An Exploration Study of Mixed-initiative Query Reformulation in Conversational Passage Retrieval

Dayu Yang
University of Delaware
dayu@udel.edu

Yue Zhang
University of Delaware
zhangyue@udel.edu

Hui Fang
University of Delaware
hfang@udel.edu

Abstract

In this paper, we report our methods and experiments for the TREC Conversational Assistance Track (CAsT) 2022. In this work, we aim to reproduce multi-stage retrieval pipelines and explore one of the potential benefits of involving mixed-initiative interaction in conversational passage retrieval scenarios: reformulating raw queries. Before the first ranking stage of a multi-stage retrieval pipeline, we propose a mixed-initiative query reformulation module, which achieves query reformulation based on the mixed-initiative interaction between the users and the system, as the replacement for the neural reformulation method. Specifically, we design an algorithm to generate appropriate questions related to the ambiguities in raw queries, and another algorithm to reformulate raw queries by parsing users' feedback and incorporating it into the raw query. For the first ranking stage of our multi-stage pipelines, we adopt a sparse ranking function: BM25, and a dense retrieval method: TCT-ColBERT. For the second-ranking step, we adopt a pointwise reranker: MonoT5, and a pairwise reranker: DuoT5. Experiments on both TREC CAsT 2021 and TREC CAsT 2022 datasets show the effectiveness of our mixed-initiative-based query reformulation method on improving retrieval performance compared with two popular reformulators: a neural reformulator: CANARD-T5 and a rule-based reformulator: historical query reformulator(HQE).

1 Introduction

The TREC Conversational Assistance Track (CAsT) is a task to facilitate the study of conversational information systems, which are information systems that adopt a conversational modality to enable conversational exchanges between the system and its users. The main objective of conversational information seeking is to satisfy users' information needs in an evolutionary fashion, which is formalized or expressed through conversation turns. It can be beneficial for many information retrieval tasks, such as sophisticated information searching, exploratory information collecting, multi-turn retrieval task completion, and recommendation. Although conversation can also exhibit other types of interactions with different characteristics and modalities, such as clicks, multi-choice selections, and other forms of feedback [6], we mainly focus on natural language conversation in the Text REtrieval Conference(TREC) Conversational Assistance Track(CAsT). Specifically, following the problem settings of TREC CAsT, a user can initialize an open-domain information request to the system, and the system is expected to retrieve relevant passages from a gigantic corpus. During the conversation, the user is free to continue on the previous topic, provide feedback on the previously retrieved passage, or shift from one topic to another.

31th Text REtrieval Conference (TREC 2022), Virtual Conference.

The overall approach of us is a multi-stage retrieval architecture that contains four main stages: the query reformulation stage, the first-ranking stage, and the second-ranking stage with an affiliated stage called fusion. In addition to adopting existing query reformulation methods: CANARD-T5 and HQE[3], to enable the mixed-initiative interaction and make the interaction helpful to the query reformulation task, we design an algorithm to generate questions seeking clarification of three types of ambiguities in the raw queries: incomplete, reference, and descriptive. After the question is generated, it will be sent to the user. Once the answer from the user is received, the answer will be parsed by another algorithm. The new clarification information will be combined with the raw query to formulate the reformulated query.

2 Method

2.1 Multi-stage Retrieval Pipeline

First, in order to achieve state-of-the-art retrieval performance, we construct an efficient and effective multi-stage retrieval pipeline. Specifically, we use a four-stage cascade structure. The first stage will be the query reformulation stage, where we implement two popular query reformulation methods: CANARD-T5 rewriter and HQE [3] to eliminate ambiguities in the raw utterances. The pipeline we build can be illustrated in Figure 1.

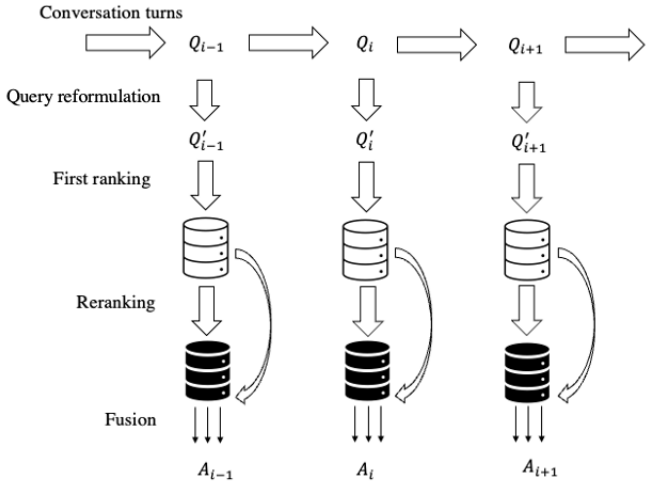

Figure 1: Demonstration of our multi-stage conversational text retrieval pipeline for creating non-mixed-initiative runs

To improve the efficiency of the multi-stage pipeline, instead of reformulating the query when we run the pipeline, we first implement query reformulations on all the queries and store the reformulated queries for later usage. This reduces the intensity of communication between the CPU and GPU and avoids unnecessary repeated query reformulation work when trying different ranking functions. For the CANARD-T5 rewriter [3] which we use for query reformulation, instead of using the parameters, we fine-tune it on TREC 2019 and 2020 data to further improve the reformulation on retrieval. The experiment on the TREC CAsT 2021 dataset shows our fine-tuned T5 rewriter outperforms the T5 with original weights.

After the ambiguities are resolved by the query reformulation stage, the first ranking stage generates the initial ranked document list and delivers it to the re-ranking stage. there will be γ_1 numbers of documents to be retrieved as the candidates for the following stage. We apply multiple ranking methods based on sparse methods and dense methods to overcome the limitation of the lexical and semantic matching capability of a single ranking function. Each $stage_i$ takes γ_{i-1} numbers of documents to ranking and outcome γ_i numbers of documents to the subsequent stage, where $\gamma_{i-1} \leq \gamma_i$. Specifically, we use BM25 ranking function as the sparse retrieval method we used in the first ranking stage. For the dense ranking in the first ranking stage, we use TCT-ColBERT [2] to independently encode queries and the documents and formulate the dense representations of

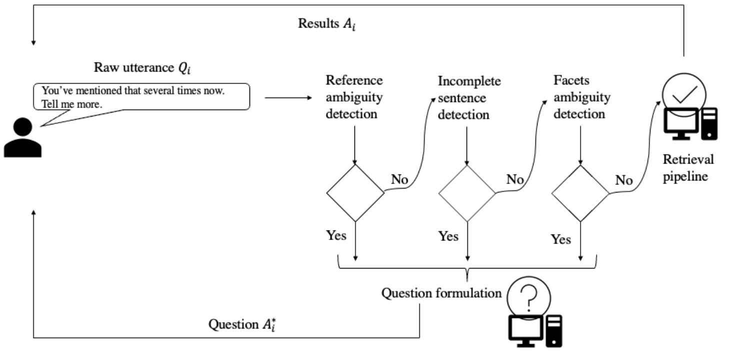
Results A_i
Raw utterance Q_i
You've mentioned that several times now. Tell me more.
Reference ambiguity detection
Incomplete sentence detection
Facets ambiguity detection
No
No
No
Yes
Yes
Yes
Retrieval pipeline
Question formulation
Question A_i^*

Figure 2: A workflow for asking clarifying questions in an open-domain conversational search system

documents and queries. After the first ranking stages are finished, we use a pointwise re-ranker MonoT5 [4] and a pairwise re-ranker DuoT5 [5] to re-rank the relevant documents (passages) that are outputted from the first ranking stage. Finally, we apply reciprocal rank fusion[1], which is efficient for combining the ranked lists obtained from re-ranking. (If an early fusion is applied, the fusion stage will behave before re-ranking and after the first ranking. After fusion, the final ranked document list will output as result. Figure 1 shows our multi-stage conversational text retrieval pipeline for creating non-mixed-initiative runs.

2.2 Mixed-initiative Query Reformulation

We observe that, for the TREC CAsT 2021 dataset, in some cases, certain ambiguities are not successfully clarified by the CANARD-T5 reformulators. Since CANARD-T5 is a generative model that samples the reformulated queries from high-dimensional distributions. It is hard for us to explicitly explore why sometimes CANARD-T5 fails to correctly clarify ambiguities. However, what we can observe is that: the automatic reformulator performs well on certain queries but not well on the rest. Therefore, with the introduction of the mixed-initiative task in the TREC CAsT 2022, we intend to explore the potential of clarifying the ambiguity with the help of users. In order to generate the question, we designed an algorithm to identify the ambiguity a raw query has and formulate the corresponding question.

Specifically, we design three questions that correspond to three types of ambiguities: references, descriptive, and incomplete sentences. When the algorithm detects an ambiguity, the system will generate a corresponding question from the template and send it to the user for further clarification. *Incomplete* ambiguity is defined as the raw query does not have any nouns. For example, the user may ask "How's that?" after the previous retrieval turns. *Reference* ambiguity is defined as the algorithm finding at least one pronoun in the raw query(and it is not at the beginning of the raw query). *Descriptive* ambiguity means the noun with a high BM25 score in the raw query does not have any descriptive information following it. For example, the raw query could be "What kind of innovation do we have?" which is missing the description of the noun "innovation". The descriptive information of "innovation" can be "innovation of US banks".

Since only one interaction is allowed for each raw query, if multiple types of ambiguities are detected, the algorithm will follow the priority order: incomplete $>$ reference $>$ descriptive to generate questions. Here are some more concrete examples from the TREC CAsT 2022 dataset, in *utterance 3-1, turn 142*, the raw query is *You've mentioned that several times now. Tell me more.*. The algorithm will first identify if this sentence is a complete query. In this case, the raw query is a complete query, so the algorithm will move on to detect the reference ambiguity. Since there is a pronoun "that" in the raw query, that means the raw query has a reference ambiguity. Then the algorithm will extract the pronoun "that" and ask the user, *What does "that" refer to in your raw query?*. If there is no reference ambiguity found by the algorithm, it will continue to try to identify whether raw utterances have descriptive ambiguities. Taking utterances 3-3, turn 132 as an example, the raw utterance is *How did other parties respond?*. The algorithm detected that an important word, *parties* does not have any

[1] We do not use early fusion. The fusion step is always employed after the second-stage ranking

descriptive information in the raw utterance. Therefore, the system will ask the user to specify the word *parties*. In this case, the answer received by the system is *parties of the Paris Agreement*.

After the answer of obtained, a reformulating algorithm will add the newly-obtained information from the answer to the original query. For an original query with the reference ambiguity, we expect the user's answer to be the entity a pronoun is referring to. So the pronoun will be directly replaced by the answer. For an incomplete original query, we expect the user's answer to be the complete query. So the entire original query will be replaced by the answer. For an original query with descriptive ambiguity, the algorithm will append the descriptive information to the corresponding noun or verb. In some cases, the user may refuse to answer the question, the answer will be "I don't know" if the user refuse to answer in TREC CAsT 2022. The algorithm will keep the original query intact if it meets the answer "I don't know".

In summary, the algorithm can enable mixed-initiative interaction with users while resolving the ambiguities that existing generative and classification query reformulation methods have difficulty resolving. The overall workflow of our mixed-initiative algorithm is shown in Figure 2.

Overall, the original automatic query reformulation steps of the multi-stage retrieval pipeline introduced in Figure 1 are replaced with the mixed-initiative query reformulation module, which means the change is made only in the "query reformulation" stage. The multi-stage conversational text retrieval pipeline we used for creating mixed-initiative runs for TREC CAsT 2022 can be illustrated in Figure 3.

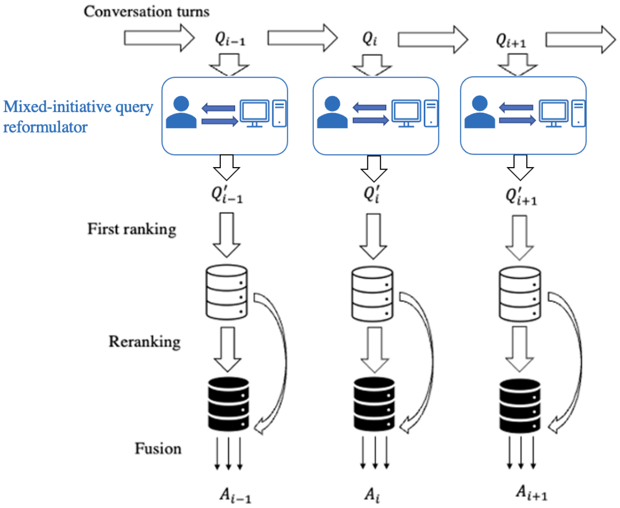

Figure 3: Demonstration of our multi-stage conversational text retrieval pipeline for creating mixed-initiative runs

3 Experimental Setup

3.1 CAsT Datasets

Since the qrel file of the TREC CAsT 2022 dataset is not available when we participate in the CAsT track. We finish experiments about hyperparameter tuning on the TREC CAsT 2021 dataset.

3.2 Query Reformulation Setup

The original T5 rewriter has trained on CANARD [3] dataset. Although many experiments from the runs of TREC CAsT 2021 show the knowledge T5 learns from CANARD transfers well to the TREC CAsT dataset, we still want to figure out *if it is beneficial to fine-tune the T5 rewriter on the previous TREC CAsT datasets: 2019 and 2020?* Therefore, we start with the weights that we borrowed from [1] and fine-tune the T5 on structured TREC 2019 and 2020 data. Table 1 shows:

Run Name	Recall@1000	Recall@500	MAP@2000	NDCG@3
The original T5 rewriter	0.5873	0.5502	0.1378	0.2335
Fine-tuned T5 rewriter	**0.6365**	**0.5892**	**0.1484**	**0.2525**

Table 1: The retrieval performance on TREC CAsT 2021 dataset of the reformulated queries reformulated by the original T5 rewriter from [1] and our fine-tuned one.

by using the query reformulated by the fine-tuned T5 rewriter, the retrieval performance on the first ranking stage can surpass the original T5 rewriter.

For the default setting of a T5 rewriter, it will consider all the canonical passages $A_{<i}$: $A_1, A_2, \ldots, A_{i-1}$ before the focal query Q_i. However, when applying it to the TREC CAsT dataset, the canonical passage is usually much longer than the canonical passage in the CANARD dataset. The excessive length brings up the issue that the total length of the input of T5 will sometimes surpass the maximum length of the token input of T5: 512. And T5 will cut off all the tokens after the 512_{th} token. This makes some latest users' utterances may be abandoned by the T5's tokenizer. However, the intuition is that the user's information needs are more likely to be contained in users' utterance $Q_{<i}$: $Q_1, Q_2, \ldots, Q_{i-1}$ instead of the canonical passages $A_{<i}$. Another issue is that the long canonical passages may bring T5 extra challenges to locate useful information from the context that contains the information about users' implicit information needs. Therefore, we think concatenating all previous canonical passages may harm the retrieval performance. The results of experiments on the TREC CAsT 2021 dataset are shown in Table 2. As we can see, the best-performing reformulated queries only take the most recent canonical passage into consideration.

No. Canonical Psg*	Recall@500	MAP@500	NDCG@500	NDCG@3
0	0.5330	0.1237	0.3384	0.2180
1	**0.5459**	**0.1356**	**0.3591**	**0.2474**
2	0.4688	0.1111	0.3012	0.2102
3	0.4688	0.1111	0.3012	0.2102

Table 2: The retrieval performances on TREC CAsT 2021 dataset of the reformulated queries considering the different numbers of canonical passages(*No. Canonical Psg = Number of canonical passages to consider in the T5 rewriter)

Another experiment we did was to explore "*if it is beneficial to consider not only the most probable output of the generative reformulator but other outputs*?" The intuition behind this experiment is that the target of a generative model is to generate the sentence with the highest probabilities instead of a sentence with more information that represents users' information needs. By our observation, we find, sometimes, the probability of generating the reformulated sentence: "*What is the price of the bike*?" is larger than the probability of "*What is the price of the sport bike of Trek*?" Therefore, we design an experiment to fuse top-probable sentences from the generative reformulator. The results are shown in Table 3. We can see that the recall can be largely improved if we consider multiple top-probable outputs and fuse them at the end of the first ranking stage. The results indicate that it may be beneficial to consider multiple generated sentences with the largest probabilities from a generative reformulator such as T5 rewriter.

Historical query reformulation(HQE) is a method that uses BM25 scores to identify if a word in the previous passage retrieval log is important or not. The words that are classified as important will be appended at the end of the raw query. We use the same BM25 threshold setting with the paper introducing HQE[3].

3.3 Tuning BM25 Parameters

For sparse ranking functions, we use the BM25 ranking function with the following parameter setting: k1=1.24, b=0.9 after hyperparameter tuning on the TREC CAsT 2021 dataset. Table 4 shows the improvement in retrieval performance using the aforementioned BM25 parameters compared with the default settings in the TREC CAsT 2021 dataset. For both sparse retrieval and dense retrieval in the first ranking stage, our pipeline will only return the top 2000 ranked passages to improve efficiency.

No. sentences to fusion	Recall@500	MAP@500	NDCG@500	NDCG@3
1	0.4854	0.1294	0.3252	0.2429
3	0.5663	0.1376	0.3662	0.2466
5	0.5793	0.1431	0.3758	0.2607
7	0.5845	**0.1503**	**0.3833**	**0.2722**
10	**0.6037**	0.1423	0.3804	0.2587

Table 3: The retrieval performances on TREC CAsT 2021 dataset of runs fusing different number of top-probable sentences in the first ranking stage (using BM25 as the ranking function)

	Recall@500	Recall@1000	Recall@3000
A non-tuning run	0.6664	0.7121	0.7790
The best run(k1=1.24, b=0.9).	**0.6730**	**0.7518**	**0.8101**

Table 4: The improvement in retrieval performance using the aforementioned BM25 parameters compared with the default settings in the TREC CAsT 2021 dataset

Also, during the first retrieval, instead of waiting for the retrieval process to finish for a single query, we delay all the retrieval processes for many reformulated queries and process them together to take full advantage of the multiprocessing capability of our CPU.

3.4 Re-ranking Setup

After the first ranking stage, we implement MonoT5 as the pointwise re-ranker and DuoT5 as the pairwise re-ranker. Due to the expensive computational requirement that come with the re-ranking process and re-rankers require online computation. We only use MonoT5 to re-rank the first 1000 documents and use DuoT5 to re-rank the first 200 documents from MonoT5.

3.5 Fusion Setup

After the re-ranking stage, we have a total of six re-ranking runs since we have three different versions of reformulated queries: G0, G1, and Hqe, and two different first ranking methods: sparse and dense ranking methods, where“G" stands for generative neural reformulator. In our case, we use the T5 rewriter specifically. We consider both “G0" and “G1", where “G0" stands for the most probable output and “G1" stands for the second most probable output. Although we observe from Table 3 that fusing more runs using different output of the generative reformulator could be beneficial to the retrieval performance. Due to the time restriction and for the simplicity of our method, we do not fuse runs using top-probable sentences generated by the reformulator for creating our submitted runs. The following chapter will include a detailed description of the four submitted runs and how we finished the fusion step.

4 Submitted Runs

For TREC CAsT 2022 participation, our team finally submitted two automatic runs: UDInfo-best2021, UDInfo-onlyd and two mixed-initiative runs: UDInfo-mi-b2021, UDInfo-onlyd-mi. Since we cannot

Method	Dataset	MI	Recall@1000	MAP@1000	NDCG@3
udinfo_best2021	CAsT 2021	✗	**0.949**	**0.287**	**0.246**
udinfo_onlyd		✗	0.904	0.286	0.240
udinfo_best2021	CAsT 2022	✗	0.681	0.181	0.325
udinfo_onlyd		✗	0.651	0.178	0.348
udinfo_best2021_mi		✓	**0.771**	**0.246**	**0.452**
udinfo_onlyd_mi		✓	0.729	0.243	0.450

Table 5: The retrieval performance on the TREC CAsT 2021 and 2022 datasets of all submitted runs. MI stands for “mixed-initiative".

obtain user feedback for the TREC CAsT 2021 dataset, we only report the two pipelines using CANARD-T5 or HQE query reformulators. The details of how to create runs in 4 are described later.

What we can observe from 4 is that: first, although the two methods we used for creating runs have a higher Recall@1000 on the TREC 2021 dataset compared with the TREC 2022 dataset, we obtain higher NDCG@3 on the TREC 2022 dataset. This could indicate that the ranking methods we used in the first ranking stage, which mainly focused on increasing Recall, can handle the TREC 2021 dataset better than the TREC 2022. On the contrary, the ranking methods we used for the second-ranking stage, which mainly focuses on increasing NDCG, can handle the TREC 2022 dataset better than the 2021 one. Secondly, the methods using the mixed-initiative query reformulation module achieve much higher retrieval performance compared with the runs using CANARD-T5 and HQE, which indicates that incorporating mixed-initiative interaction into conversational passage retrieval systems has the potential to improve retrieval performance[2].

- Run #1 (UDInfo-best2021): reciprocal fusion of three top-ranked methods on NDCG@3 in the TREC CAsT 2021 dataset(The reason for only fuse top three runs is because fusing more runs can only harm the performance by experiment on the TREC CAsT 2021 dataset), which are:
  - Using “G1" as the query reformulation method; sparse first ranking stage; Pointwise and Pairwise re-ranking.
  - Using “G0" as the query reformulation method; dense first ranking stage; Pointwise and Pairwise re-ranking.
  - Using “Hqe" as the query reformulation method; dense first ranking stage; Only re-ranking on Pointwise method.
- Run #2 (UDInfo-mi-b2021): using the clarification answers from users after the system proactively elicits; other remains the same as Run #1.
- Run #3 (UDInfo-onlyd): reciprocal fusion of all three dense methods, which are:
  - Using “G1" as the query reformulation method; dense first ranking stage; Pointwise and Pairwise re-ranking.
  - Using “G0" as the query reformulation method; dense first ranking stage; Pointwise and Pairwise re-ranking.
  - Using “Hqe" as the query reformulation method; dense first ranking stage; Pointwise and Pairwise re-ranking.
- Run #4 (UDInfo-onlyd-mi): using the clarification answers from users after the system proactively elicits; other remains the same as Run #3.

5 Conclusion

In this paper, we introduced our multi-stage retrieval pipeline that can tackle conversational search tasks. Our pipeline consists of four stages: query reformulation, first ranking, re-ranking, and fusion. In addition to the multi-stage retrieval pipeline, we also introduced our implementation of mixed-initiative interaction on query reformulation, where we design an algorithm to generate questions and seek answers from users to explicitly resolve the ambiguities in the raw queries. In the future, we will explore more methods that can enable mixed-initiative interactions, which can possibly benefit retrieval performance in conversational search.

[2] Since some of the original answers we received have unexpected bad quality(for example, many answers are “This question is not related to my search."), we manually replace those bad-quality answers by mimicking the behavior of a user. The answer file, which includes the answers we used for query reformulation and our generated questions, can be found in this link.

References

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

[2] Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. “Distilling dense representations for ranking using tightly-coupled teachers”. In: *arXiv preprint arXiv:2010.11386* (2020).

[3] Sheng-Chieh Lin et al. “Multi-stage conversational passage retrieval: An approach to fusing term importance estimation and neural query rewriting”. In: *ACM Transactions on Information Systems (TOIS)* 39.4 (2021), pp. 1–29.

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

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

[6] Hamed Zamani et al. “Conversational information seeking”. In: *arXiv preprint arXiv:2201.08808* (2022).