bm25_duo	0.5422	0.6934	0.7089	0.9476	-
(7) d_d2q_rm3_duo	0.5427	0.6900	0.7122	0.9476	-

Table 2: Results on TREC 2020 Deep Learning Track Document Ranking Task.

docT5query + BM25 + monoT5 + duoT5 (d2q_bm25_duo): Same as BM25 + RM3 + monoT5 + duoT5 except we use H_{-1} and H_0 as in docT5query + BM25.

docT5query + BM25 + RM3 + monoT5 + duoT5 (d2q_bm25_duo): Same as BM25 + RM3 + monoT5 + duoT5 except we use H_{-1} and H_0 as in docT5query + BM25 + RM3.

3.1 Results

Results from the Passage Ranking and Document Ranking Tracks are show in Tables 1 and 2 respectively. In both tables, the first row shows the mean of the median per-topic scores, representing the score of a run that received the median score on all topics. The next four rows (rows 1 - 4) show the scores of the Anserini baseline runs and the final three rows (row 5 - 7) show the scores of our neural runs.

In Table 1, we generally find that all our baseline "bag of words" runs fall below the "median" scores. However, whenever we use document expansion (rows 3 and 4), we find our systems exceed the median in terms of nDCG@1K which demonstrates the strength of docT5query. We note similar results in Table 2. except some more scores exceed the median.

In both Tracks, it is clear that document expansion helps both the BM25 baseline (rows 1 and 3) and the BM25 + RM3 baseline (rows 2 and 4). However, query expansion generally only seems to help in the BM25 baseline (rows 1 and 2). While the improvements when it is used along with document expansion is not clear in terms of the official metrics, we still note improvements in terms of R@1K, the metric H_0 aims to maximize, as the resulting candidate set is passed onto neural rerankers.

The runs that employ multi-stage neural reranking (rows 5 - 7) show large improvements in effectiveness over their respective Anserini baselines (rows 2 - 4) as expected. Upon looking at the impact of document expansion and query expansion in the multi-stage ranking pipeline, we find that document expansion clearly helps our neural rerankers (rows 5 and 7) in terms of MAP, nDCG@10 and nDCG@1K. However, it is not clear if query expansion helps on top of document expansion (rows 6 and 7) as both sets of scores are very similar.

4 Health Misinformation Track

Macavaney et al. [5] demonstrates that fine-tuning the classifiers on Med-MARCO, a medical subset of MS MARCO, helps with biomedical-domain relevance ranking. We note similar results in the TREC-COVID task in [18]. Hence, we choose to use monoT5-3B and duoT5-3B models that were fine-tuned on MS MARCO passage ranking dataset then fine-tuned (again) on Med-MARCO. All inference experiments are run with $n_{length} = 6$ and $n_{stride} = 3$.

For this Track only, we added to our pipeline the label prediction model from VerT5erini [9]. We call this model LabelT5 and describe it next.

4.1 Label Prediction-based Reranking with LabelT5

Given the topic q and the highest monoT5 scoring segment s_i from a document d_i, the model is tasked to predict a label $\hat{y}(q, s_i) \in \{true, weak, false\}$. Here, we use the following input sequence:

$$\text{Query: } q \quad \text{Document: } s_i \quad \text{Relevant:}$$

We train the label prediction model using the effective judgements from the 2019 Decision (Medical Misinformation) Track. We map effective and ineffective judgements to "true" and "false" respectively. The documents judged as inconclusive, no info or not relevant are all labelled "weak" label.

We fine-tuned our LabelT5-3B model with a constant learning rate of 10^{-3} for 500 iterations with batches of size 128. We use a maximum of 512 inputs tokens and one output token. Training LabelT5-3B takes approximately 40 minutes on a single Google's TPU v3-8.

At inference time, to compute probabilities for each query–document pair (in a reranking setting), we apply a softmax only on the logits of the "true", "weak", and "false" tokens. Then, we rerank the documents according to the probabilities assigned to the "true" token if the answer field is "yes" and the probabilities assigned to the "false" token if the answer field is "no".

4.2 Runs

We submitted a total of nine runs to the Health Misinformation Track Ad-hoc Retrieval Task, which are described below.

BM25 (m1): This is our baseline bag-of-words retrieval, i.e., H_0 is the only stage of ranking. We adopt all the default settings. We index all the news articles found in the CommonCrawl News crawl from January 1st, 2020 to April 30th, 2020. At inference time, we retrieve the top-k_0 (=1000) documents per query. We do not use doc2query expansions (stage H_{-1}) in this track due to the high costs of expanding the very large corpus.

BM25 + monoT5 (m2): We rerank the top-k_0 (=1000) documents retrieved by BM25 using our pointwise reranker, monoT5-3B.

BM25 + monoT5$_{\text{answer}}$ (m3): We employ BM25 + monoT5 but we modify the query based on the answer field in the topic prior to neural reranking. We rephrase the question "Can *X Y* COVID-19?" (where *X* is a treatment and *Y* is one of five effect terms) to "*X* can *Y* COVID-19" if the answer field is "yes" and "*X* can not *Y* COVID-19" if the answer field is "no". The goal of this submission was to see if there are any improvements brought by aligning the query with the answer field.

BM25 + monoT5 + duoT5 (m4): We rerank the top-k_1 (=50) documents returned by BM25 + monoT5 using our pairwise reranker, duoT5-3B.

BM25 + monoT5$_{\text{answer}}$ + duoT5$_{\text{answer}}$ (m5): We rerank the top-k_1 (=50) documents returned by BM25 + monoT5$_{\text{answer}}$ using our pairwise reranker, duoT5-3B. duoT5-3B uses the rephrased query as that in BM25 + monoT5$_{\text{answer}}$.

BM25 + monoT5$_{\text{credible}}$ (m6): Same as BM25 + monoT5$_{\text{answer}}$ except that while rephrasing we also prefix the query with the text "Clinical Studies, FDA, CDC, Health Officials, WHO or researchers say". Note that in our submission, we called this run m10 but from here on we refer to it as m6.

BM25 + monoT5$_{\text{answer}}$ + LabelT5 (m7): We rerank the top-k_1 (=1000) document segments returned by BM25 + monoT5$_{\text{answer}}$ using our label prediction model, LabelT5.

BM25 + Average of monoT5$_{\text{answer}}$ and LabelT5 (m8): We take the mean of the scores of top-k_1 (=1000) documents returned by BM25 + monoT5$_{\text{answer}}$ and the scores of the same documents by LabelT5.

BM25 + Linear Combination of monoT5$_{\text{credible}}$ and monoT5$_{\text{answer}}$ (m9): We take a linear combination of the score s_{answer} from BM25 + monoT5$_{\text{answer}}$ and the score $s_{credible}$ from BM25 + monoT5$_{\text{credible}}$. Then, we re-weight the top-1000 documents as $s = 2s_{credible} + s_{answer}$, where the weights 2 and 1 are arbitrarily selected so that we weigh credibility scores higher, while trying to retain some notion of standard relevance.

4.3 Results

Harmful and helpful compatibility scores are provided in Table 3. First, we notice that all our systems score higher than the "median" helpful compatibility score. Pointwise reranking helps on top of the BM25 baseline (rows 1 and 2) and pairwise reranking helps on top of pointwise reranking (rows 2 and 4) as expected. The helpful compatibility score is also improved when the query is rephrased based on the alignment (rows 2-5). Interestingly, using LabelT5 results in a drop in the helpful compatibility scores (rows 3 and 7). This is perhaps because the amount of training data we have from the TREC 2019 Health Misinformation Track is limited. Averaging scores with the relevance classifier helps mitigate some of this (rows 3, 7 and 8). Using query prefix in m6 seems to have a detrimental effect compared to the standard alignment-based rephrasing (rows 3 and 6). Again in this case, the linear combination with the relevance classifiers scores helps bridge the gap (rows 3, 6 and 9). Yet, our zero-shot neural reranking pipeline m5 shines in terms of helpful compatibility scores.

Looking at harmful compatibility scores alone, a quantity we aim to minimize, we find the scores of m7 and m8 (rows 7 and 8) are much lower than those of the other systems. While having a low harmful compatibility score doesn't mean anything in isolation since a system that just outputs irrelevant information will have a score of 0, we note that both these runs have much higher than "median" helpful compatibility scores (the system that output irrelevant information would have a helpful compatibility score of 0 too, which isn't desirable at all). The improvement in harmful compatibility is perhaps due to the fact that the label prediction model is trained to specifically look out for misinformation and hence documents with incorrect information would rarely show up higher on the list. Averaging scores with the relevance classifier results in a very small loss in terms of harmful compatibility scores while largely bridging the gap in terms of helpful compatibility scores (rows 3, 7 and 8). Hence, m8 is a pretty notable run as it succeeds in finding helpful information while minimizing the exposure of incorrect information. Another trend we notice with harmful compatibility scores is that using the alignment information to rephrase queries helps a lot (rows 2 and 3 as well as rows 4 and 5).

In Table 4, other metrics based on usefulness, correctness and credibility are provided. Here we again note that all our runs have better scores than the "median" run. Surprisingly, our zero-shot neural reranking pipeline m5 outperforms all other systems in terms of the official metrics in this set of results.

5 Precision Medicine Track

Like the Health Misinformation Track, we use the monoT5-3B and duoT5-3B models that were fine-tuned on MS MARCO passage ranking dataset then fine-tuned (again) on Med-MARCO since the task is in the biomedical domain. All inference experiments are run with $n_{length} = 10$ and $n_{stride} = 5$. We submitted a total of six runs to the Precision Medicine Track, which are described below.

BM25: This is our baseline bag-of-words retrieval, i.e., H_0 is the only stage of ranking. We index all abstracts in PubMed. We do not use their full texts. At inference time, we retrieve the top-k_0 (=1000) documents per query. We do not use doc2query expansions (stage H_{-1}) in this track due to the high costs of expanding the PubMed corpus.

Model	COMP$_{harmful}$	COMP$_{helpful}$
(0) Median	0.0747	0.3337
(1) m1	0.1197	0.3679
(2) m2	0.1132	0.4404
(3) m3	0.0750	0.5107
(4) m4	0.1200	0.4658
(5) m5	0.0800	0.5487
(6) m6	0.0653	0.4832
(7) m7	0.0150	0.4486
(8) m8	0.0163	0.4900
(9) m9	0.0747	0.5017

Table 3: Harmful and helpful compatibility scores on TREC 2020 Health Misinformation Track.

Model	CMAP$_{co,cr}$	CMAP$_{us,cr}$	CMAP$_{co,us,cr}$	nDCG$_{us}$	nDCG$_{co}$	nDCG$_{cr}$	nDCG$_{all}$
(0) Median	0.1003	0.1717	0.1389	0.4699	0.3380	0.3308	0.4471
(1) m1	0.1911	0.2765	0.2369	0.6077	0.4997	0.5774	0.4853
(2) m2	0.2211	0.3149	0.2730	0.6387	0.5341	0.5945	0.5014
(3) m3	0.2534	0.3287	0.2915	0.6443	0.5620	0.6072	0.5306
(4) m4	0.2483	0.3402	0.2971	0.6596	0.5549	0.6185	0.5221
(5) m5	0.2898	0.3560	0.3187	0.6661	0.5901	0.6309	0.5609
(6) m6	0.2800	0.3105	0.2856	0.6281	0.5833	0.6065	0.5596
(7) m7	0.2423	0.2265	0.2216	0.5758	0.5638	0.5486	0.5212
(8) m8	0.2703	0.2633	0.2531	0.6018	0.5805	0.5725	0.5437
(9) m9	0.2733	0.3308	0.2974	0.6439	0.5774	0.6174	0.5553

Table 4: Usefulness, correctness and credibility results on TREC 2020 Health Misinformation Track.

BM25 + monoT5: We rerank the top-k_0 (=1000) documents retrieved by BM25 using our pointwise reranker, monoT5-3B. We transform queries into the natural language questions with the following template: "is *treatment* treatment effective for *disease* disease in patients with *gene* gene mutation?", where the *treatment*, *disease* and *gene* are the given keyword queries.

BM25 + monoT5 + duoT5: We rerank the top-k_1 (=50) documents returned by BM25+monoT5 using our pairwise reranker, duoT5-3B. The queries are transformed into natural language questions in the same way as BM25 + monoT5.

BM25 + monoT5$_{\text{rct}}$: Same as BM25 + monoT5, except prior to reranking we prepend the keywords "meta-analysis" and "randomized controlled trial (RCT)" to the query. The natural language queries are prepared in the same way as BM25 + monoT5. The primary goal here is to see if monoT5 can leverage these additional keywords to prioritize abstracts that belong to the top critical evidence tier according to the TREC 2020 Precision Medicine Track guidelines.

BM25 + monoT5$_{\text{rct}}$+duoT5$_{\text{rct}}$: Same as BM25 + monoT5 + duoT5, except prior to reranking we prepend the keywords "meta-analysis" and "randomized controlled trial (RCT)" to the query. The natural language queries are prepared in the same way as BM25 + monoT5. The goal of this run is to yet again see if duoT5 can leverage the same additional keywords described in the prior run.

BM25 + monoT5$_{\text{e1}}$: We rerank the top-k_1 (=100) documents returned by BM25+monoT5$_{\text{rct}}$ by penalizing the top documents from the BM25 + monoT5. Specifically, we re-weight the top 100 documents as $s' = 5s_{rct} - s$, where s is the score from BM25 + monoT5, s_{rct} is the score from BM25 + monoT5$_{\text{rct}}$, and the weights 5 and 1 are arbitrarily chosen to downweight by a small amount the documents that are "relevant" yet show little or no critical evidence.

All of our submissions are zero-shot, i.e., our models are trained solely on MS MARCO passage ranking dataset and we do not adjust their hyperparameters based on any of the prior years' TREC Precision Medicine Track.

Model	infNDCG	P@10	RPrec
(0) Median	0.4316	0.4645	0.3259
(1) BM25	0.4978	0.5355	0.3979
(2) + monoT5	0.5028	0.5000	0.4018
(3) + duoT5	0.5116	0.5290	0.3958
(4) + monoT5$_{rct}$	0.4039	0.4323	0.2998
(5) + duoT5$_{rct}$	0.4344	0.4839	0.3289
(6) + monoT5$_{e1}$	0.3635	0.2710	0.2845

Table 5: Phase 1 (Relevance Assessment) Results on TREC 2020 Precision Medicine Track.

Model	nDCG@30	nDCG@5
(0) Median	0.2857	0.2529
(1) BM25	0.3081	0.2793
(2) + monoT5	0.3341	0.2925
(3) + duoT5	0.3643	0.3093
(4) + monoT5$_{rct}$	0.4193	0.3998
(5) + duoT5$_{rct}$	0.3989	0.3747
(6) + monoT5$_{e1}$	0.3276	0.2055

Table 6: Phase 2 (Evidence Assessment) Results on TREC 2020 Precision Medicine Track.

5.1 Results

Table 5 shows the results from Phase 1 (Relevance Assessment) Evaluation of the TREC 2020 Precision Medicine Track. The first row shows the median score of all submitted runs. Our BM25 baseline (row 1) is only slightly worse than monoT5 (row 2) and duoT5 (row 3) with respect to infNDCG, but it is better than both neural models in terms of P@10. Results in terms of RPrec are mixed.

The variants monoT5$_{rct}$ (row 4), duoT5$_{rct}$ (row 5), and monoT5$_{e1}$ (row 6) performed worse than their respective base models (rows 2 and 3). These results however are not surprising as Relevance Assessment relies heavily on matching exact keywords from the topics.

Table 6 shows the results from Phase 2 (Evidence Assessment) Evaluation of the TREC 2020 Precision Medicine Track. Here, we note that most of our submissions score way higher than the median submission. In this case, we see the same trend from the other tracks where pointwise reranking helps on top of the BM25 baseline (rows 1 and 2) and pairwise reranking helps on top of pointwise reranking (rows 2 and 3).

Our experimental submission (row 6) performed poorly compared to all our other systems. This is perhaps because we penalized too much based on our monoT5 scores (row 2).

Leveraging the additional keywords belonging to the top evidence tier in the TREC 2020 Precision Medicine Track guidelines clearly helps in the case of the pointwise reranker as we see an 8.5 point improvement in effectiveness (rows 2 and 4). While we still notice a gain in performance in the pairwise reranker case (rows 3 and 5), we see the unexpected result that duoT5$_{rct}$ reranking the top segments from monoT5$_{rct}$ results in a drop in effectiveness (rows 4 and 5). We believe that this is because the additional keywords confuse the pairwise reranker more than it would a pointwise reranker, since it requires comparison between the two documents on two dimensions: which document is more relevant for the topic while also comparing based on the type of study. This is something a pairwise model trained only on a relevance ranking dataset might not capture well.

6 Conclusion

We have described our submissions to Deep Learning, Health Misinformation, and Precision Medicine Tracks of TREC 2020. Our standard pipeline (docT5query + BM25 + monoT5 + duoT5) has previously demonstrated strong zero-shot transfer capabilities on various domains such as news articles (Robust04) [6] and COVID-related scientific articles (TREC-COVID) [18]. These results were once again confirmed on all three tracks, in which our pipeline achieved much better effectiveness than BM25 baselines as well as the median system performance, while being minimally adapted to these tasks.

In the Health Misinformation Track, we note that a label prediction model makes the system much less prone to retrieving harmful documents while minimally decreasing the number of helpful documents retrieved. Future work improving on robust systems of this sort is critical to better misinformation-free retrieval.

In the Precision Medicine Track, we see that a simple query expansion model that prepends words from the top evidence tier, helps improve the quality of the run returned by our pointwise neural reranker. Further exploration involving aggregating scores across queries designed for multiple evidence tiers as well as other fields like citation counts and publication type might be important in building better retrieval systems for the Precision Medicine Track.

References

[1] 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:1611.09268*, 2016.

[2] A. Fan, M. Lewis, and Y. Dauphin. Hierarchical Neural Story Generation. *arXiv:1805.04833*, 2018.

[3] N. Kitaev, Łukasz. Kaiser, and A. Levskaya. Reformer: The efficient transformer. In *Proceedings of the 8th International Conference on Learning Representations (ICLR 2020)*, 2020.

[4] T.-Y. Liu. Learning to Rank for Information Retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

[5] S. MacAvaney, A. Cohan, and N. Goharian. SLEDGE: A Simple Yet Effective Baseline for COVID-19 Scientific Knowledge Search. *arXiv:2005.02365*, 2020.

[6] R. Nogueira, Z. Jiang, R. Pradeep, and J. Lin. Document ranking with a pretrained sequence-to-sequence model. In *Findings of EMNLP*, 2020.

[7] R. Nogueira and J. Lin. From doc2query to docTTTTTquery. 2019.

[8] R. Nogueira, W. Yang, J. Lin, and K. Cho. Document Expansion by Query Prediction. *arXiv:1904.08375*, 2019.

[9] R. Pradeep, X. Ma, R. Nogueira, and J. Lin. Scientific Claim Verification with VERT5ERINI. *arXiv:2010.11930*, 2020.

[10] R. Pradeep, R. Nogueira, and J. Lin. The Expando-Mono-Duo Design Pattern for Text Ranking with Pretrained Sequence-to-Sequence Models. *arXiv:2101.05667*, 2021.

[11] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67, 2020.

[12] S. E. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In *Proceedings of the 3rd Text REtrieval Conference (TREC-3)*, pages 109–126, 1994.

[13] P. Shaw, J. Uszkoreit, and A. Vaswani. Self-Attention with Relative Position Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468, June 2018.

[14] P. Yang, H. Fang, and J. Lin. Anserini: Enabling the use of Lucene for information retrieval research. In *Proceedings of the 40th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2017)*, pages 1253–1256, Tokyo, Japan, 2017.

[15] P. Yang, H. Fang, and J. Lin. Anserini: Enabling the use of Lucene for information retrieval research. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1253–1256, 2017.

[16] P. Yang, H. Fang, and J. Lin. Anserini: Reproducible ranking baselines using Lucene. *Journal of Data and Information Quality*, 10(4):Article 16, 2018.

[17] W. Yang, K. Lu, P. Yang, and J. Lin. Critically Examining the "Neural Hype". *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, Jul 2019.

[18] E. Zhang, N. Gupta, R. Tang, X. Han, R. Pradeep, K. Lu, Y. Zhang, R. Nogueira, K. Cho, H. Fang, et al. Covidex: Neural Ranking Models and Keyword Search Infrastructure for the COVID-19 Open Research Dataset. *arXiv:2007.07846*, 2020.