H_2oloo at TAC 2020: Epidemic Question Answering

Justin Borromeo*, Ronak Pradeep*, 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 TAC 2020 Epidemic QA (EPIC-QA) track. Our primary goal was to explore the effectiveness of "Mono-Duo-T5", an approach empirically validated for several *ad hoc* retrieval tasks in different domains, when transferred in a zero-shot manner to the task of answering questions related to the COVID-19 disease, the SARS-CoV-2 virus responsible for the illness, related coronaviruses, and response guidelines to the pandemic. Our system comprises a candidate generation stage using bag-of-words BM25 retrieval, followed by pointwise reranking (Mono) then pairwise reranking (Duo) using T5, a sequence-to-sequence transformer language model. To improve diversity in the system output, we use the maximal marginal relevance (MMR) algorithm to rerank sentences from the top segments. With this approach, we submitted the best run for EPIC-QA Task B (Consumer QA) according to all the metrics considered and demonstrated competitive results for EPIC-QA Task A (Expert QA).

1 Introduction

The University of Waterloo participated in the Epidemic QA track at TAC 2020. This paper describes our multi-stage ranking approach, which we employ for both Task A (Expert QA) and Task B (Consumer QA).

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.

We describe each component of the overall architecture (see Figure 1) in detail below.

2.1 H_0: "Bag of Words" BM25

The stage H_0 receives the user query q as input and produces top-k_0 candidate segments R_0. A candidate segment is defined as a sentence, which we call the *central sentence*, with some context sentences before and after it. Through experimentation, it was found that three sentences of context before and two sentences after the central sentence provided the most relevant answers.

*equal contribution

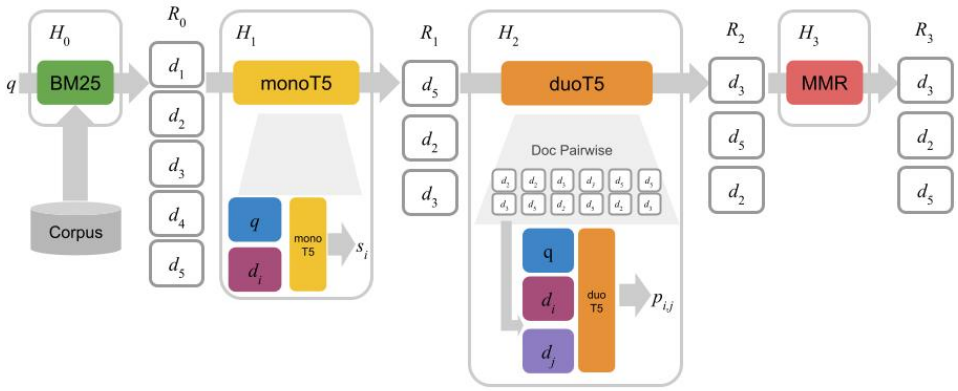

Figure 1: Illustration of our multi-stage ranking architecture. In stage H_0, given a query q, the top-k_0 ($= 5$ in the figure) candidate documents R_0 are retrieved using BM25. In stage H_1, monoT5 produces a relevance score s_i for each pair of query q and candidate $d_i \in R_0$. The top-k_1 ($= 3$ 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 list of candidates R_2 is formed by reranking the candidates according to these scores. The final output R_3 is generated by applying maximal marginal relevance ranking to the R_2 candidates.

In our implementation, the query is treated as a "bag of words" for ranking sentences from the corpus using a standard inverted index based on the BM25 scoring function [12]. We used Pyserini [1, 4],[2] which is the Python wrapper to the Anserini [15, 16],[3] IR toolkit. Anserini is built on the popular open-source Lucene search engine with the goals of supporting replicable IR research and bringing research practices into better alignment with real-world search applications.

2.2 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 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. [7], 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 to the logits of the "true" and "false" tokens and rerank based on the probabilities assigned to the "true" token, per Nogueira et al. [7].

We train our models on MS MARCO passage [2], a passage ranking dataset with 8.8M passages derived from Bing search results. The training set contains approximately 500K pairs of query and relevant passages; each query has one relevant passage, on average. Non-relevant passages are also provided as part of the training dataset.

MacAvaney et al. [6] demonstrated that fine-tuning BERT-based relevance classifiers on Med-MARCO, a medical subset of MS MARCO where only queries containing medical terms are kept, improves relevance ranking in the biomedical domain. We note similar results in the CORD-19

[2] `http://pyserini.io/`
[3] `http://anserini.io/`

corpus [14] in our own work [8, 17]. Hence, we choose to use monoT5-3B that was fine-tuned on MS MARCO and then fine-tuned (again) on Med-MARCO. We fine-tuned our monoT5-3B model on the MS MARCO training set with a constant learning rate of 10^{-3} for 10K iterations with class-balanced batches of size 128. For the second Med-MARCO fine-tuning, we trained for 1K steps using the same batch size and learning rate. We use a maximum of 512 input tokens and one output token. In the MS MARCO passage dataset, none of the inputs have to be truncated when using this length. Training monoT5-3B takes approximately 160 hours overall on a single Google TPU v3-8.

At inference time, the pointwise reranker considers the top segments from our BM25 keyword search as the retrieval units.

2.3 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 [10], which also uses T5. Within the framework of "learning to rank", duoT5 can be characterized as a "pairwise" approach, while monoT5 can be characterized as a "pointwise" approach [5]. 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 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 is more or less relevant 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. The number of inferences performed per query is given by the number of candidate pairs, i.e., $k_1(k_1 - 1)$. 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 arrive at an improved list of candidates R_2.

The pairwise reranker, duoT5, which also uses T5-3B, is trained in the same manner as monoT5. During inference, however, we increased the maximum input tokens 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.

2.4 H_3: Maximal Marginal Relevance

We rerank R_2, the candidates from the duoT5 stage, using the maximal marginal relevance (MMR) algorithm [3], which aims to balance sentence relevance (measured using the duoT5 scores) against redundancy from higher-ranked sentences. The final output R_3 is built incrementally with the highest-scoring candidate according to the following equation being added each iteration:

$$\text{MMR} = \arg \max_{d_i \in R_2 \setminus S} \left[\lambda \cdot \text{Sim}_1(d_i, q) - (1 - \lambda) \cdot \max_{d_j \in S} \text{Sim}_2(d_i, d_j) \right] \tag{2}$$

where Sim_1 and Sim_2 are similarity functions. Sim_1 compares the central sentence of candidate segment d_i to the query q. Sim_2, on the other hand, compares the central sentence of d_i with other central sentences from the current MMR result set S. Finally, λ is a parameter that controls the relevance–diversity tradeoff.

In our system, Sim_1 is the score assigned to the query-segment pair computed in the pairwise reranking stage, while Sim_2 is the cosine similarity between the BM25 vectors of the two segments. We also experimented with Sim_2 being the cosine similarity of TF-IDF vectors, but we found it less effective based on the Normalized Discounted Novelty Score (NDNS) scoring function in the preliminary round.

Run	NDNS-Partial	NDNS-Relaxed	NDNS-Exact
(1) Median	0.3377	0.3387	0.3802
(2) Max	0.3700	0.3709	0.4207
(3) Run 1 ($\lambda = 0.375$)	0.3381	0.3390	0.3880
(4) Run 2 ($\lambda = 0.420$)	0.3404	0.3412	0.3901
(5) Run 3 ($\lambda = 1.000$)	0.3284	0.3292	0.3755

Table 1: Mean (across questions) NDNS scores for EPIC-QA Task A (Expert) Primary Round

Run	NDNS-Partial	NDNS-Relaxed	NDNS-Exact
(1) Median	0.3142	0.2858	0.2845
(2) Max	0.3662	0.3675	0.4143
(3) Run 1 ($\lambda = 0.750$)	0.3593	0.3607	0.4065
(4) Run 2 ($\lambda = 0.700$)	0.3662	0.3675	0.4143
(5) Run 3 ($\lambda = 1.000$)	0.3382	0.3395	0.3825

Table 2: Mean (across questions) NDNS scores for EPIC-QA Task B (Consumer) Primary Round

Since the EPIC-QA tasks require submission of the top 1000 results for each question, the remaining $1000 - k_2$ segments were ordered by their monoT5 scores. Only the central sentence of each segment is returned in the final combined output R_3 (i.e. without context sentences). This mitigates the effect of the segment length penalty in the NDNS scoring function, In our current framework, this serves as the input to computing the final evaluation metrics.

3 EPIC-QA

For all six runs of both EPIC-QA tracks, we use the same multi-stage ranking pipeline. We set k_0, k_1, and k_2 to 10K, 50, and 50, respectively, following our experiments on the MS MARCO document ranking task.

Our submitted runs only varied in the choice of the MMR λ parameter. For both tracks, one of the runs set λ as one; doing so effectively disables the MMR stage by prioritizing accuracy at all costs (i.e., reverting to the vanilla Mono-Duo-T5 setting). The other two runs use λ values that yielded the highest effectiveness in the preliminary round.

Expert task runs 1, 2, and 3 use λ values of 0.375, 0.42, and 1, respectively. Consumer task runs 1, 2, and 3 use λ values of 0.75, 0.7, and 1, respectively. We found that the preliminary round consumer task benefits far less from diversity than those of the expert task since higher λ values maximize the objective NDNS.

It is worth emphasizing that other than this MMR λ parameter selection, all of our submissions are zero-shot, i.e., our models are trained solely on MS MARCO, as we did not adjust any model weights based on preliminary round data.

4 Results

A summary of results for the primary round of EPIC-QA Task A (Expert QA) and Task B (Consumer QA) are shown in Tables 1 and 2, respectively. The organizers evaluate the runs with three variations of Normalized Discounted Novelty Score (NDNS), a metric they devised. Exact NDNS penalizes long answers and nugget repetition, relaxed NDNS only nugget repetition, and partial NDNS performs no such penalization. Tables 1 and 2 present mean NDNS scores averaged across all questions in the run. Rows 1 and 2 show the median and maximum NDNS scores across all submissions. The next three rows show mean NDNS scores across all questions for each of $h_2$oloo's runs.

Based on these results, it is clear that sequence-to-sequence based retrieval techniques are effective for question answering tasks such as Epidemic QA. Even without MMR reranking, the "Mono-Duo-T5" pipeline achieves NDNS scores slightly below the median scores for the expert task and significantly above the median scores for the consumer task.

For both tracks, the addition of an MMR stage with λ values tuned using the preliminary round judgments results in non-trivial score increases of up to 0.03 for all NDNS variants. This improvement demonstrates that MMR is an effective method of improving answer diversity. For the expert task, answers generated by pipelines with MMR achieved scores higher than the median but lower than the maximum. For the consumer task, answers generated by the best pipeline with MMR achieved the highest score among all EPIC-QA submissions.

We avoid directly comparing our expert task scores to our consumer task scores since NDNS is a function of the number of nuggets. However, for our pipelines, we observe better relevance and/or diversity scores for the consumer task than the expert task when compared to the median score of other teams' submissions. Based on discussion at TAC 2020, this discrepancy can mostly be explained by our lack of fine-tuning on preliminary round judgments other than selecting suitable λ values for MMR. The top teams in the expert track performed such fine tuning. Since the expert judgments include more nuggets than the consumer judgments, approaches performing such fine-tuning would have benefited more for the expert track.

5 Conclusions

We have described our submissions to the Epidemic QA track of TAC 2020. Our standard pipeline "Mono-Duo-T5" has previously demonstrated strong zero-shot transfer capabilities on various domains such as news articles (Robust04) [7], biomedical articles (TREC 2020 Precision Medicine Track) [9], and COVID-specific scientific articles (TREC-COVID) [10, 17]. These results were yet again confirmed in the Epidemic QA track, in which our pipeline achieved competitive results with minimal adaptations. We further demonstrate that MMR is an effective method at improving answer diversity. Despite this success, we note that our pipeline performs better in the consumer track compared to the expert track relative to other systems. The cause of this difference remains unknown and warrants further investigation.

6 Acknowledgements

This research was supported in part by the Canada First Research Excellence Fund, the Natural Sciences and Engineering Research Council (NSERC) of Canada, and the Waterloo–Huawei Joint Innovation Laboratory. Additionally, we would like to thank Google for computational resources in the form of Google Cloud credits.

References

[1] Z. Akkalyoncu Yilmaz, C. L. A. Clarke, and J. Lin. A lightweight environment for learning experimental IR research practices. In *Proceedings of the 43rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2020)*, pages 2113–2116, 2020.

[2] 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.

[3] J. Carbonell and J. Goldstein-Stewart. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1998)*, pages 335–336, 1998.

[4] J. Lin, X. Ma, S.-C. Lin, J.-H. Yang, R. Pradeep, and R. Nogueira. Pyserini: An easy-to-use Python toolkit to support replicable IR research with sparse and dense representations. *arXiv:2102.10073*, 2021.

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

[6] S. MacAvaney, A. Cohan, and N. Goharian. SLEDGE: A simple yet effective baseline for COVID-19 scientific knowledge search. *arXiv:2005.02365*, 2020.

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

[8] R. Pradeep, X. Ma, R. Nogueira, and J. Lin. Scientific claim verification with VerT5erini. *arXiv:2010.11930*, 2020.

[9] R. Pradeep, X. Ma, X. Zhang, H. Cui, R. Xu, R. Nogueira, and J. Lin. H_2oloo at TREC 2020: When all you got is a hammer... Deep Learning, Health Misinformation, and Precision Medicine. In *Proceedings of the Twenty-Ninth Text REtrieval Conference (TREC 2020)*, 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(140):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, 2018.

[14] L. L. Wang, K. Lo, Y. Chandrasekhar, R. Reas, J. Yang, D. Burdick, D. Eide, K. Funk, Y. Katsis, R. Kinney, Y. Li, Z. Liu, W. Merrill, P. Mooney, D. Murdick, D. Rishi, J. Sheehan, Z. Shen, B. Stilson, A. Wade, K. Wang, N. X. R. Wang, C. Wilhelm, B. Xie, D. Raymond, D. S. Weld, O. Etzioni, and S. Kohlmeier. CORD-19: The COVID-19 Open Research Dataset. *arXiv:2004.10706*, 2020.

[15] 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.

[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] E. Zhang, N. Gupta, R. Tang, X. Han, R. Pradeep, K. Lu, Y. Zhang, R. Nogueira, K. Cho, H. Fang, and J. Lin. Covidex: Neural ranking models and keyword search infrastructure for the COVID-19 Open Research Dataset. *arXiv:2007.07846*, 2020.