Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.081336

ARTICLE

Conversational Query Reformulation with the Guidance of Retrieved Documents

Jeonghyun Park and Hwanhee Lee*

Department of Artificial Intelligence, Chung-Ang University, 84 Heukseok-ro, Dongjak-gu, Seoul, Republic of Korea

*Corresponding Author: Hwanhee Lee. Email: hwanheelee@cau.ac.kr

Received: 28 February 2026; Accepted: 05 May 2026; Published: 23 July 2026

ABSTRACT: Given a multi-turn conversational context and a raw user query, the goal of Conversational Query Reformulation (CQR) is to transform the query into a de-contextualized form that maximizes retrieval effectiveness for a downstream passage retriever. Conversational search seeks to retrieve relevant passages for the given questions in a conversational question answering system. Conversational Query Reformulation (CQR) improves conversational search by refining the original queries into de-contextualized forms to address issues such as omissions and coreferences. Previous CQR methods focus on imitating human-written queries, which may not always yield meaningful search results for the retriever. In this paper, we introduce *GuideCQR*, a framework that refines queries for CQR by leveraging key information from the initially retrieved documents. Specifically, *GuideCQR* extracts keywords and generates expected answers from the retrieved documents, then unifies them with the queries after filtering to add useful information that enhances the search process. Experimental results demonstrate that our proposed method achieves state-of-the-art performance across multiple datasets, outperforming previous CQR methods. Specifically, GuideCQR achieves MRR gains of 5.4% over LLM4CS on CAsT-19 and NDCG@3 gains of 29.2% on QReCC, and consistently improves retrieval across CAsT-19, CAsT-20, and QReCC benchmarks, demonstrating strong adaptability to various query types including human-rewritten queries. Additionally, we show that *GuideCQR* can get additional performance gains in conversational search using various types of queries, even for queries written by humans.

KEYWORDS: Conversational question answering; guided documents; information retrieval; large language model; query reformulation

1 Introduction

In a conversational question-answering task (ConvQA), conversational search aims to retrieve relevant passages that provide the necessary information to answer the current query. This process occurs within the framework of a multi-turn conversation, where each query builds upon the previous interactions. The questions in ConvQA often involve challenges like omissions and coreferences, which make it difficult to achieve the desired search results using the original query. Previous research focuses on transforming queries into stand-alone forms to better understand their intent, making them more independent and robust [1]. This process, known as Conversational Query Reformulation (CQR), helps clarify the original queries and enhances query understanding.

With the advent of Large Language Models (LLMs), their application in CQR methods has become increasingly widespread. A recent study [2] involves instructing an LLM to generate relevant passages related to the query. This method makes the query more reasonable and human-understandable by expanding

the sentence. LLM4CS [3] also utilizes LLMs to rewrite queries through various prompting methods and aggregation techniques. However, although these approaches aim to create human-friendly and easily comprehensible queries, they may not always yield the effective results that retrievers desire. Prior methods, such as LLM4CS, optimize for linguistic fluency and human readability, they do not explicitly align query representations with the retrieval objective, which can lead to suboptimal dense retrieval performance. For instance, as shown at the bottom of Fig. 1, although the reformulated query may seem less fluent due to the additional keywords, it achieves a higher retriever score than a base query because of its increased similarity to the ground-truth document. This example highlights the necessity of prioritizing retriever-optimized queries over human-friendly ones to enhance CQR methods. Specifically, the base query "Is throat cancer the same as esophageal cancer?" fails to rank the gold passage first (MRR: 72.2) because the gold passage contains the critical term *'interchangeable'*, which is absent from the base query. However, this term appears in both the 2nd- and 3rd-ranked retrieved documents, demonstrating that initially retrieved documents already contain the signals needed to reach the gold passage. To achieve this, we observe that the initially retrieved documents from a retrieval process with a base query set can help the generation of retrieval-optimized queries. Specifically, we find that although some of the documents cannot provide clues for answering the questions, most of the retrieved documents contain words or signals that are highly influential in subsequent retrieval processes. For instance, in the example illustrated in Fig. 1, the addition of the terms *'interchangeable'*, an important keyword found in the gold passage within the guided documents, enhances the performance that the original query alone cannot achieve. By appending this extracted keyword to the base query, GuideCQR raises MRR from 72.2 to 81.8, confirming that retrieval-oriented augmentation outperforms human-readable reformulation alone. In this way, signals in the initially retrieved documents obtained through a base query can guide the search for the gold passage.

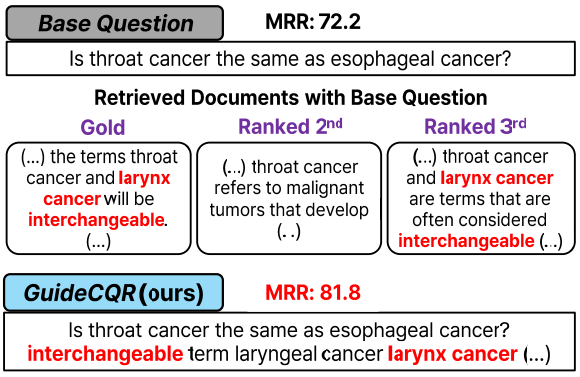

Figure 1: Example queries for ConvQA where the reformulated query achieves a higher MRR score by effectively extracting clues from the initially retrieved documents, compared to the base query.

In this paper, we propose *GuideCQR*, an effective CQR framework designed to generate retrieval-friendly conversational queries by leveraging the guidance of initially retrieved documents from the base query. *GuideCQR* first obtains the initially retrieved documents through a retrieval process using the baseline query reformulated by LLM. We then perform a re-ranking process to refine the order of the retrieved documents based on their similarity to the query. Through this re-ranking process, we carefully select a small number of documents from the initially retrieved ones to extract signals that are more likely to contain the gold passage. Next, we extract the keywords and generate the expected answer from the re-ranked guided documents to obtain components for creating a retriever-friendly query. We then independently filter both the keywords and the expected answers based on their similarity to the original query and previous

utterances to eliminate redundant information. Finally, we construct the final query set by concatenating the filtered keywords and expected answers with the baseline query.

Experimental results show that *GuideCQR* achieves state-of-the-art across several CQR datasets compared to the baseline systems. We also demonstrate the robustness of the *GuideCQR* framework in enhancing the retrievability of queries, making them more retriever-friendly. Furthermore, we validate the effectiveness of our method by analyzing the overlap between the augmented keywords and the relevant documents for the given question. Additionally, we show that *GuideCQR* is model-agnostic and compatible with any LLM-based rewriter, achieving consistent improvements across various query sets in the adaptability analysis.

2 Related Works

2.1 Conversational Search

The goal of conversational search [4] is to retrieve passages containing the necessary information to answer the current query within a multi-turn conversation. To get the desired answers from the given conversational dialogue, understanding the meaning of the query is important in this field [5]. However, ConvQA has several problems with its queries. For instance, queries may include pronouns such as "he," "she," or "it," necessitating the identification of the referred entity, and there is also the challenge of omission, where essential information is not included [3,6].

To address such challenges in conversational search, two main methods that enable conversational search are Conversational Dense Retrieval (CDR) and CQR. CDR focuses on enhancing the representation of the current query by incorporating historical context achieved through training dense retrievers [7]. On the other hand, CQR transforms the conversational search into a traditional ad-hoc search by converting the entire search session into a single standalone query [8].

In our work, we focus on utilizing the dialogue history to contextualize query rewrites. Additionally, we employ the multi-turn dialogue as a criterion for filtering keywords and answers, which are essential components of *GuideCQR*.

2.2 Conversational Query Reformulation

CQR methods aim to transform queries into de-contextualized forms, enabling them to be understood based solely on their content and ensuring they convey the intended meaning effectively. These methods are designed to enhance conversational search performance by refining and expanding user queries within a conversational context.

We categorize existing CQR approaches into three groups based on their primary learning signal, as this axis best captures the fundamental differences in how each line of work exploits supervision and retrieval feedback. The first group relies on human-annotated rewrites as direct supervision; the second reduces annotation cost through weak supervision or token-level selection; the third leverages LLM objectives for generative or reinforcement-based reformulation. Crucially, none of these groups explicitly leverages the retrieval signals present in documents surfaced at inference time, which is the gap our method addresses.

Supervised Rewriting Methods. Early CQR approaches rely on human-annotated rewrites as supervision signals. T5QR [9] fine-tunes a T5 model on human-generated rewrites to produce de-contextualized queries, while Transformer++ [10] fine-tunes GPT-2 on the CANARD dataset [8] for the same purpose. Reference [11] proposes a pipeline combining Conversational Term Selection (CTS) to de-contextualize user queries with Multi-View Reranking (MVR) to fuse multiple ranking views. Reference [12] focuses on query resolution under limited supervision by leveraging historical turns to clarify the current user query. Although

these methods produce fluent and human-readable rewrites, they are trained to imitate human annotation rather than to optimize retrieval signals, which limits their effectiveness in dense retrieval settings.

Weakly-Supervised and Token Selection Methods. To reduce reliance on expensive human annotations, a second line of work employs weak supervision or token-level selection strategies. CQE-Sparse [13] uses contextualized query embeddings to select the most informative tokens from the conversational context without requiring manually labeled rewrites. QuReTeC [12] trains a sequence tagger under weak supervision to determine which historical terms should be appended to the current query. Reference [5] further refines this direction by selectively incorporating only relevant historical queries for expansion, mitigating noise from unrelated turns. While these methods reduce annotation cost, they operate at the token level and lack the ability to generate novel, contextually rich query formulations that reflect the full user intent.

LLM-Based Query Expansion and Optimization. With the advent of large language models (LLMs), recent work has shifted toward generative and reinforcement-based reformulation. CONQRR [14] directly optimizes query rewriting for retrieval through Self-Critical Sequence Training [15], and ConvGQR [1] improves retrieval by combining query rewriting with generative query expansion. Reference [2] instructs an LLM to generate hypothetical relevant passages, and LLM4CS [3] further explores various prompting strategies and aggregation techniques to leverage LLM capabilities for CQR. More recently, CHIQ [16] adopts a two-step approach that uses LLMs to resolve ambiguities in conversation history before performing query rewriting, and CONVINV [17] transforms dense session embeddings into interpretable text via Vec2Text to produce rewritten queries. IterCQR [18] iteratively refines rewrites using retrieval signals as rewards, bringing the query closer to relevant documents across training iterations. Despite their strong performance, these methods optimize queries for human readability or general language model objectives, without explicitly leveraging signals from the documents that the retriever actually surfaces during inference.

To the best of our knowledge, no prior study has directly utilized the signals within initially retrieved documents to guide CQR at inference time. Our proposed method, *GuideCQR*, addresses this gap by extracting keywords and generating expected answers from the retrieved documents to construct retriever-friendly queries, complementing any existing rewriting method.

3 GuideCQR

We propose *GuideCQR*, a novel framework designed to reformulate conversational queries by utilizing guidance from initially retrieved documents. Fig. 2 provides an overview of the proposed query reformulation process, and Algorithm 1 summarizes the full procedure. *GuideCQR* consists of three stages: We first retrieve an initial set of guided documents using a query reformulated by LLM. This step retrieves documents that are likely to contain gold passages to guide the query reformulation process (Section 3.1). Next, we extract keywords and generate expected answers from the guided documents, creating components that contribute to making the query more retriever-friendly (Section 3.2). Finally, we apply a filtering process that evaluates both keywords and answers based on their similarity scores relative to the baseline query and dialogue history. We then unify and concatenate these components with the baseline query to construct the final query for post retrieval (Section 3.3).

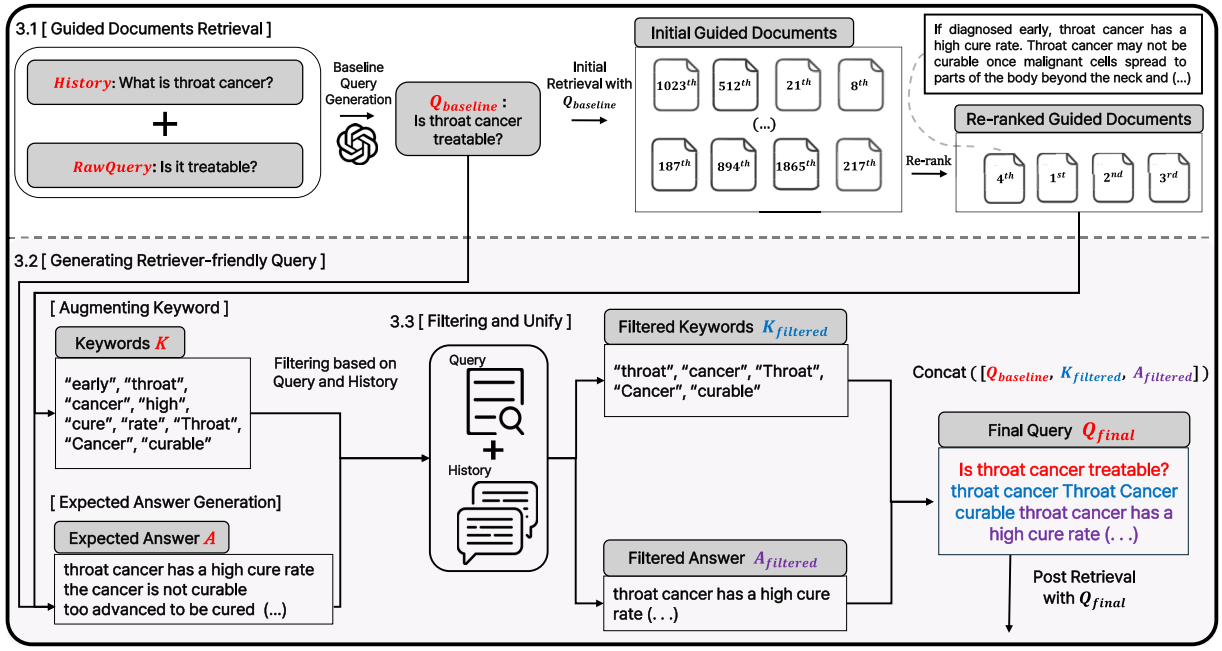

Figure 2: Overall framework of *GuideCQR*: For clarity, we visualize only the top-ranked document and present augmented keywords from this document, along with three answer pairs extracted from the top three ranked documents.

Algorithm 1: GuideCQR

Input: $RawQuery$; $History$; LLM rewriter $Rewrite(\cdot)$; retriever R; re-ranker S; hyperparams: n, m, τ
Output: Final reformulated query Q_{final}

; //**Stage 1: Guided Documents Retrieval**
$Q_{base} \leftarrow Rewrite(History,\ RawQuery)$;
$D \leftarrow R(Q_{base})$; //Retrieve top-N docs (N=2000)
$D \leftarrow S(D,\ Q_{base})$; //Re-rank
$D_{top} \leftarrow D[1:n]$; //Select top-n guided docs
; //**Stage 2: Generating Retriever-friendly Components**
$K \leftarrow \varnothing$; $A \leftarrow \varnothing$;
for $i = 1$ **to** n **do**
    $K \leftarrow K \cup \text{ExtractKeywords}(D_{top}[i],\ m)$;
    $A \leftarrow A \cup \{\text{GenerateAnswer}(Q_{base},\ D_{top}[i])\}$;
; //**Stage 3: Filtering and Unifying**
for $x \in (K \cup A)$ **do**

$$FS(x) \leftarrow \frac{10 \cdot \text{cosSim}(Q_{base},\ x) \ +\ 10 \cdot \max_{h \in History} \text{cosSim}(h,\ x)}{2};$$

$K_f \leftarrow \{k \in K \mid FS(k) \geq \tau\}$; $A_f \leftarrow \{a \in A \mid FS(a) \geq \tau\}$;
$Q_{final} \leftarrow \text{Concat}(Q_{base},\ K_f,\ A_f)$;
return Q_{final}

3.1 Guided Documents Retrieval

3.1.1 Initial Documents Retrieval

In the process of developing a retriever-friendly query, initially retrieved documents can play a crucial role as guiding resources by providing foundational insights for the CQR process. For example, in the contents of retrieved documents, these signals can include critical keywords or contextual signs that are necessary to search the gold passages, such as "cure," "rate," and "curable" from the document shown in Fig. 2.

Inspired by these points, *GuideCQR* begins by retrieving an initial set of documents to gain meaningful signals from the retriever. We obtain these initial documents by retrieving the documents using the baseline query set generated by LLM. We empirically determined $N = 2000$ as the initial retrieval size based on validation set performance and the observed effectiveness–efficiency trade-off. We denote baseline query as $Q_{baseline}$,

$$Q_{baseline} = Rewrite_{LLM}(History, RawQuery), \tag{1}$$

where $Rewrite_{LLM}$ represents an operation that resolves omissions or coreferences using OpenAI *gpt3.5-turbo-16k* and $RawQuery$ denotes the raw question in the dataset, without any reformulation applied. $History$ denotes the dialogue history of $RawQuery$, especially consisting of previous dialogues' queries, rewritten queries, and responses by human annotators. Based on both the $RawQuery$ and $History$, we generate the $Q_{baseline}$. Using this $Q_{baseline}$, we obtain the initial documents composed of 2000 documents for each question. They are crucial for creating our final query set and a strong foundation for identifying meaningful signals and guiding the retriever towards the most relevant passages.

3.1.2 Re-Ranking Documents

To further improve the quality of the guided set of documents, we implement a re-ranking process to generate the initially retrieved documents. By reordering the initial documents using a different retrieval model, we aim to capture better documents by reducing biases that may arise from relying on a single retriever. Specifically, we employ Sentence-Transformer [19] to re-rank the top 2000 documents for each question, selecting the final guided set by choosing the top 10 documents based on their similarity scores to the query. We employ Sentence-Transformer for re-ranking 2000 candidates per query due to its computational efficiency. Unlike cross-encoders, which require O(N) pairwise forward passes at inference time since document representations cannot be precomputed independently of the query, Sentence-Transformer encodes queries and documents separately, allowing document embeddings to be precomputed and enabling re-ranking via efficient cosine similarity computation.

3.2 Generating Retriever-Friendly Query

Since many retrieved documents may contain influential words or signals that significantly impact subsequent retrieval stages, identifying key elements from these documents is crucial for constructing more effective queries. Based on the guided documents obtained from the previous step, we generate retriever-friendly queries by incorporating extra information into the query from two approaches: *Augmenting Keywords*, *Expected Answer generation*.

3.2.1 Augmenting Keywords

We find that keywords from the initial documents play a critical role in forming retriever-friendly queries, as they capture the most relevant and significant terms within the document. For example, keywords such as "early", "throat", "cancer", "high", and "cure" can be beneficial if they are augmented to the search query as shown in Fig. 2.

We leverage KeyBERT [20] to extract keywords from the re-ranked documents. The process begins by using BERT [21] to compute embeddings of documents, which create a representation for the entire document. Embeddings for N-gram words or phrases within the document are then extracted. By calculating the cosine similarity between these words/phrases and the document representation, the method identifies which words/phrases are most similar to the document.

In line with this principle, we enhance the base queries by augmenting keywords through two hyper-parameters. The first one involves determining the number of documents to extract keywords from, which establishes the level of guidance we intend to provide. For each guided document, we extract keywords of a specified span length. The second aspect pertains to the span length, which indicates the number of tokens or keywords selected per document. For instance, when augmenting with top-2 span-3 keywords, we extract three keywords from each of the top two documents, yielding a total of six keywords.

Consequently, we augment a total keyword list K composed of nm keywords from the top-n documents and span length m:

$$K = [k_{11}, k_{12}, k_{13}, ..., k_{1m}, k_{21}, ..., k_{nm}], \tag{2}$$

where k denotes unit keyword, n is the number of documents and m is keyword span length.

3.2.2 Expected Answer Generation

Guided documents often include gold answers to the query, serving as a valuable resource for efficiently reformulating the query. Based on this idea, we use guided documents as context to generate expected answers to enhance the query.

As illustrated in Fig. 2, although it's not obtained from the gold passage, the expected answer is a concise and informative response that can potentially address the user's query since they can provide direct insights into the user's intent, thereby improving the relevance and accuracy of search results. Specifically, we generate these expected answers as follows:

$$A = [a_1, a_2, a_3, ..., a_n], \tag{3}$$

where A represents answer list and a_n denotes a unit answer extracted from a single document. To generate these answers, both the query and the relevant context are necessary. We use the query as the baseline query $Q_{baseline}$ and the context as the guided documents. We generate a single expected answer for each document. Consequently, this process produces one expected answer per guided document, yielding a set of answers that collectively reflect diverse retrieval signals across the guided documents. We generate expected answers using an extractive reading comprehension model with $Q_{baseline}$ as the question and each guided document as the context, ensuring answers are grounded in retrieved content (see Appendix A for implementation details).

3.3 Filtering and Unify

We observe that redundant elements, such as the keyword '*rate*' as shown in Fig. 2, can emerge from both the augmented keywords and the generated answers. We find that *GuideCQR* might augment keywords or answers derived from irrelevant documents. Hence, irrelevant signals from irrelevant documents can have a negative impact when creating retriever-friendly queries.

To address this, we introduce an additional filtering stage to more effectively remove irrelevant keywords and answers from the reformulated query. We guide this filtering process through the metric *FilterScore*, which leverages both $QueryScore$ and $HistoryScore$, calculated using cosine similarity, as follows:

$$cosSim(x, y) = \frac{x \cdot y}{\|x\|\|y\|}, \tag{4}$$

where *cosSim* denotes cosine similarity ranging from 0 to 1 and x, y denote embedding vectors of two items being compared. Based on $cosSim$, we firstly define $QueryScore$ as follows:

$$QueryScore = 10 \cdot cosSim(query, item), \tag{5}$$

where *query* represents the embedding of the current turn's query, and *item* refers to the embedding of either a keyword or an answer sentence. So $QueryScore$ is the cosine similarity between query and item, ranging from 0 to 10. And we define $HistoryScore$:

$$HistoryScore = 10 \cdot \max\left(cosSim(history[i], item)\right), \tag{6}$$

where *history* is the history query list of current utterances and i indexes each utterance in the dialogue history. So *HistoryScore* is the maximum cosine similarity value between the dialogue history query element in the history query list and the current item. Finally, we define the *FilterScore* as the average of the *QueryScore* and the *HistoryScore*, ranging from 1 to 10:

$$FilterScore = \frac{QueryScore + HistoryScore}{2}. \tag{7}$$

We define FilterScore as the arithmetic mean of QueryScore and HistoryScore, treating current query context and historical dialogue context as equally important signals for relevance assessment. This symmetric weighting reflects the multi-turn nature of conversational search, where both the immediate query and prior turns contribute to understanding user intent. History queries play a crucial role in understanding the current query, as in a dialogue, comprehending the question is more effective when based on the preceding conversation. Thus, we account for history queries through the $HistoryScore$, which allows us to traverse the entire dialogue and capture the global context from past to present. Using this $FilterScore$, we can eliminate signals that are irrelevant to the current dialogue. Since keywords and answers may vary in retriever-friendliness across different datasets, we treat them as distinct units and apply different filtering scores rather than using the same score for both.

Using the $FilterScore$, we filter out keywords and answers with a score below the specified threshold. Finally, we unify the remaining keywords and answers and integrate them into the $Q_{baseline}$ to construct the final reformulated query as follows:

$$Q_{final} = Concat([Q_{baseline}, K_{filtered}, A_{filtered}]), \tag{8}$$

where $K_{filtered}$ and $A_{filtered}$ are remaining keywords, answers. The thresholds applied to FilterScore differ across datasets because retrieval characteristics and document relevance distributions vary significantly across corpora. We select these values via grid search on each dataset's validation set. The symmetric averaging in FilterScore is motivated by the requirement that a useful augmentation signal must simultaneously satisfy two independent conditions: semantic alignment with the current query intent (QueryScore) and consistency with the broader conversational context (HistoryScore). A signal matching the current query but contradicting the history likely reflects a topic shift; one matching the history but not the current query adds anachronistic context. The symmetric average is the minimal unbiased aggregation under this two-condition requirement. The dataset-specific thresholds are a direct consequence of the varying retrieval characteristics across corpora. CAsT datasets provide graded 4-level relevance judgments with many relevant passages per query, producing a broad FilterScore distribution in which a lower threshold is needed to retain sufficient

signal. In contrast, QReCC pairs each query with a single gold passage, resulting in a narrow high-relevance band where a stricter threshold better separates useful signals from noise. Our sensitivity analysis shows that performance remains stable within ±0.5 of the optimal threshold, confirming that a coarse grid search on a small validation set is sufficient and that the method is not brittle to exact threshold values.

4 Experiments

4.1 Datasets and Metrics

We utilize three CQR benchmark datasets, TREC CAsT-19 [22], TREC CAsT-20 [23] and QReCC dataset [24] for our work. CAsT-19 and CAsT-20 are publicly available through the official TREC CAsT repository (https://github.com/daltonj/treccastweb), and QReCC is available at https://github.com/apple/ml-qrecc. Further details on data availability are provided in the "Availability of Data and Materials" section (see page 63). CAsT-19 and CAsT-20 consist of 50 and 25 conversations, respectively. Both CAsT datasets share the same document collection and provide passage-level relevance judgments, as well as human rewrites for each turn. Unlike CAsT-19, CAsT-20 is more realistic and complex because its queries are based on information needs drawn from commercial search logs, and they can reference prior system responses. CAsT series datasets assign a relevance score ranging from 1 to 4 to each passage, indicating the degree of relevance to the query, with documents scoring a 4 being considered gold passages. For the QReCC dataset, each query is paired with a single gold passage different from the CAsT dataset. QReCC dataset includes a training set and a test set, and we sample 2K conversations from the training set to create a development set, following the previous work [25].

Following the prior CQR research [1,3,18], we use three widely used evaluation metrics for CQR to compare the performance: Mean Reciprocal Rank (MRR), Normalized Discounted Cumulative Gain at three documents (NDCG@3), and Recall@10. MRR measures the rank of the first relevant document; NDCG@3 evaluates ranking quality within the top 3 results, accounting for graded relevance; Recall@10 measures the proportion of relevant documents retrieved within the top 10. We utilize *pytrec_eval* [26] tool to compute the score. We conduct the statistical significance tests using paired t-tests at $p < 0.05$ level.

4.2 Implementation Details

To generate a baseline query $Q_{baseline}$ for CAsT-19 and CAsT-20, we utilize a commercial LLM-based rewriter, combined with the Maxprob approach as proposed in LLM4CS. We simply generate this query by instructing LLM to rewrite the raw query based on dialog history and sampling the highest generation probability with LLMs, resolving only omissions and coreferences. For the QReCC dataset, rather than generating $Q_{baseline}$ ourselves, we adopt the final query output generated by InfoCQR [27] as $Q_{baseline}$. This approach leverages prompts from *gpt3.5-turbo* to enhance query generation.

Following previous studies [3,18], we use ANCE [28] pre-trained on the MSMARCO [29] as our retriever. For further implementation details, please refer to Appendix A. For the full prompt used to generate Q_{baseline}, see Table A1.

4.3 Baselines

We compare *GuideCQR* with the following various CQR baselines: (1) **Transformer++** [10]**:** A GPT-2 based CQR model fine-tuned on CANARD [8] dataset. (2) **CQE-Sparse** [13]**:** A weakly-supervised method to select important tokens only from the context via contextualized query embeddings. (3) **QuReTeC** [12]**:** A weakly-supervised method to train a sequence tagger to decide whether each term contained in a historical context should be added to the current query. (4) **T5QR** [9]**:** A conversational query rewriter based on

T5, trained using human-generated rewrites. (5) **ConvGQR** [1]**:** A query reformulation framework that combines query rewriting with generative query expansion. (6) **CONVINV** [17]**:** Framework that transforms conversational session embeddings into interpretable text using Vec2Text. (7) **CHIQ** [16]**:** A two-step method that leverages the capabilities of LLMs to resolve ambiguities in the conversation history before query rewriting. (8) **IterCQR** [18]**:** CQR framework through iterative refinement based on the similarity between the passage and the query. (9) **LLM4CS:** Query rewriting based on LLM and various prompting methods.

4.4 Performance Comparison

Main Results. As shown in Table 1, *GuideCQR* significantly enhances performance metrics compared to the $Q_{baseline}$ across all datasets. *GuideCQR* achieves state-of-the-art performance in terms of average score. In addition, *GuideCQR* achieves either the best or second-best performance compared to all baselines and demonstrates its robustness. Specifically, *GuideCQR* outperforms **LLM4CS** by 5.4% in MRR on the CAsT-19 dataset and by 29.2% in NDCG@3 on the QReCC dataset. For CAsT-20, its performance is almost the second-best. These results highlight the robustness and effectiveness of *GuideCQR*.

Table 1: Performance comparison for conversational search on various CQR methods on CAsT-19, CAsT-20, and QReCC dataset. We present MRR, NDCG@3, R@10, and the average of all scores for each method. The best results are in bold, and the second best are underlined. The dashes ('-') indicate performance values that could not be measured due to differences in evaluation metrics or the unavailability of results from closed-source systems. Baseline scores are reported as published in their original papers, except for $Q_{baseline}$ and LLM4CS, which we reproduce using publicly available implementations.

Methods	CAsT-19			CAsT-20			QReCC			Avg
	MRR	NDCG@3	R@10	MRR	NDCG@3	R@10	MRR	NDCG@3	R@10	
RawQuery	41.2	24.3	5.8	23.2	15.4	5.5	10.2	9.3	15.7	16.7
Transformer++	69.6	44.1	–	29.6	18.5	–	–	–	–	–
CQE-Sparse	67.1	42.3	–	42.3	27.1	–	32.0	30.1	51.3	–
QuReTeC	68.9	43.0	–	43.0	28.7	–	35.0	32.6	55.0	–
T5QR	70.1	41.7	–	42.3	29.9	–	34.5	31.8	53.1	–
ConvGQR	70.8	43.4	–	46.5	33.1	–	42.0	39.1	63.5	–
CONVINV	74.2	44.9	–	47.6	34.4	–	–	–	–	–
CHIQ	73.3	**50.5**	12.9	54.0	38.0	19.3	47.0	44.2	**70.8**	45.5
IterCQR	–	–	–	–	–	–	42.9	40.2	65.5	–
$Q_{baseline}$	72.2	45.2	12.2	53.4	37.2	18.2	43.8	51.9	66.4	44.5
LLM4CS	77.6	51.5	10.9	**61.5**	**45.5**	17.0	44.8	42.1	66.4	46.3
GuideCQR (ours)	**81.8**	**53.7**	**13.9**	59.0	42.7	**21.1**	**47.2**	**54.4**	67.0	**48.9**
Human Rewrite	74.0	46.1	12.9	59.1	42.2	21.0	43.1	51.6	58.6	45.4

To verify the statistical reliability of these improvements, we conducted paired t-tests comparing *GuideCQR* against Q_{baseline} across all 173 evaluated queries on CAsT-19. The gains are statistically significant on both MRR ($p = 0.003$) and NDCG@3 ($p = 0.006$), confirming that the observed improvements are not attributable to chance. We attribute the lower score for CAsT-20 compared to CAsT-19 to the increased complexity of the topics in CAsT-20. Specifically, in the retrieval process, we set the parameter *rel threshold*, which represents the minimum query relevance score for a document to be considered relevant to the query. CAsT-19 uses this threshold as 1, and CAsT-20 uses 2, so the minimum criteria for relevance is higher in CAsT-20. As a result, the number of relevant documents in CAsT-20 is significantly lower than in CAsT-19. Furthermore, the query relevance score file for CAsT-20 contains relatively few documents with high relevance scores; in most cases, the score is 0. This makes CAsT-20 a more challenging dataset for

applying *GuideCQR*. In other words, the signals derived from irrelevant guiding documents may not perform effectively to *GuideCQR* in CAsT-20.

Ablation Study. To evaluate the effectiveness of individual components involved in creating retriever-friendly queries in our proposed CQR framework, we conduct an ablation study. As demonstrated in Table 2, removing any part of *GuideCQR* leads to a decrease in performance, indicating that each component plays an important role in *GuideCQR*.

Table 2: Ablation study on each component of *GuideCQR*.

	CAsT-19		CAsT-20	
	MRR	**NDCG@3**	**MRR**	**NDCG@3**
GuideCQR (ours)	81.8	53.7	59.0	42.7
- w/o Keywords	72.0	47.3	54.7	38.6
- w/o Answers	81.0	52.2	57.9	41.6
- w/o Filtering	79.7	52.2	58.4	43.2

4.5 Analysis

The CAsT dataset includes multiple relevant passages, each with a relevance score. We focus on CAsT due to these unique label features.

Performance among Top-k Documents for Keyword and Answer. As shown in Table 3, we verify the impact of varying the number of documents used in augmenting keywords and generating expected answers. Our findings indicate that incorporating a larger number of documents generally provides additional information, which can improve performance. However, using too many documents may cause a decline in overall performance similar to the amount of initial guided documents.

Table 3: Performance on the CAsT-19 dataset with different numbers of documents for extracting keywords. We augment 15 spans of keywords from each document.

	Keyword			Answer		
	MRR	**NDCG@3**	**Avg**	**MRR**	**NDCG@3**	**Avg**
$Q_{baseline}$	72.2	45.2	58.7	72.2	45.2	58.7
top-2	78.3	50.1	64.2	76.7	48.3	62.5
top-3	80.4	52.1	66.2	77.5	49.0	63.2
top-4	81.0	52.2	66.6	76.9	50.0	63.4
top-5	80.2	52.3	66.2	77.5	49.6	63.5

Performance among the Number of Initial Guided Documents. We evaluate the impact and robustness of each step in the *GuideCQR* setup. Initially, we adjust the number of guided documents to observe the proper quantity and present the results in Table 4. Our findings demonstrate that increasing the number of guided documents consistently enhances performance. However, retrieving an excessive number of documents leads to longer inference times. Hence, considering the computation cost, we decide to use 2000 documents, which is the point on the dev set where there is no additional performance improvement in at least one metric. We also use the same criteria for keyword span length.

Table 4: Evaluation on the CAsT-19 dataset with varying amounts of guided documents.

# of Initial Guided Documents	MRR	NDCG@3	Avg
10	75.7	50.5	63.1
100	78.2	51.4	64.8
1000	79.5	52.1	65.8
2000	81.0	52.2	66.6

Query Relevance Analysis. To demonstrate the effectiveness of the keyword augmentation step, we evaluate the precision score using the following formula:

$$Precision = \frac{K_{rel}}{K_{total}}, \tag{9}$$

where K_{rel} represents the number of unique matched keywords found in the guided documents that have a query relevance score of rel, and K_{total} represents the total number of unique augmented keywords. Note that the document that has a relevance score of 1 or is regarded as the most relevant document with user queries in the CAsT datasets. Matching keywords in these documents can confirm that augmenting keywords plays a crucial role in generating retriever-friendly queries. As shown in Table 5, the precision score for relevant passages is higher than that for irrelevant passages, indicating that augmenting keywords plays a crucial role in retrieving relevant documents.

Table 5: Precision for irrelevant and relevant passages of CAsT-19 dataset.

		Precision
Irrelevant	Passage (0)	57.21
Relevant	Passage (1)	67.39
	Passage (2)	71.36
	Passage (3)	72.26
	Passage (4)	70.45

Failure Case Study. We analyze a representative failure case of our proposed method by sampling the final query in the CAsT-19 dataset. Following previous works [3], we first set the relevance threshold for CAsT-19 at 1, meaning that we consider retrieved documents with a score of 1 or higher relevant to the query. Therefore, in the sampling failure case process, if the top-ranked (first position) retrieved document has a relevance score of 0, we classify it as a failure case. As shown in Fig. 3, when retrieving documents using Q_{final}, the relevance score drops from 2 to 0 compared to $Q_{baseline}$, indicating a shift from relevant to irrelevant content. In $Q_{baseline}$, the query pertains to the main character of *The NeverEnding Story*, but the red-marked keywords and blue-marked answers in the retrieved documents refer to actual authors or actors, leading to a failure case. This observation suggests that performance tends to degrade when unrelated keywords and answers are introduced into the query, deviating from the original context of $Q_{baseline}$.

The root cause of this failure is that the initially retrieved documents, while topically adjacent to the query, are semantically misaligned with the user's true intent: they describe real-world biographical information rather than the fictional narrative content the user sought. As a result, GuideCQR extracts and appends misleading signals, causing the augmented query to drift away from the relevant passage. This

represents a broader risk inherent to retrieval-augmented reformulation: when the guided documents do not contain the gold passage, augmented signals may amplify retrieval errors rather than correct them. This limitation is directly tied to the quality of the first-stage retrieval, and is discussed further in Section 5.1.

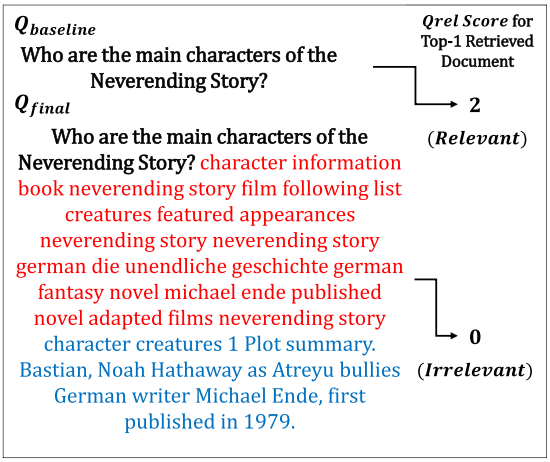

Figure 3: Failure case of *GuideCQR* for reformulating conversational query, where the system generates irrelevant keywords and answers about the $Q_{baseline}$.

To provide a more systematic characterization of potential failure modes, we analyzed the 110 queries in CAsT-19 (23.0% of the full 479-query evaluation set) where the baseline already retrieves a relevant document at rank 1, and GuideCQR introduces augmented signals, as summarized in Table 6. The dominant pattern is overlap overload (41.8%), where excessively repetitive augmented tokens add noise without contributing a new retrieval signal, suggesting that the current FilterScore threshold does not always suppress redundant content sufficiently. Answer noise (35.5%) covers cases where extractive spans are grammatically plausible but contextually off-topic, and keyword drift (22.7%) mirrors the failure pattern illustrated in Fig. 3, where augmented keywords pull the query toward a semantically misaligned subtopic. Importantly, these cases constitute a small minority of the total evaluation, and the overall MRR gain of +9.6 points over the baseline confirms that successful augmentations substantially outweigh failure cases.

Table 6: Failure mode taxonomy for GuideCQR on CAsT-19 (N = 110, 23.0% of 479 total queries).

Error Type	%	Description
Overlap overload	41.8	Repetitive augmented tokens add noise without new retrieval signal
Answer noise	35.5	Augmented answer span contextually off-topic despite grammatical plausibility
Keyword drift	22.7	Augmented keywords pull query toward topically adjacent but wrong subtopic

Adaptability of *GuideCQR*. *GuideCQR* is a highly effective framework as it operates as a query expansion method, making it adaptable for use with other methods or queries simultaneously. To demonstrate its effectiveness, we conduct experiments using human-rewritten queries, $RawQuery$, and $Q_{baseline}$ with different generator models from the CAsT-19 dataset. We create $Q_{baseline}$ using GPT-3.5-turbo-16k to

derive the final performance of GuideCQR. Furthermore, we conduct experiments with EXAONE-3.5-7.8B-Instruct [30], Mistral-7B-Instruct-v0.3 [31], and Qwen2.5-7B-Instruct [32]. As shown in Table 7, $RawQuery$, human-rewritten queries, and $Q_{baseline}$ with different generators show performance improvements. This demonstrates that our framework can be applied to any query or CQR method to achieve enhanced performance. We further clarify that the model-agnostic claim extends to the retrieval setup itself. As stated in Table A2 (Appendix B), GuideCQR consistently outperforms the BM25 baseline across all datasets, with gains of +5.8 MRR on RawQuery and +8.8 MRR on Human Rewritten queries on CAsT-19. The choice of ANCE as the primary retriever in the main experiments reflects its established status as the standard dense retriever for CAsT and QReCC benchmarks in prior CQR work, not a constraint of the framework itself.

Table 7: Performance comparison of applying *GuideCQR* on $RawQuery$, Human rewritten query and $Q_{baseline}$ with different generator models. Metrics include MRR, NDCG@3, and Recall@10, with performance improvements highlighted in red.

Methods	$Q_{baseline}$			GuideCQR		
	MRR	NDCG@3	R@10	MRR	NDCG@3	R@10
RawQuery	41.2	24.3	5.8	47.0 (+5.8)	28.8 (+4.5)	6.6 (+0.8)
Human Rewrite	74.0	46.1	12.9	82.8 (+8.8)	53.0 (+6.9)	14.4 (+1.5)
GPT3.5-turbo-16k	72.2	45.2	12.2	81.8 (+9.6)	53.7 (+8.5)	13.9 (+1.7)
Qwen2.5-7B-Instruct	62.9	39.3	10.2	65.0 (+2.1)	42.3 (+3.0)	11.7 (+1.5)
Mistral-7B-Instruct-v0.3	69.1	42.8	11.6	69.8 (+0.7)	46.0 (+3.2)	12.0 (+0.4)
EXAONE-3.5-7.8B-Instruct	66.4	41.5	11.1	72.6 (+6.2)	46.7 (+5.2)	12.5 (+1.4)

Performance among Keyword Span Length. We also investigate the impact of varying keyword spans in augmenting the keyword stage and present the result in Table 8. We observe that increasing the keyword span generally enhances performance to a certain extent. However, we find that too many signals can degrade performance due to the inclusion of redundant information in the query.

Sensitivity of FilterScore Threshold. Table 9 reports GuideCQR performance on CAsT-20 across varying FilterScore thresholds τ. Performance peaks at $\tau = 1.0$ and remains stable within ± 0.5 of the optimum, confirming that exact threshold values are not critical and that a coarse grid search on the validation set is sufficient.

Computational Cost Analysis. Table 10 reports the per-query inference time of each additional component in GuideCQR, measured on CAsT-19 queries using an NVIDIA RTX A6000 GPU (model loading time excluded). The initial and post-retrieval steps are performed with the same ANCE dense retriever used by the base CQR system and are not additional overhead unique to GuideCQR. Among the additional components, keyword extraction is the dominant cost (0.85 s), followed by re-ranking (0.23 s), answer generation (0.07 s), and filtering (0.05 s), totaling approximately 1.2 s of additional overhead per query. This overhead is well within the range acceptable for conversational search systems, which are typically deployed in interactive rather than latency-critical settings. For stricter latency requirements, reducing the keyword span range from $(1, 15)$ to $(1, 5)$ n-grams would substantially reduce the dominant cost at a marginal performance trade-off.

Table 8: Performance among different numbers of spans for extracting keywords. All keywords are augmented from the top-4 documents.

	CAsT-19		CAsT-20	
	MRR	**NDCG@3**	**MRR**	**NDCG@3**
$Q_{baseline}$	72.2	45.2	53.4	37.2
5 Span	76.9	49.9	57.7	41.5
10 Span	78.8	52.0	53.7	39.8
15 Span	81.0	52.2	54.1	40.0
20 Span	80.1	52.0	54.6	39.8

Table 9: Sensitivity of GuideCQR to FilterScore threshold τ on CAsT-20. Bold values indicate the best performance.

τ	**MRR**	**NDCG@3**	**Mean**
5.0	54.9	39.2	47.1
4.0	55.3	40.1	47.7
3.0	55.5	41.1	48.3
2.0	56.9	40.7	48.8
1.5	57.0	41.8	49.4
1.0	**58.2**	**42.6**	**50.4**
0.5	57.5	41.9	49.7

Table 10: Per-query inference time of GuideCQR's additional components (averaged over 20 CAsT-19 queries, RTX A6000, model loading time excluded). Initial and post-retrieval (ANCE) are shared with the base CQR system and are not additional overhead unique to GuideCQR.

Stage	**Mean (s)**	**Std (s)**
Initial retrieval (ANCE)	Shared with base	CQR
Re-ranking (Sentence-Transformer)	0.234	0.025
Keyword extraction (KeyBERT, top-4 docs)	0.853	0.036
Answer generation (RoBERTa-base, top-10 docs)	0.073	0.005
Filtering (cosine similarity, 14 items)	0.045	0.002
Post retrieval (ANCE)	Shared with base	CQR
Total additional overhead	**1.205**	

5 Discussion

5.1 Potential Limitations

GuideCQR introduces additional computational overhead compared to conventional CQR methods, as keyword augmentation and answer generation are performed over multiple retrieved documents at inference time. In addition, *GuideCQR* inherits a dependency on initial retrieval quality: when $Q_{baseline}$ fails to surface topically relevant documents in the guided pool, the extracted keywords and generated answers are drawn from irrelevant content, which may amplify retrieval errors. As shown in Table 4, performance degrades

when the guided pool is small ($N = 10$, MRR: 75.7 vs. 81.0 at $N = 2000$), confirming that a sufficiently large and topically relevant initial pool is necessary for reliable augmentation. The failure case analysis in Section 4.5 further illustrates how semantically misaligned guided documents cause topic drift in the augmented query. To mitigate this risk, *GuideCQR* employs the FilterScore mechanism to discard low-relevance signals before augmentation, and the entailment score (Appendix A) to detect potentially misleading answer spans. However, since retrieval difficulty and document relevance distributions differ substantially across corpora, the FilterScore threshold τ requires dataset-specific tuning via validation sets; an automatic selection mechanism such as Bayesian optimization would further improve generalizability to new domains, and we leave this as future work. Lastly, although we use an extractive QA model to minimize hallucination, entirely irrelevant guided documents can still produce misleading answer spans; incorporating an entailment check as a post-filtering step could further reduce this risk. While our aggregate results demonstrate that such failure cases are outweighed by successful augmentations, addressing these limitations remains a promising avenue for future work. Future work includes developing automatic threshold selection mechanisms for FilterScore, exploring cross-encoder re-ranking for improved document quality, and extending GuideCQR to open-domain settings with larger corpora. Another promising direction is integrating GuideCQR's augmentation mechanism into an end-to-end differentiable retrieval framework. The keyword extraction and filtering stages could be jointly optimized with the retriever through a contrastive learning objective, or framed as a reinforcement learning problem where the augmentation policy is rewarded based on downstream retrieval success. Such integration would allow the system to adapt the augmentation strategy to domain-specific retrieval characteristics without requiring dataset-specific threshold tuning.

5.2 Ethical Standards and Attribution

This research leverages multiple publicly available datasets for conversational query reformulation. We have accurately cited all the papers and sources used in our study. We intend to publish the final GuideCQR query for each dataset and the code, including the pre-trained model for our proposed framework, once the paper is accepted.

6 Conclusion

We present *GuideCQR*, a novel query reformulation framework utilizing initially retrieved documents from the query set for conversational search. *GuideCQR* effectively reformulates conversational queries to be more retriever-friendly by extracting meaningful information from these guided documents. Experimental results across various datasets and metrics confirm the capability of *GuideCQR* over previous CQR methods. We also show that GuideCQR can be effectively adapted to various types of rewritten queries.

Acknowledgement: Administrative and technical support from Chung-Ang University is gratefully acknowledged.

Funding Statement: This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) [RS-2021-II211341, Artificial Intelligence Graduate School Program (Chung-Ang University)] and Korea Institute for Advancement of Technology (KIAT) grant funded by the Korea Government (MOTIE) (RS-2025-25458133). This research was supported by the Chung-Ang University Graduate Research Scholarship in 2025.

Author Contributions: Jeonghyun Park: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing—original draft preparation, Visualization. Hwanhee Lee: Conceptualization, Validation, Writing—review and editing, Visualization, Supervision, Project administration, Funding acquisition. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The reformulated queries generated by GuideCQR for each dataset, as well as the source code and pre-trained models for the proposed framework, will be made publicly available upon acceptance of the manuscript. The benchmark datasets used in this study (CAsT-19, CAsT-20, and QReCC) are publicly accessible through their respective original sources cited in this paper: CAsT-19 and CAsT-20 at https://github.com/daltonj/treccastweb; QReCC at https://github.com/apple/ml-qrecc.

Ethics Approval: Not applicable. This study does not involve human subjects, animal experiments, or any sensitive personal data.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A Implementation Details

Models. To re-rank the guided documents as in Section 3.1.2, we select the *ember-v1*[1] and *mxbai-embed-large-v1*[2] models as the first and second re-ranking Sentence-Transformer models, respectively. We use OpenAI *text-embedding-3-small* for generating embeddings for keywords and answers in the filtering stage in *GuideCQR*. We use different embedding models at each stage to mitigate potential biases caused by a single model. We use KeyBERT [20] for augmenting keywords. For expected answer generation, we use *RoBERTa-Base-Squad2*[3], an extractive reading comprehension model, with $Q_{baseline}$ as the question and each guided document as the context. Since this is an extractive approach, no prompt design is required; the model directly identifies the most relevant span within the document, ensuring answers remain grounded in retrieved content and minimizing hallucination risk. We adopt *GPT-3.5-turbo-16k* for baseline query generation to ensure a fair comparison with LLM4CS [3], which uses the same model. We use the most common query among the five candidate queries provided by LLM4CS [3] during re-ranking, answer generation, and filtering in CAsT datasets. In the keyword augmentation process, the number of extracted keywords can be fewer than the span length, as some documents may contain fewer meaningful terms than the specified span; the keyword span length thus serves as an upper bound rather than a fixed count. We use *roberta-large-mnli*[4] for measuring the entailment score between augmented keywords and $Q_{baseline}$. In the query relevance analysis, we ensure a fair comparison by standardizing the number of documents in each set to 1700 before conducting the experiment. We leverage Faiss index [33] for the ANCE retriever.

Selecting FilterScore. We experiment with various $FilterScore$ settings to find the optimal hyper-parameter with the validation set for the proposed method. The number of documents for keywords and expected answers and optimal keyword span length varies by dataset; these values were chosen to improve retrieval results and enhance query quality for the retriever. Therefore, before filtering, we use top-4 span 15 keywords and top-10 answers for CAsT-19, and top-5 span 5 keywords and top-10 answers for CAsT-20, and top-1 span 10 keywords and top-10 answers for QReCC. And *GuideCQR* uses these keywords and answers as our final query set. We select the $FilterScore$ for each keyword and answer by experimenting with validation sets for the datasets to find the optimal scores. We achieve our final results using a $FilterScore$ of (1, 1.9) for each keyword and answer in CAsT-19, (0.1, 1.95) in CAsT-20, and (0.5, 9) in QReCC.

Process of Generating $Q_{baseline}$ of CAsT. We generate $Q_{baseline}$ of CAsT datasets by utilizing the LLM4CS REW GitHub code, which employs GPT-3.5-turbo-16k for prompting [3]. As illustrated in Table A1, the prompt consists of three parts: instruction, demonstration, and tail instruction. The final prompt is a

[1] https://huggingface.co/llmrails/ember-v1
[2] https://huggingface.co/mixedbread-ai/mxbai-embed-large-v1
[3] https://huggingface.co/deepset/roberta-base-squad2
[4] https://huggingface.co/FacebookAI/roberta-large-mnli

concatenation of these three components. Using this prompt, *GuideCQR* generates $Q_{baseline}$ by refining $RawQuery$ with $History$, resolving any coreferences or omissions.

Computing Infrastructure. We conduct our experiments using an AMD EPYC 7313 CPU (3.0 GHz) paired with four NVIDIA RTX 4090 GPUs. We use Python 3.11.5 and PyTorch 2.3.1 for the software environment.

Table A1: The prompt for generating $Q_{baseline}$ is composed of three components: instruction, demonstration, and tail instruction.

Instruction	For an information-seeking dialog, please help reformulate the question into a rewrite that can fully express the user's information needs without the need for context.
Demonstration	I will give you several example multi-turn dialogs, where each turn contains a question, a response, and a rewrite that you need to generate.
Tail Instruction	Now, you should give me the rewrite of the **Current Question** under the **Context**. The output format should always be: Rewrite: $Rewrite. Note that you should always try to rewrite it. Never ask for clarification or say you don't understand it in the generated rewrite. Go ahead!

Appendix B Result Using Sparse Retrieval

We provide the performance of *GuideCQR* using sparse retrieval. This result is not included in the main table because prior studies on the CAsT series have predominantly focused on dense retrieval. Consequently, retrieval results for baseline methods are unavailable, and the rewritten queries are not provided either. Thus, we conduct experiments with the available resources. We perform retrieval using BM25 [34], and indexing is performed via Pyserini [35]. For sparse retrieval in LLM4CS [3], we concatenate the rewritten query and response of the generated LLM4CS final query.

As shown in Table A2, *GuideCQR* significantly enhances performance metrics compared to the $Q_{baseline}$ across all datasets. *GuideCQR* achieves state-of-the-art performance in terms of the average score. Similar to earlier observations, the scores for CAsT-20 are lower due to the unique characteristics of the CAsT datasets.

Table A2: Performance comparison on CAsT-19, CAsT-20, and QReCC datasets using sparse retriever (BM25). We present MRR, NDCG@3, R@10, and the average of all scores for each method. The best results are in bold, and the second best are underlined.

Methods	CAsT-19			CAsT-20			QReCC			Avg
	MRR	NDCG@3	R@10	MRR	NDCG@3	R@10	MRR	NDCG@3	R@10	
RawQuery	31.9	13.1	3.8	9.9	6.8	2.5	6.5	5.5	11.1	10.1
$Q_{baseline}$	59.9	29.5	10.3	30.5	19.0	10.7	46.9	55.0	65.5	36.4
LLM4CS	<u>67.5</u>	<u>41.9</u>	<u>11.5</u>	**53.7**	**38.8**	**20.5**	<u>47.8</u>	<u>45.0</u>	<u>69.1</u>	<u>44.0</u>
GuideCQR (ours)	**75.4**	**49.7**	**14.0**	<u>49.1</u>	<u>34.4</u>	<u>16.4</u>	**51.2**	**58.6**	**69.0**	**46.4**
Human Rewrite	61.3	30.9	10.9	37.7	23.9	14.2	39.7	36.2	62.5	35.3

Appendix C Dataset Statistics

CAsT. Table A3 presents statistics for the CAsT datasets, including the number of conversations, queries, and documents.

Table A3: Dataset statistics for CAsT-19, CAsT-20, QReCC, QuAC-Conv, NQ-Conv, and TREC-Conv.

Dataset	Split	# Conv	# Questions	# Documents
CAsT-19	Test	50	479	38M
CAsT-20	Test	25	216	38M
QReCC	Train	8823	51,928	–
	Dev	2000	11,573	–
	Test	2775	16,451	–
QuAC-Conv	Train	6008	41,395	–
	Dev	1300	8965	–
	Test	1816	12,389	–
NQ-Conv	Train	2815	10,533	–
	Dev	700	2608	–
	Test	879	3314	–
TREC-Conv	Train	0	0	–
	Dev	0	0	–
	Test	80	748	–

QReCC. Table A3 shows the statistics for the QReCC dataset, which contains approximately 54M documents. We use a final test set of 8209 questions, following [27], after removing invalid gold passage labels. This test set includes 6396 questions for QuAC-Conv, 1442 for NQ-Conv, and 371 for TREC-Conv.

Appendix D Examples Queries on Datasets

We provide a query sample of each dataset. For CAsT, if the top-ranked (first position) retrieved document has a relevance score of 1 or higher, we classify it as a positive case. For the QReCC dataset, we refer to a retrieval result as a positive case if the gold passage is included in the top-1 result. Tables A4–A6 show positive query sample of CAsT-19, CAsT-20, and QReCC dataset.

Table A4: Reformulated Queries by GuideCQR on CAsT-19 Dataset. Overlapping words between **GuideCQR Query** and **Gold Passage (4)** are marked in red. **Gold Passage (4)** notates that the document has a query relevance score of 4, and **LLM** notates the $Q_{baseline}$.

CAsT-19 Dataset	
Previous Turns	Query: What is throat cancer? Response: “” Query: Is it treatable? Response: “” Query: Tell me about lung cancer. Response: “”
Original Query	What are its symptoms?

(Continued)

Table A4 (continued)

CAsT-19 Dataset	
Human Rewrite	What are lung cancer's symptoms?
LLM	What are the symptoms of lung cancer?
GuideCQR Query	What are the symptoms of lung cancer? signs symptoms lung cancer symptoms lung cancer signs symptoms lung cancer cough signs symptoms lung cancer cough (hemoptysis), breath, wheezing, hoarseness coughing; pain chest, coughing
Gold Passage (4)	Signs and symptoms of lung cancer typically occur only when the disease is advanced. Signs and symptoms of lung cancer may include: A new cough that doesn't go away. Changes in a chronic cough or smoker's cough. Coughing up blood, even a small amount. Shortness of breath. Chest pain.

Table A5: Reformulated Queries by GuideCQR on CAsT-20 Dataset. Overlapping words between **GuideCQR Query** and **Gold Passage (4)** are marked in red. **Gold Passage (4)** notates that the document has a query relevance score of 4, and **LLM** notates the $Q_{baseline}$.

CAsT-20 Dataset	
Previous Turns	Query: How do you know when your garage door opener is going bad? Response: Light Socket or Logic Board. If the light is still not coming on, it could be the light socket or the logic board. To determine which one it is, follow the steps below. Unplug the garage door opener, then immediately plug the opener back into the electrical outlet. Listen for the light relay on the logic board to click. If you hear a click, replace the light socket. If you do not hear a click, replace the logic board. There are four causes for the lights on a garage door opener not to come on: 1. Light bulb. 2. Contacts in the light socket. 3. Light socket. Query: Now it has stopped working. Why? Response: Garage Door Opener Problems. So, when the garage door opener decides to take a day off, it can leave you stuck outside, probably during a rain or snowstorm. Though they may seem complicated, there really are several things most homeowners can do to diagnose and repair opener failures. And, if you are careful not to damage the door or the seal on the bottom of the door, use a flat shovel or similar tool to chip away at the ice. Once you get the door open, clear any water, ice or snow from the spot on the garage floor where the door rests when closed.
Original Query	How much does it cost for someone to fix it?
Human Rewrite	How much does it cost for someone to repair a garage door opener?

(Continued)

Table A5 (continued)

CAsT-20 Dataset	
LLM	How much does it typically cost to have a professional fix a garage door opener issue?
GuideCQR Query	How much does it typically cost to have a professional fix a garage door opener issue? garage door opener repair costgarage door opener repair cost cost repairbrokengarage door openergarage door opener repair costgarage door opener repair cost fix garage door opener
Gold Passage (4)	On average, homeowners report paying $210 to have a garage door opener repaired by a handyman. Most homeowners pay between $170 and $250 for the cost of such a repair project. The minimum reported cost to repair a brokengarage door opener is $50, while the most costly repair may amount to $350. If an electrician is needed to repair faulty wiring, the cost of the repair will increase. The minimum reported cost to repair a brokengarage door opener is $50, while the most costly repair may amount to $350. If an electrician is needed to repair faulty wiring, the cost of the repair will increase.

Table A6: Reformulated Queries by GuideCQR on QReCC Dataset. Overlapping words between **GuideCQR Query** and **Gold Passage**, **Gold Answer** are marked in red. **LLM** notates the $Q_{baseline}$.

QReCC Dataset	
Previous Turns	Query: What are the main breeds of goat?, Answer: Abaza...Zhongwei
Original Query	Tell me about boer goats.
Human Rewrite	Tell me about boer goats.
LLM	What can you tell me about boer goats in relation to the main breeds of goats?
GuideCQR Query	What can you tell me about boer goats in relation to the main breeds of goats? goat boer breed south developed
Gold Answer	The Boer goat is a breed of goat that was developed in South Africa in the early 1900s for meat production. Their name is derived from the Afrikaans (Dutch) word boer, meaning farmer.

(Continued)

Table A6 (continued)

QReCC Dataset	
Gold Passage	Boer goat—Wikipedia CentralNotice Boer goat From Wikipedia, the free encyclopedia Jump to navigation Jump to search A commercial Boer goat buck Commercial Boer goat buck Boer goat, Doe The Boer goat is a breed of goat that was developed in South Africa in the early 1900s and is a popular breed for meat production. Their name is derived from the Afrikaans (Dutch) word boer, meaning farmer. Contents 1 Origins and characteristics 2 Doe 3 Crossbreeding 4 Show goats 5 References 6 External links Origins and characteristics [edit] The Boer goat was probably bred from the indigenous South African goats kept by the Namaqua, San, and Fooku tribes, with some crossing of Indian and European bloodlines being possible. They were selected for meat rather than milk production; due to selective breeding and improvement, the Boer goat has a fast growth rate and excellent carcass qualities, making it one of the most popular breeds of meat goat in the world. Boer goats have a high resistance to disease and adapt well to hot, dry semideserts. United States production is centered in west-central Texas, particularly in and around San Angelo and Menard. The original US breeding stock came from herds located in New Zealand. Only later were they imported directly from Africa Boer goats commonly have white bodies and distinctive brown heads. Some Boer goats can be completely brown or white or paint, which means large spots of a different color are on their bodies. Like the Nubian goat, they possess long, pendulous ears. They are noted for being docile, fast-growing, and having high fertility rates. Does are reported to have superior mothering skills as compared to other breeds.

References

1. Mo F, Mao K, Zhu Y, Wu Y, Huang K, Nie J-Y. ConvGQR: generative query reformulation for conversational search. In: Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics. Kerrville, TX, USA: ACL; 2023. p. 4998–5012.
2. Jagerman R, Zhuang H, Qin Z, Wang X, Bendersky M. Query expansion by prompting large language models. arXiv:2305.03653. 2023.
3. Mao K, Dou Z, Mo F, Hou J, Chen H, Qian H. Large language models know your contextual search intent: a prompting framework for conversational search. In: Findings of ACL: EMNLP 2023. Kerrville, TX, USA: ACL; 2023. p. 1211–25.
4. Gao J, Xiong C, Bennett P, Craswell N. Neural approaches to conversational information retrieval. Berlin/Heidelberg, Germany: Springer; 2022.
5. Mo F, Nie J-Y, Huang K, Mao K, Zhu Y, Li P, et al. Learning to relate to previous turns in conversational search. In: Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23). New York, NY, USA: ACM; 2023. p. 1722–32. doi:10.1145/3580305.3599411.
6. Wang L, Yang N, Wei F. Query2doc: query expansion with large language models. In: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. Kerrville, TX, USA: ACL; 2023. p. 9414–23.

7. Qu C, Yang L, Chen C, Qiu M, Croft WB, Iyyer M. Open-retrieval conversational question answering. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. New York, NY, USA: ACM; 2020. p. 539–48. doi:10.1145/3397271.3401110.
8. Elgohary A, Peskov D, Boyd-Graber J. Can you unpack that? Learning to rewrite questions-in-context. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). Kerrville, TX, USA: ACL; 2019. p. 5918–24. doi:10.18653/v1/D19-1605.
9. 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:2004.01909. 2020.
10. Vakulenko S, Longpre S, Tu Z, Anantha R. Question rewriting for conversational question answering. In: Proceedings of the 14th ACM International Conference on Web Search and Data Mining. New York, NY, USA: ACM; 2021. p. 355–63.
11. Kumar V, Callan J. Making information seeking easier: an improved pipeline for conversational search. In: Findings of the Association for Computational Linguistics: EMNLP 2020. Kerrville, TX, USA: ACL; 2020. p. 3971–80. doi:10.18653/v1/2020.findings-emnlp.354.
12. Voskarides N, Li D, Ren P, Kanoulas E, de Rijke M. Query resolution for conversational search with limited supervision. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20). New York, NY, USA: ACM; 2020. p. 921–30. doi:10.1145/3397271.3401130.
13. Lin S-C, Yang J-H, Lin J. Contextualized query embeddings for conversational search. In: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing. Kerrville, TX, USA: ACL; 2021. p. 1004–15.
14. Wu Z, Luan Y, Rashkin H, Reitter D, Hajishirzi H, Ostendorf M, et al. CONQRR: conversational query rewriting for retrieval with reinforcement learning. In: Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing. Kerrville, TX, USA: ACL; 2022. p. 10000–14. doi:10.18653/v1/2022.emnlp-main.679.
15. Rennie S J, Marcheret E, Mroueh Y, Ross J, Goel V. Self-critical sequence training for image captioning. In: Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway, NJ, USA: IEEE; 2017. p. 1179–95. doi:10.1109/CVPR.2017.131.
16. Mo F, Ghaddar A, Mao K, Rezagholizadeh M, Chen B, Liu Q, et al. CHIQ: contextual history enhancement for improving query rewriting in conversational search. arXiv:2406.05013. 2024.
17. Cheng Y, Mao K, Dou Z. Interpreting conversational dense retrieval by rewriting-enhanced inversion of session embedding. In: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics. Kerrville, TX, USA: ACL; 2024. p. 2879–93.
18. Jang Y, Lee K-I, Bae H, Lee H, Jung K. IterCQR: iterative conversational query reformulation with retrieval guidance. In: Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Kerrville, TX, USA: ACL; 2024. p. 8121–38. doi:10.18653/v1/2024.naacl-long.449.
19. Reimers N, Gurevych I. Sentence-BERT: sentence embeddings using siamese BERT-networks. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). Kerrville, TX, USA: ACL; 2019. p. 3982–92.
20. Grootendorst M. KeyBERT: minimal keyword extraction with BERT. Zenodo, v0.3.0; 2020. doi:10.5281/zenodo.4461265.
21. Devlin J, Chang M-W, Lee K, Toutanova K. BERT: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Kerrville, TX, USA: ACL; 2019. p. 4171–86. doi:10.18653/v1/N19-1423.
22. Dalton J, Xiong C, Kumar V, Callan J. CAsT-19: a dataset for conversational information seeking. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. New York, NY, USA: ACM; 2020. p. 1985–8. doi:10.1145/3397271.3401206.
23. Dalton J, Xiong C, Callan J. CAsT 2020: the conversational assistance track overview. In: Proceedings of the Text Retrieval Conference (TREC) 2020. Gaithersburg, MD, USA: NIST; 2020.

24. Anantha R, Vakulenko S, Tu Z, Longpre S, Pulman S, Chappidi S. Open-domain question answering goes conversational via question rewriting. In: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Kerrville, TX, USA: ACL; 2021. p. 520–34. doi:10.18653/v1/2021.naacl-main.44.
25. Kim S, Kim G. Saving dense retriever from shortcut dependency in conversational search. In: Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing. Kerrville, TX, USA: ACL; 2022. p. 10278–87.
26. Gysel C V, de Rijke M. Pytrec_eval: an extremely fast Python interface to trec_eval. In: Proceedings of the 41st International ACM SIGIR Conference on Research and Development in Information Retrieval. New York, NY, USA: ACM; 2018. p. 873–6.
27. Ye F, Fang M, Li S, Yilmaz E. Enhancing conversational search: large language model-aided informative query rewriting. In: Findings of the Association for Computational Linguistics: EMNLP 2023. Kerrville, TX, USA: ACL; 2023. p. 5985–6006.
28. Xiong L, Xiong C, Li Y, Tang K-F, Liu J, Bennett PN, et al. Approximate nearest neighbor negative contrastive learning for dense text retrieval. arXiv:2007.00808. 2020.
29. Campos DF, Nguyen T, Rosenberg M, Song X, Gao J, Tiwary S, et al. MS MARCO: a human generated machine reading comprehension dataset. arXiv:1611.09268. 2016.
30. LG AI Research. EXAONE 3.5: series of large language models for real-world use cases. arXiv:2412.04862. 2024.
31. Jiang AQ, Sablayrolles A, Mensch A, Bamford C, Chaplot DS, de las Casas D, et al. Mistral 7B. arXiv:2310.06825. 2023.
32. Qwen Team. Qwen2.5: a party of foundation models. 2024 [cited 2026 Apr 20]. Available from: https://qwenlm.github.io/blog/qwen2.5/.
33. Johnson J, Douze M, Jégou H. Billion-scale similarity search with GPUs. IEEE Trans Big Data. 2019;7(3):535–47. doi:10.1109/TBDATA.2019.2921572.
34. Robertson S, Zaragoza H. The probabilistic relevance framework: BM25 and beyond. Found Trends® Inf Retrieval. 2009;3(4):333–89. doi:10.1561/1500000019.
35. Lin J, Ma X, Lin S-C, Yang J-H, Pradeep R, Nogueira R F. Pyserini: a Python toolkit for reproducible information retrieval research with sparse and dense representations. In: SIGIR '21: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval. New York, NY, USA: ACM; 2021. p. 2356–62. doi:10.1145/3404835.3463238.