Comparative Study of Conversational Search Engine Retrieval Pipelines

Group ID: #5

Til Mohr
tv.mohr@stud.uis.no

Ishaac Ourahou
i.ourahou@stud.uis.no

ABSTRACT

In our master's degree program, we delved into the realms of information retrieval and text mining. Our group project focuses on developing a system capable of searching for specific passages within a conversation. This entails considering the historical context of previous exchanges when interpreting the current user's query. The system reformulates the user's query to better align with the conversation's overarching topic and then seeks the most relevant passages from a vast collection of 8.8 million documents, specifically from the MS MARCO document collection. We initially create a baseline information retrieval pipeline, which serves as our foundational reference. We then explore advanced techniques and methods in the field of conversational search engines. Our report concludes with a comparative analysis between the baseline system and our advanced retrieval mechanisms, outlining the strengths and shortcomings of our approaches, and offering conclusions.

KEYWORDS

information retrieval, conversational search engine, conversational search system

1 INTRODUCTION

With the rise of conversational agents like Amazon Alexa and Apple Siri, the ability to accurately and relevantly respond to user requests has become increasingly important. These agents are designed to interact fluidly with users, and the essence of their functionality rests on providing precise information in their responses.

Our project centers on building a conversational search engine, a system that not only addresses the user's immediate question but also takes into account previous queries and the ongoing conversation's topic. This provides a more holistic approach to the user's information needs. Initially, we will construct a basic architecture, establishing our baseline system. This baseline will serve as a comparison point against more sophisticated techniques. It's required, however, that even this foundational system meets or exceeds the performance of the reference system set by our project guides.

Our research will utilize the MS MARCO dataset, a comprehensive collection of 8.8 million documents. Efficiently retrieving relevant documents from this vast pool necessitates an indexing process, which we will execute using the `pyterrier` API.

Section 2 will articulate the specific problem we aim to address. In Section 3 we will explore literature relevant to our retrieval pipelines. We will then elaborate on the methodologies behind our retrieval pipelines in Sections 4, 5.1, 5.2, and 5.3. Our report will conclude in Section 6, where we evaluate and compare the effectiveness of our systems, highlighting the strengths and potential areas of enhancement in our advanced approaches.

The codebase for our project can be found on GitHub[1].

2 PROBLEM STATEMENT

This project focuses on developing a Conversational Search Engine (CSE). What distinguishes a CSE from a traditional search engine is its inherent context-aware nature. In a CSE, queries asked by a user within a session are semantically interconnected. Rather than treating each query in isolation, a CSE leverages information from both the previous queries and the system's responses to those queries to retrieve relevant passages. It must cater to the continuous thread of a dialogue, while also being agile enough to accommodate sudden shifts in conversation topics. For every user query, the goal is to generate a ranked listing of the top 1000 passages out of an expansive 8.8 million document collection.

A vital component that sets a CSE apart is its query rewriting mechanism. This feature reformulates the presented query to better reflect the ongoing conversation topic. Figure 1 offers a schematic representation of how the various stages of a CSE retrieval pipeline interact. Analogous to traditional search engines, a CSE also requires prior indexing of the document collection, enabling computationally efficient retrieval and ranking processes.

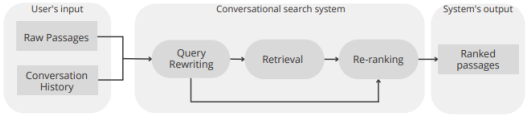

Figure 1: Overall system's pipeline architecture

3 RELATED WORK

In this Section, we delve into the research encompassing the realms of conversational search engines and the broader area of information retrieval. While certain highlighted studies do not directly cater to conversational search engines or explicit information retrieval, their techniques remain invaluable in various stages of the conversational retrieval process.

3.1 Pseudo-Relevance Feedback by Query Expansion

Oftentimes, queries entered by users are too short and include too little hint for the search engine to retrieve all relevant documents [13]. To counter this, the user-generated query must be reformulated to better reflect the user's informational needs. One approach

[1]URL: https://github.com/CodingTil/2023_24---IRTM---Group-Project

to this is known as query expansion using relevance feedback: Assume we have a set of documents relevant to the user-generated query. One can then reformulate the query such that the retrieval system retrieves documents similar to the set of relevant documents, i.e. documents that are probably also relevant to the user's query. This expansion is accomplished by selecting keywords from the set of relevant documents and appending those to the query. Different strategies for this exist [13].

However, in a real setting, we do not have a set of relevant documents. Instead, we can generate a set of documents that are hopefully relevant to the user's query, and then follow the same process. Since the relevance of the generated set of documents is unknown, this approach is called pseudo-relevance feedback. In order to retrieve this set, we can use the original user entered query to retrieve only the top p documents. If p is very small, it is likely that all those documents are relevant to the query. Expanding the query using those documents will then likely result in a better retrieval of all relevant documents [13].

One particular technique for this is known as `RM3` [1]. `RM3` uses relevance-based language models [5] to estimate the relevance of a word based on the word's probability in the language models of the (pseudo-)relevant documents. This relevance of the word to the query is then interpolated with the original query language model to avoid the query drifting too far away from the original query. The resulting language model is then used to generate a new query. Overall, the `RM3` approach can be parameterized in the number of documents to use for the pseudo-relevance feedback, the number of terms to add to the query, and the interpolation parameter [1, 3, 5].

3.2 Text-to-Text Transfer Transformer

The vast domain of natural language processing (NLP) revolves around the understanding of natural language, whether presented in text or speech form. NLP aspires to equip computers with the capability to grasp the depth of human language and harness this understanding to execute a range of tasks, such as text summarization, machine translation, and question answering. Given the diverse nature of these tasks in terms of their input, output, and underlying challenges, developing a unified model proficient across the entire spectrum poses a significant challenge.

Enter the Text-to-Text Transfer Transformer (`T5`) [11]. This work by Raffel et al. introduces transfer learning in NLP, aiming to craft a versatile model that can be used for any NLP problem. In essence, `T5` models first learn the basics of language. Then, they're sharpened for particular tasks using targeted data. It's common to find models that have been trained in this manner for any specific NLP problem.

3.3 `doc2query` Document Expansion

Traditional retrieval techniques, such as `BM25`, rely primarily on term occurrences in both queries and documents. However, they often overlook the semantics of the content. As a result, documents that may be semantically relevant to a query might be scored as non-relevant due to differences in syntax or terminology. Dense retrieval methods, which emphasize semantic similarities between texts, can address this problem but are computationally taxing during retrieval.

A notable solution to this is the `doc2query` method proposed by Nogueira et al. [9]. It employs a text-to-text transformer to convert documents into queries. By generating and appending a few of these transformed queries to the original document, classical retrieval methods show significantly improved performance. This is because these additional queries often capture semantic nuances similar to those in the actual query [7, 9, 10]. Importantly, `doc2query` shifts the computational load to the indexing phase, ensuring minimal performance lag during retrieval. By leveraging the `T5` model, the authors further enhanced the query generation quality, leading to the variation known as `docTTTTTquery`, `doc-T5query`, or `doc-2query-T5` [7].

3.4 `SPARTA` Sparse Retrieval

`SPARTA`, introduced by Zhao et al. [16], is an approach in the field of sparse information retrieval. At its core, it works by encoding documents into sparse representations during the indexing phase. These representations not only capture the document's actual content but also incorporate terms that are semantically resonant, even if they're not present in the document. This underlying principle echoes the rationale of approaches like `doc2query` and dense retrieval models.

Yet, where `SPARTA` differentiates itself is in its retrieval phase. Unlike dense retrieval models, it retrieves relevant documents using straightforward index lookups, mirroring lexical retrieval strategies like `BM25` [16]. However, in real-world applications, `SPARTA` faces challenges: Several other models, including `BM25` and `doc2query-T5`, surpass it in ranking quality. Additionally, its generated index is substantially larger compared to alternatives like `doc2query-T5` [12].

3.5 `ANCE` Dense Retrieval

Dense retrieval is a newer approach in information retrieval. Traditional methods, like `BM25`, rely on specific term matching in high-dimensional spaces. These methods can sometimes miss the deeper meaning or context because they focus on explicit word matches. In contrast, dense retrieval uses continuous vector spaces to embed both queries and documents. In this approach, items that have similar meanings are closer together in the vector space, even if they don't share exact terms.

`ANCE` (Approximate Nearest Neighbor Negative Contrastive Estimation) is an innovation in dense retrieval introduced by Xiong et al. [14]. The main idea behind `ANCE` is to fix a problem in the training of dense retrieval models. Often, the irrelevant documents or "negatives", i.e. documents that should be ranked the lowest for a given query, used during training don't match the kind of irrelevant documents that come up during real-world testing. This mismatch can weaken the model's performance. To address this, `ANCE` updates its training data using an Approximate Nearest Neighbor (ANN) index. This index gets refreshed during the training process, making sure that the irrelevant documents chosen for training are similar to what the model will see in real-world scenarios. Thanks to this, `ANCE` retrieval pipelines perform better than those with other dense and sparse retrieval methods. In fact, `ANCE` achieves nearly the same accuracy as some sophisticated methods but is about 100 times faster [14].

3.6 monoT5 & duoT5 Rerankers

monoT5 and duoT5 are neural re-rankers, also developed by Nogueira et al., which attempt to inject semantic understanding into the retrieval process [6, 8]. Using the T5 model, they re-rank a list of documents based on their semantic relevance to a given query. Specifically, monoT5 processes a query and a single document, outputting a relevance score. In contrast, duoT5 considers a query and two documents, determining which document is more relevant. Although duoT5 offers a more nuanced ranking, its pairwise comparison method makes it computationally heavier. Hence, a staged re-ranking approach is proposed: first using monoT5 for the top k documents and subsequently applying duoT5 to a smaller subset, the top l, where $l \ll k$ [8, 10].

3.7 Expando-Mono-Duo Design Pattern

The same research team introduced a design pattern for integrating the above tools into retrieval pipelines, termed the Expando-Mono-Duo design pattern [10]. Here's how it works: During indexing, doc2query-T5 is employed to enhance document representation and better the initial retrieval results from methods like BM25. The retrieved results are then re-ranked with monoT5. A selected top tier from this list undergoes another re-ranking using duoT5. Trials show that this composite approach leads to marked improvements in result quality across multiple evaluation metrics [10].

3.8 Conversational Query Rewriting

Conversational search engines distinguish themselves from standard search engines by determining document relevance through the entirety of a conversation, not just the immediate query. In conversational contexts, subsequent questions often lean on prior interactions, implying that previous questions and answers must be factored in when fetching relevant documents. However, there's also a need to cater to conversation shifts where the immediate query doesn't relate to preceding exchanges. Blindly considering the entire conversational history in such cases could be detrimental to retrieval accuracy.

Elgohary et al. address this challenge with an innovative approach [4]. They suggest reshaping the current query based on the overarching conversation. This reformulated query is designed to function autonomously within conventional retrieval pipelines. In essence, this technique extends the utility of standard search engines to conversational question-answering scenarios by introducing a preceding conversational query rewriting stage.

Employing text-to-text transformers, like T5, can be instrumental in achieving this rewrite. These models are trained to reformulate the user entered query, factoring in the conversational context. Studies validate the effectiveness of this approach, highlighting its capacity to enhance the retrieval accuracy of traditional search engines in conversational contexts [2, 4, 17].

3.9 WaterlooClarke Conversational Search Engine

In the third year of the TREC Conversational Assistance Track[2], the standout system was WaterlooClarke. It used a complex retrieval pipeline, as outlined in [15, 17]. The process starts with a T5 component that rewrites the user's query. This step transforms a context-dependent question into one that's more general, free from the specific conversation's context, as outlined in Section 3.8. Following this, the rewritten query is passed into two parallel retrieval stages: firstly, sparse retrieval through BM25, enhanced with pseudo-relevance feedback, and, secondly, dense retrieval powered by ANCE. To accommodate both approaches, WaterlooClarke employs two indices: a lexical one for BM25 and an ANN index for ANCE. Once both retrieval stages conclude, their results are fused into a single list of ranked documents. This ranked list undergoes further refinement, first with monoT5 and then a narrower re-ranking using duoT5 for the top-tier documents [15, 17].

4 BASELINE METHOD

Our baseline method is inspired by the baseline method presented by Łajewska et al. [17]. It is structured in the following sequence:

(1) T5 Query Rewriting
(2) BM25 Retrieval
(3) Re-ranking
   (a) Re-ranking using monoT5
   (b) Top-document re-ranking using duoT5

4.1 T5 Query Rewriting

In conversation search engines, query rewriting is the crucial component to include the semantics of the conversation history into the currently asked query, which results into a singular rewritten query that can be fed into the retrieval pipeline.

For this purpose, we include all the previously rewritten queries $q'_0 \dots q'_{n-1}$ of our conversation, as well as the response r_{n-1} of the CSE to the previous rewritten query q'_{n-1} into the current query q_n. This is done by concatenating the previous rewritten queries and the response into a single string:

$$q'_n \coloneqq q'_0\text{<sep>} \dots \text{<sep>}q'_{n-1}\text{<sep>}r_{n-1}\text{<sep>}q_n$$

This approach resembles the approach taken by Łajewska et al. [17].

We have found through experimentation that this mere concatenation is insufficient to produce satisfiable results: The concatenated string is too long, and too much focus during the later retrieval is being put on r_{n-1}, which make responses sudden topic changes impossible.

Driven by these insights, we have turned our attention to other query rewriting techniques. Pyterrier provides the SequentialDependence query rewriting method[3]. We have found, however, that this rewriter also does not produce the desired results.

Subsequent exploration led us to the T5 neural query rewriter trained for conversational question rewriting, see Section 3.8. With this method, q'_n closely mirrored q_n, subtly infusing it with the conversation's context, particularly when no drastic topic alterations were identified. A valuable by-product was the concise nature of

[2]URL: https://www.treccast.ai/
[3]URL: https://pyterrier.readthedocs.io/en/latest/rewrite.html#sequentialdependence

the rewritten query, a departure from the growing length observed previously. Since retrieval latency is a critical factor, we utilized a smaller T5 model: castorini/t5-base-canard[4]

4.2 BM25 Retrieval

We settled on the BM25 retrieval method, a commonly used formula in the realm of information retrieval, for its simplicity and its deployment in the reference system, allowing for direct comparisons.

4.3 Re-ranking

The re-ranking stage of our baseline system consists of two stages: First, the top 1000 documents retrieved by the BM25 retrieval method are re-ranked using the monoT5 re-ranker. Afterwards, the top 50 documents of the previous re-ranking stage are rearranged using the duoT5 re-ranker, see Section 3.6. The precise count of documents subject to re-ranking at each stage is a hyperparameter of our system, allowing to balance computational cost and result quality. These re-rankers were implemented in the pyterrier_t5 library.[5] Again, since a low latency of our retrieval pipeline is crucial to us, we utilized smaller T5 models: castorini/monot5-base-msmarco[6] for monoT5 and castorini/duot5-base-msmarco[7] for duoT5.

5 ADVANCED METHOD

In this paper we propose two separate extensions to the baseline model at different stages of the retrieval pipeline, making way for a total of three advanced retrieval methods. The first extension is the integration of the RM3 query expansion method, see Section 5.1. The second extension is the integration of the doc2query document expansion method, see Section 5.2. Both extensions can be combined, resulting in the final advanced retrieval method doc2query + RM3, see Section 5.3. In the following we will describe these individual methods in detail.

5.1 Incorporating Pseudo-Relevance Feedback into Our Baseline

Recognizing the substantial performance enhancements associated with pseudo-relevance feedback, we felt compelled to integrate a query expansion mechanism into our baseline retrieval method, see Section 4. Our choice fell upon the RM3 query expansion technique, well-established for its robustness and acceptance within the information retrieval community. For a deeper dive into its mechanics and principles, readers are directed to Section 3.1.

In the Pyterrier framework, the setup requires that any query expansion follows an initial retrieval phase. This initial retrieval fetches the top p documents, forming the foundation for subsequent query expansion by n words using RM3. With the query expanded, it's then passed into a secondary retrieval phase to retrieve the final document set for the end-user. And, to fine-tune the output, we again apply re-ranking using both monoT5 and duoT5.

Henceforth, we'll label this integrated retrieval approach as "baseline + RM3", which is structured as follows:

(1) T5 Query Rewriting
(2) BM25 Retrieval
(3) RM3 Pseudo-Relevance Feedback Query Expansion
(4) BM25 Retrieval
(5) Re-ranking
  (a) Re-ranking using monoT5
  (b) Top-document re-ranking using duoT5

5.2 Document Expansion Method

To improve the results of our baseline system, we made the decision to integrate a document expansion mechanism, and after careful consideration, our choice landed on doc2query-T5, see Section 3.3. This approach combines the power of document expansion with the capabilities of the T5 sequence-to-sequence model to enhance the effectiveness of our information retrieval system.

The core idea behind the doc2query-T5 model is to dynamically generate specific questions or queries that are closely related to the content of a given document. These generated questions are then seamlessly incorporated into the document. The goal of this process is to expand the document's content, thereby providing additional information that can significantly improve the effectiveness of our information retrieval system. By generating relevant queries based on the document's content, we are essentially expanding the scope of potential search terms, enabling our system to better capture the user's intent and find more relevant documents.

The integration of the T5 model allows us to transform the document into highly relevant queries tailored to the content of the document. This is achieved by utilizing a T5 model fine-tuned on this task of understanding the contextual relationships within the document and generate queries that effectively summarize the key points of the document. In particular, we use the castorini/doc2query-t5-large-msmarco[8] model.

The use of doc2query-T5 will be added to our baseline, which will otherwise remain unchanged. In particular, doc2query-T5 can be seen as a preprocessing step to indexing, where first m queries can be generated for each document in the collection, which then will be appended to the original document to form the input for the indexing stage. Therefore, this pipeline follows the Expando-Mono-Duo Design Pattern, as introduced in Section 3.7. The system architecture for this pipeline, which we will refer to as "doc2query", will therefore take the following form:

(0) doc2query-T5 Document Expansion
(1) T5 Query Rewriting
(2) BM25 Retrieval
(3) Re-ranking
  (a) Re-ranking using monoT5
  (b) Top-document re-ranking using duoT5

5.3 Extending the Document Expansion Method with Pseudo-Relevance Feedback

The combined "doc2query + RM3" approach represents a powerful retrieval pipeline that leverages the strengths of both document expansion and query expansion. By seamlessly integrating document expansion through doc2query-T5 and the established

[4]URL: https://huggingface.co/castorini/t5-base-canard
[5]URL: https://github.com/terrierteam/pyterrier_t5
[6]URL: https://huggingface.co/castorini/monot5-base-msmarco
[7]URL: https://huggingface.co/castorini/duot5-base-msmarco
[8]https://huggingface.co/castorini/doc2query-t5-large-msmarco

pseudo-relevance feedback method RM3, we are able to improve our search capabilities in a number of ways.

This advanced retrieval method allows us to create more contextually relevant queries, starting with the generation of document-specific questions and refining user queries using T5. The subsequent search phase, guided by BM25, reduces the number of candidate documents. RM3 then uses these candidates to create additional queries, thereby broadening the search field. In a final round of searching using BM25, we broaden the set of documents. To further improve the quality of the results, our monoT5 and duoT5 re-ranking steps ensure that the most relevant documents come out on top. This approach offers a holistic solution that not only improves accuracy but also explores a wider range of potentially relevant documents, providing users with an improved and efficient information search experience. Ultimately, our architecture is a combination of RM3 and doc2query-T5, see Section 3, and will take the following form:

(0) doc2query-T5 Document Expansion
(1) T5 Query Rewriting
(2) BM25 Retrieval
(3) RM3 Pseudo-Relevance Feedback Query Expansion
(4) BM25 Retrieval
(5) Re-ranking
  (a) Re-ranking using monoT5
  (b) Top-document re-ranking using duoT5

6 RESULTS

The individual methods were evaluated on the MS MARCO document collection and the following provided files: `queries_train.csv`, comprising of a list of queries grouped together into several conversation sessions, and `qrels_train.txt` that contains the relevance assessments for the training queries. Our evaluation focused on a suite of metrics:

- Recall at 1000 (R@1000)
- Mean Average Precision (MAP)
- Mean Reciprocal Rank (MRR)
- Normalized Discounted Cumulative Gain (NDCG_Cut@3)

These metrics were calculated using the `trec_eval` tool:

```
trec_eval -c -m recall.1000 -m map -m recip_rank
-m ndcg_cut.3 -l2 -M1000 qrels_train.txt
{GENERATED_TREC_RUNFILE}
```

As stated in Section 4, the baseline method can be parameterized in a few different ways. For this evaluation, we utilized the following configurations: The document retrieval (BM25) used the default parameters from pyterrier[9] to retrieve the 1000 most-relevant documents for each query. All 1000 documents were then re-ranked using the monoT5 re-ranker. Because of the high computational cost of the duoT5 re-ranker, only of those 1000 documents the best 50 documents were then reordered using this re-ranker.

For the first extension of the baseline, the baseline + RM3 method, we utilized the same configuration for these components. The RM3 query expansion component was parameterized to expand the query by 26 terms, using the top 17 documents retrieved by the initial BM25 search.

[9]URL: https://pyterrier.readthedocs.io/en/latest/terrier-retrieval.html

The doc2query and doc2query + RM3 strategies built upon the baseline and baseline + RM3 designs. The distinction is in the initial indexing stage (refer to Sections 5.2 and 5.3). Keeping everything else constant for a fair comparison, the indexing stage was adjusted to generate 3 descriptive queries for each document. The subsequent documents incorporated these queries before indexing.

We ran our evaluations on a Google Cloud Compute Instance, powered by a single Nvidia L4 Tensor Core GPU. All retrieval pipelines exhibited similar retrieval times.

Table 1: Performance of the different methods on the MS MARCO document collection. Best results are in bold, second-best results are underlined.

	MAP	MRR	R@1000	NDCG_Cut@3
Reference System	0.07			0.09
Baseline	**0.2213**	**0.3698**	0.5656	**0.2807**
Baseline + RM3	0.2124	0.3600	0.5710	0.2673
doc2query	0.2109	0.3571	0.5735	0.2700
doc2query + RM3	0.2126	0.3498	**0.5790**	0.2721

As can be seen in table 1, our baseline method is at least as efficient as the reference system provided by the project owners. Thus, our baseline method fulfills the hard requirement of the project.

Furthermore, when we compare the baseline results with those of the more advanced methods, it becomes clear that the addition of RM3 and doc2query does not lead to the anticipated enhancement in retrieval effectiveness. Specifically, only in the R@1000 metric do we observe any significant gains over the baseline, with both the RM3 query expansion and the doc2query document expansion contributing modestly. The combination of both, doc2query + RM3, shows the most improvement in this particular metric. A similar trend is noticeable in the NDCG_Cut@3 metric, but here all advanced methods still fall short of the baseline. For the MAP metric, it's evident that RM3 alone has a less detrimental effect than doc2query, but their combination narrows the performance gap with the baseline. Lastly, the MRR metric suffers a decline with the addition of either RM3 or doc2query, with the lowest performance observed when both are combined. Comparing query expansion with document expansion, their effectiveness appears similar. However, their combined application is more effective, surpassing the other advanced methods in three out of the four metrics.

As for index generation, the baseline methods processed the 8.8 million documents of MS MARCO effortlessly, completing in less than an hour. On the other hand, doc2query required approximately six days for generating all descriptive queries. Subsequent indexing was again rapid, completed in under an hour. Both indices maintained the original document content for query rewriting purposes (see Section 3.8). The baseline methods produced a 4.2GB index, while the additional terms from doc2query slightly increased the index size to 4.5GB. Despite the larger size, retrieval speeds from the bigger index showed no significant variation.

7 DISCUSSION AND CONCLUSIONS

This study has introduced and scrutinized four conversational retrieval strategies for the MS MARCO document collection, building upon the baseline approach of Łajewska et al. [17]. Our baseline method proved to be as efficient, if not more so, than the reference system proposed by the project organizers.

Our findings underscore the consequences of augmenting the baseline with query and document expansion techniques. These extensions enable the retrieval system to find more relevant documents, as reflected in the R@1000 metric. However, this advantage comes with a compromise in the precise ranking of documents, observable in the MAP, MRR, and NDCG_CUT@3 metrics. One likely explanation is that these extensions not only fetch more relevant documents but also pull in more irrelevant ones that are falsely assessed as relevant in re-ranking. Despite this, both query and document expansion techniques have shown success in other retrieval pipelines, as discussed in Sections 3.1 and 3.3, necessitating further investigation into their specific implementations in our context.

If our hypothesis is correct, exploring dense retrieval techniques like ANCE (refer to Section 3.5) could be a promising direction for future research. Such techniques might enable us to source more relevant documents without relying on query or document expansion, potentially improving the ranking of retrieved documents.

In summary, while integrating both query and document expansion strategies leads to a significant increase in the number of relevant documents retrieved, this is not without its trade-offs. The silver lining is that these enhancements add negligible extra computational load during retrieval. However, there's a slight decline in the quality of result ranking. This is a vital consideration for conversational search engines, where immediate access to the most relevant information is crucial. In this light, our baseline approach stands out as a viable option, effectively balancing efficiency and effectiveness.

REFERENCES

[1] Nasreen Abdul-Jaleel, James Allan, W Bruce Croft, Fernando Diaz, Leah Larkey, Xiaoyan Li, Mark D Smucker, and Courtney Wade. 2004. UMass at TREC 2004: Novelty and HARD. *Computer Science Department Faculty Publication Series* (2004), 189.

[2] Raviteja Anantha, Svitlana Vakulenko, Zhucheng Tu, Shayne Longpre, Stephen Pulman, and Srinivas Chappidi. 2020. Open-domain question answering goes conversational via question rewriting. *arXiv preprint arXiv:2010.04898* (2020).

[3] Zhaofeng Chen, Naixuan Guo, Jiu Sun, Yuanyuan Wang, Feng Zhou, Sen Xu, and Rugang Wang. 2022. Pseudo-Relevance Feedback Method Based on the Topic Relevance Model. *Mathematical Problems in Engineering* 2022 (2022).

[4] Ahmed Elgohary, Denis Peskov, and Jordan Boyd-Graber. 2019. Can you unpack that? learning to rewrite questions-in-context. *Can You Unpack That? Learning to Rewrite Questions-in-Context* (2019).

[5] Victor Lavrenko and W Bruce Croft. 2017. Relevance-based language models. In *ACM SIGIR Forum*, Vol. 51. ACM New York, NY, USA, 260–267.

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

[7] Rodrigo Nogueira, Jimmy Lin, and AI Epistemic. 2019. From doc2query to docTTTTTquery. *Online preprint* 6 (2019), 2.

[8] Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. 2019. Multi-stage document ranking with BERT. *arXiv preprint arXiv:1910.14424* (2019).

[9] Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019. Document expansion by query prediction. *arXiv preprint arXiv:1904.08375* (2019).

[10] Ronak Pradeep, Rodrigo Nogueira, and Jimmy Lin. 2021. The expando-mono-duo design pattern for text ranking with pretrained sequence-to-sequence models. *arXiv preprint arXiv:2101.05667* (2021).

[11] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research* 21, 1 (2020), 5485–5551.

[12] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models. *arXiv preprint arXiv:2104.08663* (2021).

[13] Rekha Vaidyanathan, Sujoy Das, and Namita Srivastava. 2015. Query expansion strategy based on pseudo relevance feedback and term weight scheme for monolingual retrieval. *arXiv preprint arXiv:1502.05168* (2015).

[14] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *arXiv preprint arXiv:2007.00808* (2020).

[15] Xinyi Yan, CL Clarke, and Negar Arabzadeh. 2021. WaterlooClarke at the TREC 2021 conversational assistant track. In *The Thirtieth Text REtrieval Conference Proceedings, TREC*, Vol. 21.

[16] Tiancheng Zhao, Xiaopeng Lu, and Kyusong Lee. 2020. SPARTA: Efficient open-domain question answering via sparse transformer matching retrieval. *arXiv preprint arXiv:2009.13013* (2020).

[17] Weronika Łajewska and Krisztian Balog. 2023. From Baseline to Top Performer: A Reproducibility Study of Approaches at the TREC 2021 Conversational Assistance Track. In *European Conference on Information Retrieval (ECIR '23)*. Springer, 177–191. https://doi.org/10.1007/978-3-031-28241-6_12

A DIVISION OF WORK DURING THE PROJECT

Concerning the distribution of tasks on the project, we worked in a fairly classic way, trying to be as organized as possible to make sure we did things well. In general, we tried to work together on each part and then split the work when necessary.

For part G1, we both worked together on the baseline, thinking together about the architecture and how to code.

For the G2 part, given that it was purely research work, we divided it up in order to move forward correctly. At the same time, we constantly exchanged information on our research and what could be interesting to discuss in our report.

For part G3, we worked together on the advanced methods that we discussed in G2. We took each method one by one and implemented it in such a way that we could understand the code together and evaluate its effectiveness.