Full-Collection Search with Passage and Document Evidence: Maryland at the TREC 2021 Conversational Assistance Track

Xin Qian and Douglas W. Oard

University of Maryland, College Park
{xinq,oard}@umd.edu

Abstract. The University of Maryland (UMD) team submitted four runs to the automatic rewrite settings of the track, exploring three ideas: (1) indexing both document-scale and passage-scale features, (2) using sharding to scale dense retrieval to large collections, and (3) combining results from sparse and dense methods using re-ranking and result fusion. Compared with the three-stage baseline pipeline of query rewriting using T5-base, document retrieval using BM25, and passage re-ranking using monoT5, UMD Run #1 modifies the second stage to retrieve passages rather than documents. UMD Run #2 retrieves documents in the second stage, but augments each second-stage document with additional document-scale evidence. UMD Run #3, our best run, fuses the final output of three runs: UMD Run #1, UMD Run #2, and the organizer-provided baseline. UMD Run #4, fuses results from TCT-ColBERT (a distilled ColBERT model) passage retrieval with results from BM25 document retrieval as the second stage. The TCT-ColBERT passage retrieval uses sharding to accommodate the large collection size. In each case, the first stage is the organizer-provided baseline query rewriter, and the third stage is a re-implementation of the baseline's third stage, but using monoBERT-large.

1 Introduction

The TREC Conversational Assistance Track (CAsT) is a shared task to study conversational information seeking, where an information system engages in conversational exchanges with human users to help satisfy information needs [2]. The user guides a human-machine conversation in CAsT, who can freely choose to introduce a topic, continue on that topic, or shift from one topic to another. Interpreting a current question in the context of conversational interaction is natural for humans, but challenging for machines. CAsT operationalizes this interpretation process as query rewriting. The goal is to rewrite a question in a way containing all of the necessary contexts to answer that question. In 2020, the best result for the automatic canonical condition (i.e., for automatic query rewriting based on canonical responses) was 0.493 for NDCG@3, quite close to the best result for the manual condition (i.e., for manually rewritten queries), which was 0.530 for NDCG@3 [3].

This success motivated development by the track organizers of a three-stage baseline pipeline in CAsT 2021 to which participating systems can be compared. In that pipeline (detailed in Section 3.1), the first stage is an automatic question rewriter, implemented using a T5-base model trained on the CANARD question answering dataset,[1] and the third stage is a pointwise passage re-ranker implemented using Mono-T5 model trained on MS MARCO [9]. For our experiments, we adopted the baseline's first-stage question rewriter. We also implemented a simple pointwise re-ranker using monoBERT and consistently used that as our third stage in our four submitted runs, and we experimented with a different second-stage full-collection retriever. In keeping with the track guidelines, we measure the relative effectiveness of our second-stage full-collection retriever using the end-to-end ranking quality of the full three-stage pipeline as an extrinsic measure of retrieval effectiveness for the conversational assistance task (as detailed in Section 4.2).

In this work, we explore two approaches to improve the second (full-collection retrieval) stage. In one approach (Run #2), we augment the content representation with additional document-level evidence (title terms and tokenized URL terms) and then use BM25 ($k1 = 4.46$, $b = 0.82$, consistent with the organizer-provided baseline) for full-collection ranked retrieval. In the other approach (Run #4), we use a TCT-ColBERT bi-encoder (a distilled ColBERT model) to perform late-interaction ranked retrieval using dense representations in a way that is sufficiently efficient to be performed over the full collection. Given the

[1] `https://github.com/daltonj/treccastweb/tree/master/2021/baselines`

sizes of the collection searched in CAsT, we use sharding to enable parallel processing. In this run, we augment each passage with the same document-level evidence as in Run #2, adding the same evidence for each passage from the same document. Traditional and neural methods have complementary strengths, so in Run #4 we fuse results from TCT-ColBERT (a distilled ColBERT model) with BM25 ($k1 = 4.46$, $b = 0.82$) results using weighted sum fusion before third-stage re-ranking. To encourage diversity, we compute these BM25 results without using the additional document-level evidence.

We compare each of these approaches to two baseline approaches. The full-collection search in our re-implemented low baseline (Run #1) is passage-level BM-25 ($k1 = 0.82$, $b = 0.68$). Our results show that Run #2 and Run #4 both statistically significantly outperform Run #1 by Mean Average Precision (MAP), although (because of our use of monoBERT-large rather than monoT5) our Run #1 yields results numerically (but not statistically significantly) below the organizer's baseline by every measure in Table 3). As a high baseline, our Run #3, uses reciprocal rank fusion to combine the ranked lists from Run #1, Run 2 and the organizers' baseline, statistically significantly outperforming all three of those component approaches by normalized Discounted Cumulative Gain (nDCG@500). Run #4 also statistically significantly outperforms both Run #1 and Run #2 by nDCG@500. Finally, we also locally scored three post hoc runs, including one in which we augmented both the second-stage retriever from Run #1 and the third-stage re-ranker with the additional document-level evidence. As shown in Table 4, that post hoc run statistically significantly outperforms Run #1 by nDCG@500, thus further illustrating the potential benefit of indexing title and URL terms. Our other two post hoc runs serve to illustrate the substantial benefit of the baseline (first-stage) question rewriting that we have used.

The remainder of this paper describes our techniques in greater detail. We start with the problem settings for TREC CAsT, then provide details on our systems, followed by results and analysis.

2 Task

In the submission category *automatic rewrite*, specifically, with the *raw utterance and canonical responses*, the input is a raw utterance U_k, the conversational context prior to the current k-th utterance consisting of raw utterances $U' = \{U_1, U_2, \ldots, U_{k-1}\}$, and canonical system responses to each of those utterances denoted $R' = \{R_1, R_2, \ldots, R_{k-1}\}$. For each raw utterance U_k, the task is to retrieve a list of passages $P_k = \{p_{k_1}, p_{k_2}, ...\}$ as relevant responses to the utterance. While an initial retrieval pass, the second stage of our three-stage pipeline, can use traditional IR techniques for efficiency, a competitive system will presumably do a final-pass, often the third-stage of our pipeline, using a more expensive re-ranker, where the system will learn and do inference with a re-ranker model $\mathbb{M}$ that scores the probability of a passage p being relevant in the context of the current utterance U_k, denoted as $\mathbb{M}(rel = 1|p, U_k, U', R')$. The collection to be retrieved from is in general heterogeneous, constructed from several document collections $C = \{C_1, C_2, ...\}$. For CAsT 2021, there are three such collections: an English Wikipedia dump, Version 1 of the MS MARCO document collection, and the Washington Post (WaPo) collection, where WaPo is a new addition this year. Table 1 summarizes some statistics for these three parts of the CAsT 2021 collection.

Table 1: Statistics of the CAsT 2021 collection.

	Wikipedia	MS MARCO	WaPo
# Documents	5.0M	3.2M	0.7M
# Passages	20.0M	22.0M	3.8M
Avg # Passages per Document	4.0	7.0	5.3

3 System Design

As noted above, all of our systems are patterned on the three-stage baseline pipeline in the track guidelines:[2] (1) query rewriting using T5-base; (2) document retrieval using BM25 followed by segmentation to passages; (3) passage re-ranking using monoT5. In this section, we describe the components we developed and the ways we used those components together in our submitted runs.

3.1 Baseline Architecture: Query rewriting, retrieval, and re-ranking

The baseline architecture uses a T5-base model [8] to generate automatic question rewrites. All of our systems do the same—when given a query, along with the conversational context (including canonical responses), we rewrite the query with the same pre-trained T5-base question rewriter,[3] publicly available on Huggingface. [4] This T5 rewriter is trained on the CANARD dataset [4], which includes 40K questions to train models for question-in-context rewriting. Each instance in the CANARD dataset contains a question together with prior questions and answers, and a human-created rewritten question.

As input to the rewriter, we provide the current utterance (user question), all previous utterances, and (at most) three most recent canonical system responses. We concatenate these inputs using the special separator token (|||), and the model then generates a rewritten query.

With the rewritten query, our simplest systems then perform full-collection retrieval using BM25, returning the top-1000 indexed units. We have two options for the indexed units, corresponding to two granularities:

– Run #1: **Passages** are segmented from documents as pre-processing, using the same passage chunker as in the organizers' baseline, then indexed, and retrieved using BM25 with default Anserini parameters ($k1 = 0.82$, $b = 0.68$, Figure 1).
– Run #2: **Full Documents** (without segmentation) are indexed, and retrieved using BM25 with organizers' baseline parameters ($k1 = 4.46$, $b = 0.82$, Figure 1). The retrieved documents are then chunked into passages in the same way as in Run #1. Note that because 1000 documents are retrieved, this results in more than 1000 passages.

Our third stage re-ranks the list of passages generated by the second stage using a pointwise monoBERT-large model that was pre-trained on MS MARCO. We used monoBERT-large rather than a monoT5 model used in the organizers' baseline because our pilot experiments showed limited improvements from using monoT5-base rather than monoBERT-large. In retrospect, a better choice might have been either (1) the monoT5-large model, which was found to outperform monoBERT-large by the winning team at CAsT 2020 [3], or (2) the duoBERT-large two-stage pipeline for leveraging both pointwise and pairwise training that yielded strong results for another high-scoring team at CAsT 2020 [5].

3.2 Augmenting with Additional Document-Level Evidence

In Run #2 and Run #4 we experiment with adding additional document-level features to the second-stage (full-collection) retrieval process. Each collection includes a URL and some form of title for each document. In the Washington Post collection, the title is the headline of the news story. In the Wikipedia dump, the title is the HTML title field for the Wikipedia page. For the MS MARCO collection, we used the title field formatted in the raw collection tsv file. Table 2 lists examples. We tokenized both titles and URLs using Lucene's default English analyzer.

The University of Waterloo (which was called to our attention by this year's TREC Deep Learning Track) reported substantial improvements (around 20 points absolute on recall@100 on dev set queries) from the use of additional content (which in their case also included section headers; we did not use the section headers in the augmented MS MARCO collection). This report was our inspiration for exploring the use of additional document-level evidence for all three CAsT collections. The inspiration also motivated

[2] `https://github.com/daltonj/treccastweb/tree/master/2021/baselines`
[3] Note, however, that in section 4.3 we show that our rewriter actually generates different rewrites!
[4] `https://huggingface.co/castorini/t5-base-canard`

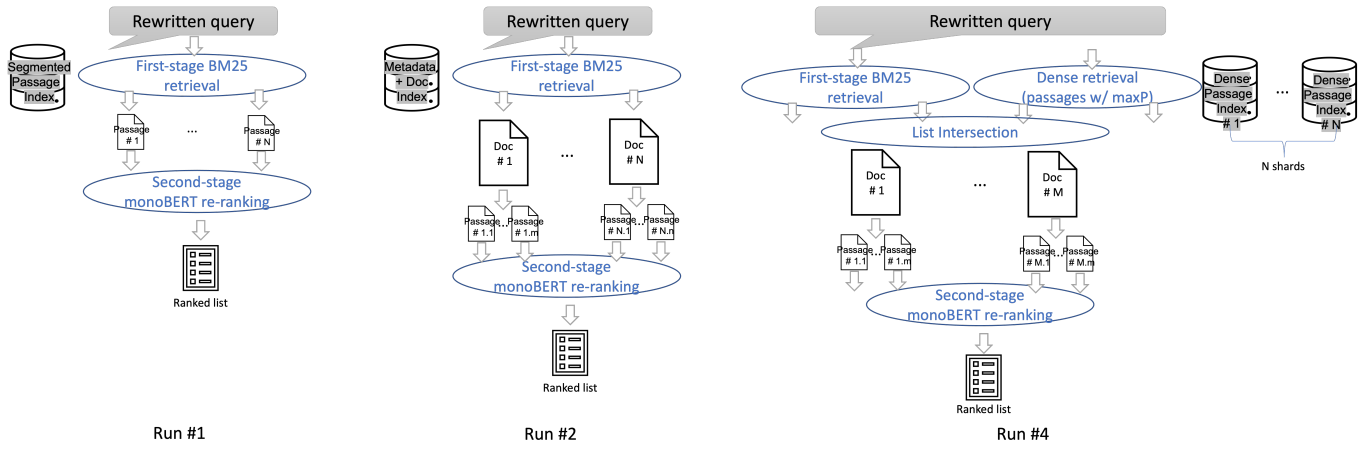

Fig. 1: Compared with the baseline pipeline of question rewriting (T5-base), document retrieval (BM25), and then passage re-ranking (monoBERT-large), our Run #1 modifies the second-stage indexing unit to be passages rather than documents. Run #2 indexes documents in the second stage, but augments each document with document-level evidence. Run #3, our best run (not shown here), fuses the final output of Runs #1, #2, and the organizer-provided baseline. Run #4 fuses TCT-ColBERT (a distilled ColBERT model) passage retrieval with BM25 document retrieval as the second stage, using sharding to accommodate the large collection size.

a post hoc experiment in which we also used this additional document-level evidence during (third-stage) re-ranking. This additional evidence might help in two ways: it might add terms not present in the text, or it might serve to reinforce (i.e., up weight) important terms present in both the text and the additional evidence.

Collection	**URL**	**Title/headline**
Wikipedia	`https://en.wikipedia.org/w/index.php?title=Origin%20of%20the%20domestic%20dog&oldid=908221312`	Origin of the domestic dog
MS Marco	`https://www.sciencedaily.com/releases/2014/01/140107102634.htm`	Cancer Statistics 2014: Death rates continue to drop
WaPo	https://www.washingtonpost.com/sports/colleges/danny-coale-jarrett-boykin-are-a-perfect-1-2-punch-for-virginia-tech/2011/12/31/gIQAAaW4SP$_s$tory.html	Danny Coale, Jarrett Boykin are a perfect 1-2 punch for Virginia Tech

Table 2: Examples of additional document-level evidence from titles and URLs.

3.3 Dense Retrieval and Sharding

Neural ranking methods using dense representations have been shown to achieve better recall than traditional ranking methods that rely on sparse representations, such as the BM25 model that we used in Run #1 and Run #2 [6]. Recently, fairly effective neural bi-encoders that are sufficiently efficient for use with moderately large collections have been introduced. Scaling such approaches up to collections of the size of CAsT 2021 still requires some parallelism, however. We therefore combined sharding with an efficient neural bi-encoder. The key idea in this approach is to perform shallow-depth retrieval on shards, using an efficient approximate nearest neighbor on dense representations that are computed separately for each query and each document. The hope is that this would improve recall over that achieved using sparse methods, although achieving that benefit depends on the relevant documents being reasonably well distributed across the shards. Sharding also reduces GPU memory requirements, making it possible to fit each shard into our compute infrastructure's 200GB memory limit.

A dense encoder based on the Transformer model has length limitations on its input, so we first divide documents into passages as in Run #1. We then augment each passage with the additional document-level evidence for the document from which the passage was extracted. Next, we divide the passage collection into 200 shards of equal size, with 17 shards from the Washington Post collection, 100 shards from MS MARCO, and 82 shards from Wikipedia. When performing this sharding, we respect passage boundaries, but not document boundaries, and we assign passages to shards in order, without randomization.

For the specific dense encoder, we used the TCT-ColBERT model, which is publicly available on Huggingface.[5] This is a distilled version of ColBERT that replaces ColBERT's MaxSim operation with a less expensive dot product, improving efficiency while retaining most of the ColBERT model's effectiveness. ANN search is performed with FAISS, the dense vector similarity search library. This process produces a score for each passage. We then calculate each document score using maxP (i.e., the largest of its passage scores). Independently, BM25 (on document terms only, with no augmentation) also produces a score for each document. We combine the two scores using weighted CombSUM, giving 100% of the weight to the Augmented TCT-ColBERT MaxP score and 10% of the weight to the BM25 score, as recommended in the TCT-ColBERT documentation.[6]

Unlike our other submitted runs, in Run #4 we perform third-stage re-ranking on a per-shard basis, for all passages from the top-scored documents in each shard. Because we want a final depth-1000 ranking over the full collection, we then essentially take the union of the top-scoring 50 passages from each of the 200 shards. Although the same passage can not appear in two shards, we actually perform this union by doing CombMAX fusion using the polyfuse library.[7] The procedure is illustrated in Figure 2.

Stage three re-ranking was performed on shards rather than the union of the collection for reasons of implementation convenience, and we expect this effect to be small in practice. Because we use the same pointwise monoBERT-large re-ranker for all our submitted runs, performing third-stage re-ranking on a per-shard basis is guaranteed to produce the same final ranking as would have resulted from re-ranking the union of the shards, down to at least rank 50. Below rank 50, quantization effects could result in the omission of some passages that might otherwise have been highly ranked.

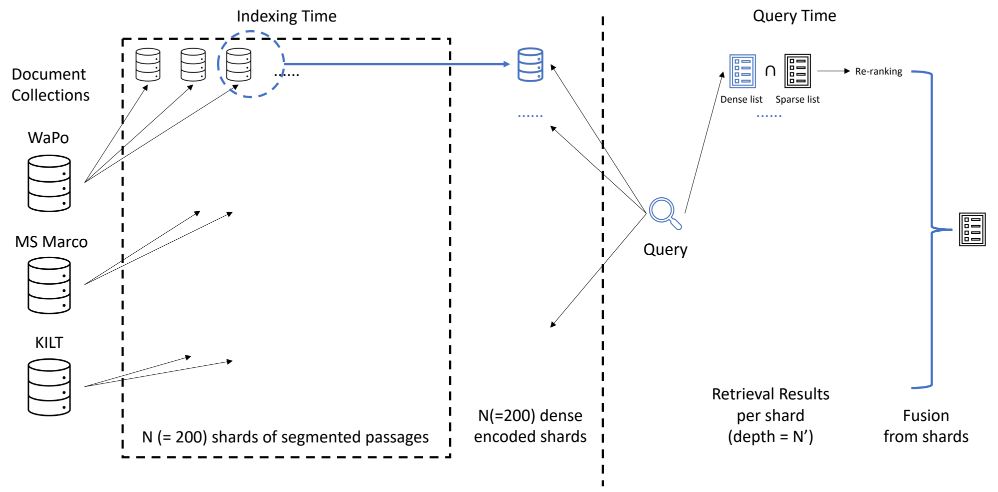

Fig. 2: An illustration of Run #4, using sharding and combining dense with sparse retrieval.

[5] castorini/tct$_c$olbert-v2-hnp-msmarco

[6] No normalization is performed before we apply weighted CombSUM; in future work we plan to more fully explore the design space for fusion techniques.

[7] https://github.com/rmit-ir/polyfuse

3.4 Fusion

Fusing the results from two or more runs is often an effective technique to improve ranking quality [7]. We tried two fusion techniques.

Run #3: To establish a high baseline, we combine the results from Run #1, Run #2, and the organizer's baseline. We use reciprocal rank fusion (RRF) with $k = 60$, which has been shown to be fairly robust [1].

Run #4: As described in Section 3.3, sharding produces multiple ranked lists, one per shard. We do weighted CombSUM fusion within each shard to combine the results from dense retrieval (using TCT-ColBERT) and sparse retrieval (using BM25). The recombination of results from multiple shards is also a fusion operation [10], although a simple one using CombMAX because the shards are logically disjoint.

3.5 Summary of Submitted Runs

Submitted runs are four system variants built from combinations of the above techniques, as Figure 1 illustrates.

- Run #1: Baseline BM25 implementation with passage index;
- Run #2: BM25 with document index, augmented with additional document-level evidence;
- Run #3: Reciprocal rank fusion of Run #1, Run #2, and the organizer's baseline;
- Run #4: Fused dense and sparse retrieval with sharding.

4 Results and Analysis

In this section, we describe the evaluation process, present and discuss our results.

4.1 Evaluation

We submitted a ranked list of passages for each run, but inconsistencies in passage definitions made passage-based evaluation impractical this year. The organizers thus chose instead to perform relevance judgments and evaluation at document scale. This was done by mapping ranked lists from passages to documents using MaxP (i.e., by replacing passage identifiers with the document identifier from that passage and then deduplicating the resulting document ranking by removing lower-ranked duplicates). This resulted in shallower judgment pools, and it required some changes to the relevance judgment guidelines to avoid penalizing the presence of extraneous information. We have implemented these changes to evaluate two post hoc runs that we report with (first-stage) question rewriting omitted, and a third post hoc run in which we augmented the passage representation used for (third-stage) re-ranking with additional document-scale evidence. Our other results are reported as they were received from the track organizers.

The primary evaluation measure is normalized Discounted Cumulative Gain at position 3 (nDCG@3), which focuses only on the top 3 results, and which gives decreasing weight to results in positions 2 or 3. nDCG@5, nDCG@500 and Mean Average Precision at 500 (MAP@500) are also reported to characterize results even lower in the ranked list. All evaluation measures are reported as averages over the computed measure for each query (i.e., for each evaluated conversational turn).

4.2 Results

Table 3 summarizes the results for our submitted runs. We also include best and median from the TREC website, averaged over judged topics, as well as results from the organizers' baseline. '*' indicates a statistically significant improvement over the organizers' baseline using a two-sided paired t-test at $p < 0.05$ with Bonferroni correction. Similarly, '†' indicates a statistically significant improvement over Run #1, and '$\diamond$' over Run #2. Locally scored results of post hoc runs without the query formulation stage are also shown

Table 3: Results for submitted runs. '*' indicates significant improvement over the organizers' baseline, '†' over Run #1, '◇' over Run #2. Results without question rewriting were scored locally.

	nDCG@3	nDCG@5	nDCG@500	MAP@500	Judging priority
Best	0.8067	0.7716	0.7668	0.5483	—
Median	0.3820	0.3872	0.4499	0.2431	—
Baseline	**0.4357**	0.4265	0.5036	0.3135	—
Run #1	0.3875	0.3764	0.4625	0.2619	2
wo rewriting	0.2216	0.2100	0.2613	0.1435	—
Run #2	0.3985	0.3904	0.4784	$0.2811^{\dagger}$	2
wo rewriting	0.1888	0.1878	0.2375	0.1257	—
Run #3	$0.4252^{\dagger}$	$\mathbf{0.4275}^{\dagger\diamond}$	$\mathbf{0.5388}^{*\dagger\diamond}$	$\mathbf{0.3221}^{\dagger\diamond}$	3
Run #4	0.3768	0.3744	$\mathbf{0.5116}^{\dagger\diamond}$	$0.2841^{\dagger}$	1

as contrastive conditions.

Effect of first-stage question rewriting: The improvements from question rewriting shown in Table 3 are quite substantial. Comparing Run #1 or #2, with and without (wo) rewriting, illustrates this simple ablation study. Question rewriting brings very substantial absolute improvements of 0.16 or 0.21 in nDCG@3 for Run #1 or Run #2, respectively, with similar benefits observed for other measures.

Benefit of augmentation with document-level evidence: Table 4 shows a benefit from augmenting passages with terms from the title and URL of the document from which the passage was drawn. In this case, the additional evidence was used in both the second-stage retrieval and the third-stage re-ranking. The 0.029 absolute apparent improvement in nDCG@3 is not statistically significant, but the comparable improvements in nDCG@5 and nDCG@500 are both statistically significant. One caveat to this analysis is that computing evaluation measures on documents rather than passages, as has been done this year, may favor the addition of document-level evidence. Note that we can not tell with this study design whether the benefit results from the effects of augmentation on the second-stage retrieval, the third-stage re-ranking, or both. Comparing to Run #2, which used augmentation only in second-stage retrieval, is confounded by the fact that Run #2 indexed documents rather than passages for stage two. Note also that because the same document-level evidence is indexed repeatedly for each passage from the same document, there is some additional storage overhead for the index, (in this case, a relative increase of 9%).

Benefit of fusion: Comparing Run #3 to Runs #1, #2, and the organizer's baseline, we see that the fused run (Run #3) statistically significantly outperforms every individual run in the combination by nDCG@500, with an absolute improvement of 0.035 over the best of the three constituent runs. No statistically significant improvement over the best of the three constituent runs was observed for nDCG@3 or nDCG@5, however, indicating that the benefit observed in nDCG@500 occurs later in the ranked list.

Table 4: Benefit of retrieval from a non-augmented passage index (Run #1) vs. a passage index augmented with document title and URL terms (scored locally). '‡' indicates a statistically significant improvement over Run #1.

	Index size	nDCG@3	nDCG@5	nDCG@500	MAP@500
Original index (Run #1)	66G	0.3875	0.3764	0.4625	0.2619
Augmented index + re-ranking	72G	**0.4162**	$\mathbf{0.4100}^{\ddagger}$	$\mathbf{0.4815}^{\ddagger}$	**0.2770**

Benefit of combining dense and sparse retrieval: As Table 3 shows, Run #4, which combines dense and sparse retrieval, achieves a statistically significant improvement of 0.033 over Run #2, our most closely comparable sparse-only run, by nDCG@500. However, we note no improvement from Run #2 to Run #4 in nDCG@3 or nDCG@5, indicating that the benefits we observe in nDCG@500 are coming later in the ranked list. Note also that as we have shown with Run #3, fusion can be useful even when both runs use sparse retrieval, and our study design is not able to separate that effect from specific benefits that may be accruing from using dense retrieval in the set of runs being fused.

Efficiency: The end-to-end throughput for all three stages of the Run #1 (passage-as-indexing-unit) and Run #2 (document-as-indexing-unit) pipelines are 47.3 and 304.6 seconds per query, respectively. As Table 2 shows, the average number of passages per topic varies between 4 and 7, and this factor of 6 or so in end-to-end processing time is thus dominated by the fact that the document-as-indexing-unit pipeline results in the generation of more passages that require re-ranking.

4.3 Case Study

Table 5 shows some examples of automatic question rewriting and a comparison with the manually rewritten questions.

Turn	Original	Manual	Official	Ours
1	I just had a breast biopsy for cancer. What are the most common types?	I just had a breast biopsy for cancer. What are the most common types of **breast cancer?**	What are the most common types of cancer **in regards to breast biopsy?**	What are the most common types of **cancer**?
2	Once it breaks out, how likely is **it** to spread?	Once it breaks out, how likely is **lobular carcinoma breast cancer** to spread?	Once the cancer breaks out, how likely is it to spread?	
3	How deadly is **it**?	How deadly is lobular carcinoma in situ?	How deadly is **LCIS?**	How deadly is **lobular carcinoma** in situ?
4	**What? No, I want to know about** the deadliness of lobular carcinoma in situ.		What is the deadliness of lobular carcinoma in situ?	I want to know about the deadliness of lobular carcinoma in situ.
5	**Wow, that's better than I thought.** What are common treatments?	**Wow, that's better than I thought.** What are common treatments for lobular carcinoma in situ?	What are common treatments for lobular carcinoma in situ?	
6	How does it behave differently from PLCIS?	How does LCIS behave differently from PLCIS?	How does LCIS behave differently from **ALH** and PLCIS?	How does lobular neoplasia behave differently from PLCIS?
7	What makes lobular cancer distinct?		What makes lobular cancer distinct from PLCIS?	What makes lobular cancer distinct?
8	For the first stage, what are the alternatives to surgery?	For the first stage, what are the alternatives to surgery for invasive lobular cancer?	For the first stage of IdC, what are the alternatives to surgery?	For the first stage of **lobular carcinoma**, what are the alternatives to surgery?
9	No, I meant for lobular.	**No, I meant -** what are the alternatives to surgery for stage 1 invasive lobular cancer	**No, I meant for lobular carcinoma.**	
10	Does freezing work?	Does freezing tumors work **as an alternative to surgery for stage 1 invasive lobular cancer?**	Does freezing work for breast cancer?	

Table 5: Original questions for one example topic, compared with the manually rewritten "silver standard", the automatically rewritten question from the organizers' baseline, and our automatically rewritten question. We note that in seven of the ten turns, our automatic rewrite differs from the automatic rewrite used in the organizers' baseline, despite our effort to closely replicate the usage of a T5-base model from their description. IdC = Invasive ductal carcinoma.

While the manual rewrites demonstrate desired rewriting effects, the automatic rewrites achieve some useful improvements:

Utterance Simplification: Question rewriting simplifies colloquial utterances into a single-sentence, interrogative sentence, as in turns 1 and 4, making terms re-weighted.

Pronoun Resolution: Question rewriting expands pronouns such as replacing *it* with *lobular carcinoma breast cancer*, or expanding the abbreviation *LCIS*. However, this effect only appears from turn 3, but not earlier, possibly due to our context concatenation approach in which inference is on three previous canonical responses.

Clause Addition: Question rewriting adds clauses based on the context to complement the original utterance, where the meaning of the whole query is substantiated. Examples include turns 1 and 8, where we see *the most common types* ***of cancer***, and *the first stage* ***of lobular carcinoma***.

5 Conclusion

In this work, we present our submissions to TREC CAsT 2021. Run #3, our best run, fuses the final output of Run #1 doing indexing on passages, Run #2 doing retrieval on documents and augmenting each document in the index with additional document-level evidence, and the organizer's baseline. That run achieves a comparable nDCG@3 to the organizer's baseline, and it statistically significantly outperforms the organizers' baseline by nDCG@500. Run #4 is also interesting because it fuses dense retrieval with sparse retrieval, using sharding to accommodate the large collection size, thus beginning our exploration of an important design space.

References

1. Aslam, J.A., Montague, M.: Bayes optimal metasearch: a probabilistic model for combining the results of multiple retrieval systems. In: Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 379–381 (2000)
2. Culpepper, J.S., Diaz, F., Smucker, M.D.: Research frontiers in information retrieval: Report from the third strategic workshop on information retrieval in Lorne (SWIRL 2018). In: ACM SIGIR Forum. vol. 52, pp. 34–90. ACM New York, NY, USA (2018)
3. Dalton, J., Xiong, C., Callan, J.: Trec cast 2019: The conversational assistance track overview. arXiv preprint arXiv:2003.13624 (2020)
4. Elgohary, A., Peskov, D., Boyd-Graber, J.: Can you unpack that? learning to rewrite questions-in-context. In: Empirical Methods in Natural Language Processing (2019)
5. Gemmell, C., Dalton, J.: Glasgow representation and information learning lab (GRILL) at the conversational assistance track 2020. In: Proceedings of the Twenty-Ninth Text REtrieval Conference. NIST Special Publication, vol. 1266. National Institute of Standards and Technology (NIST) (2020)
6. Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., Yih, W.t.: Dense passage retrieval for open-domain question answering. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 6769–6781 (Nov 2020)
7. Lillis, D., Toolan, F., Collier, R., Dunnion, J.: Probfuse: a probabilistic approach to data fusion. In: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 139–146 (2006)
8. Lin, S.C., Yang, J.H., Nogueira, R., Tsai, M.F., Wang, C.J., Lin, J.: Conversational question reformulation via sequence-to-sequence architectures and pretrained language models. arXiv preprint arXiv:2004.01909 (2020)
9. Nogueira, R., Jiang, Z., Pradeep, R., Lin, J.: Document ranking with a pretrained sequence-to-sequence model. In: Findings of the Association for Computational Linguistics: EMNLP 2020. Association for Computational Linguistics, Online (Nov 2020)
10. Voorhees, E.M., Gupta, N.K., Johnson-laird, B.: The collection fusion problem. In: Proceedings of the Third Text Retrieval Conference (TREC-3). pp. 95–104 (1995)