

RCEM: Robust Conversational Search EMbedder in Distributional Shift

Kilho Son^{1,*}, Bo-June Paul Hsu¹, Cha Zhang¹, Dinei Florencio²
¹{kilhoson, paulhsu, chazhang}@microsoft.com, ²dinei@dflorencio.com

¹Microsoft Corporation

Abstract

We propose RCEM, a Robust Conversational search EMbedder that augments text embedders with LLM query reformulation capabilities while approximately preserving their original generalization performance. Unlike prior conversational dense retrieval approaches that learn direct conversation-to-passage matching, RCEM aligns conversations, prepended by special token, to LLM-rewritten queries, while attempting to preserve the original embedding space. The approximately preserved embedding space automatically maps the rewritten queries to the relevant passages. As a result, RCEM (1) reduces overfitting by aligning conversations to shorter rewritten queries rather than full passages, (2) eliminates the need for conversation-to-passage relevance labels for training, and (3) approximately preserves the original embedding space, enabling conversational retrieval over indexes built by the base embedder without re-indexing. Extensive experiments show that RCEM consistently outperforms prior approaches, improving up to 30% retrieval accuracy under distributional shift.

1 Introduction

Conversational search is a core component of retrieval-augmented generation (RAG) systems, where users interact with AI assistants through multi-turn dialogue rather than single queries. In this setting, queries are often context-dependent and omit previously stated information, making user inputs vague, ambiguous, or incomplete (Gao et al., 2023). Resolving such contextual dependencies is, therefore, essential for retrieving relevant passages and generating accurate responses.

A common approach is to first rewrite conversational queries into standalone queries using a large language model (LLM), which can then be embedded and used for dense retrieval. Recent advances

* Corresponding author.

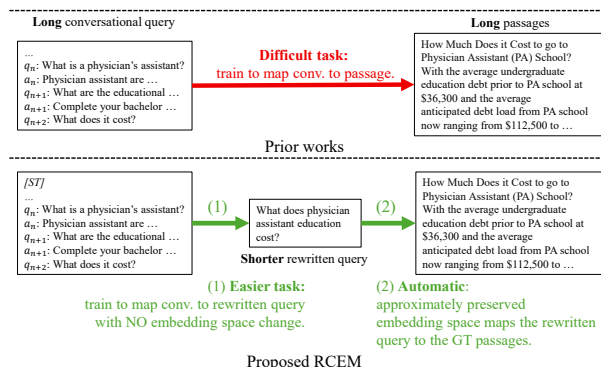


Figure 1: **Overview.** RCEM is trained to map conversational queries (prepended by a special token, *[ST]*) to LLM-rewritten queries and preserve its original embedding space, which naturally supports retrieval of relevant passages. During inference, *[ST]* triggers RCEM to implicitly convert a conversational query into a clear LLM-rewritten query and computes its embedding in one-step inference, without running an LLM model. Unlike prior methods that directly align long conversations with long passages, the simplified objective reduces overfitting. The approximately preserved embedding space makes RCEM compatible with existing indexes and only activates conversational search when the special token is presented. Otherwise, RCEM behaves similarly to the original embedder. Moreover, RCEM removes the need for expensive conversational query-to-passage relevance annotations for training, enabling greater scalability.

in query reformulation have significantly improved retrieval accuracy in multi-turn settings (Mao et al., 2023b; Mo et al., 2023, 2024a; Lai et al., 2025a). These LLM-based pipelines, however, incur expensive generation cost at inference time, leading to high latency and computational cost, which limits scalability in production. Conversational dense retrieval methods instead aim to generate embeddings directly from conversations, avoiding LLM calls at inference (Lin et al., 2021; Yang et al., 2025b). These approaches finetune embedders to align conversational queries with relevant passages, requiring high-quality relevance labels. Yet, map-

ping long conversational queries to long passages is an inherently challenging task, making the models prone to overfitting.

We propose RCEM, a **Robust Conversational search EMBedder** that *additionally* learns LLM-based query reformulation capability for conversational search, while preserving the original embedding space and generalization performance. Unlike prior conversational dense retrieval methods (Lin et al., 2021; Yang et al., 2025b), which directly map conversational queries to ground-truth passages, RCEM aligns conversational queries (prefixed by a special token) with LLM-rewritten queries internally and generates their embeddings, while preserving the original embedding space. Because the embedding space is approximately preserved, the resulting embeddings naturally retrieve relevant passages. By aligning conversational queries to shorter LLM-rewritten queries rather than long passages, RCEM simplifies the learning task, reducing overfitting. Maintaining the original embedding space enables conversational retrieval on existing indexes built by original embedder, removing the need for index rebuilding. And conversational search is only triggered for inputs with the special token. Otherwise, the model behaves similarly to the original embedder which increases compatibility with existing search system. Furthermore, RCEM eliminates the expensive cost for high-quality conversational query-to-passage relevance annotations for training, improving scalability (See Figure 1). Extensive experiments show that RCEM consistently outperforms prior work, with particularly large gains under distributional shift.

2 Related Works

Existing conversational search approaches generally fall into two categories: conversational query reformulation (CQR) and conversational dense retrieval (CDR). CQR rewrites context-dependent conversational queries into standalone queries prior to sparse or dense retrieval, while CDR directly encodes conversational context into dense embeddings for retrieval.

Conversational Query Reformulation (CQR). Recent CQR research has focused on improving query reformulation quality through query expansion, intent preservation, ambiguity resolution, and retrieval-aware optimization. Mo et al. (2023) integrate generative query rewriting with potential

answer generation for query expansion, while Mao et al. (2023a) introduce search-oriented conversational query editing to better preserve retrieval intent during reformulation. Mao et al. (2023b) further improve retrieval effectiveness by aggregating multiple query rewrites and their hypothetical responses into a single retrieval representation. Subsequent work enhances reformulation through more sophisticated generation pipelines, such as rewrite-then-edit strategies (Ye et al., 2023), ambiguity reduction in conversations prior to rewriting (Mo et al., 2024a), retrieval-aware training objectives that align reformulation with both sparse and dense retrieval (Lai et al., 2025b), and test-time adaptation methods that generate multiple candidate rewrites and select the most effective one for retrieval (Lai et al., 2025a). CQR approaches are modular and retriever-agnostic, allowing seamless integration with existing sparse and dense retrieval systems. They also naturally benefit from advances in LLMs and prompting techniques. However, they incur substantial cost and latency at inference due to repeated LLM calls, making large-scale deployment challenging.

Conversational Dense Retrieval (CDR). CDR methods directly encode conversational context into dense retrieval representations. Lin et al. (2021); Yang et al. (2025b) introduce contextualized query embeddings that incorporate conversation history for conversational retrieval by training to map conversational queries to the relevant passages. Subsequent research improves retrieval effectiveness through context-denoised representations and pseudo-supervision mined from historical turns (Mo et al., 2024b), generalized conversational retrieval models based on LLMs and dual-learning objectives (Mao et al., 2024), and methods that enhance interpretability by decoding dense conversational embeddings into human-readable queries while preserving retrieval performance (Cheng et al., 2024). These approaches benefit from end-to-end optimization and deep integration of conversational context into retrieval representations. However, existing CDR methods typically rely on contrastive training between conversational queries and relevant passages using high-quality conversation-to-passage relevance annotations. Acquiring such supervision is expensive and challenging, limiting scalability. Furthermore, training on relatively small conversational retrieval datasets can lead to overfitting, while contrastive

objectives may alter the original embedding space and reduce compatibility with existing retrieval systems. In contrast, RCEM eliminates the need for conversation-to-passage relevance supervision and explicitly attempts to preserve the original embedding space, improving generalization, scalability, and compatibility with existing retrieval infrastructures.

3 Proposed RCEM

We propose RCEM, a robust conversational search embedder that directly outputs an embedding for conversational search. Unlike conventional conversational search pipelines that rely on explicit LLM-based query rewriting during inference, RCEM intrinsically converts a conversational query into an LLM-rewritten query and generates an embedding of the rewritten query in a single step, which reduces inference computational cost and latency for conversational search. RCEM enables this by *additionally* injecting LLM query reformulation capability into the embedder while attempting to preserve the original embedder’s capability. RCEM is described in detail in this section.

Setting. Let q_i denote the current query at dialogue turn i , and let $q_{<i}, a_{<i}$ denote the conversational history. Thus, the conversational query is $q_i; q_{<i}, a_{<i}$. Existing conversational search systems commonly use an LLM-based rewriter

$$r_i = \text{LLM}(q_i; q_{<i}, a_{<i}), \quad (1)$$

where r_i is an LLM-rewritten query suitable for retrieval. To distinguish conversational search from standard retrieval, we prepend a special token $[ST]$ to the conversational query,

$$c_i = [ST] : q_i; q_{<i} a_{<i}. \quad (2)$$

Embedder architecture. Two embedders are defined: RCEM $F_\theta(x)$ and the original embedder $G(x)$, where

$$F_\theta(x) = \text{MLP}_{\text{SELU}}(\text{LoRA}(G(x))). \quad (3)$$

$F_\theta(x)$ is built on top of the frozen $G(x)$ with LoRA adapters and a lightweight two-layer perceptron head with SELU activation (Klambauer et al., 2017). The backbone parameters are initialized from $G(x)$, while only the LoRA adapters and the MLP head, parameterized with θ , are updated during training.

Structure Learning (SL). $F_\theta(x)$ is trained by minimizing the loss below,

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{point}}(\theta) + \lambda \mathcal{L}_{\text{pair}}(\theta), \quad (4)$$

where

$$\mathcal{L}_{\text{point}}(\theta) = \sum_{j=0}^{l+m+n} \|F_\theta(x_{j,1}) - G(x_{j,2})\|, \quad (5)$$

$$\mathcal{L}_{\text{pair}}(\theta) = \sum_{j=0}^{l+m+n} \sum_{k>j}^{l+m+n} \left\| \|F_\theta(x_{j,1}) - F_\theta(x_{k,1})\| - \|G(x_{j,2}) - G(x_{k,2})\| \right\|. \quad (6)$$

$x_{j,1}$ and $x_{j,2}$ are the first and second elements of the j -th pair in a set of pairs,

$$\Omega = \{(q_i, q_i)\}_{i=0}^l \cup \{(p_i, p_i)\}_{i=0}^m \cup \{(c_i, r_i)\}_{i=0}^n. \quad (7)$$

Ω consists of three subsets of matching pairs, l pairs of general queries (q_i) to themselves, m pairs of passages (p_i) to themselves, and n conversational queries prepended by $[ST]$ to their LLM-rewritten queries. The last set of matching pairs are for training RCEM to align a conversational queries prepended by the special token toward an LLM-rewritten queries, $F_\theta(c_i) \rightarrow G(r_i)$. This enables RCEM to learn the LLM-based rewriting capability and directly encode the conversational query into the embedding of the rewritten query without explicitly generating r_i during inference. The first two sets of pairs, $\{(q_i, q_i)\}_{i=0}^l \cup \{(p_i, p_i)\}_{i=0}^m$ are for preserving the embedding space. While distilling LLM’s query rewriting capability into the embedder, RCEM also attempts to preserve the embedding space by mapping $F_\theta(q_i) \rightarrow G(q_i)$ and $F_\theta(p_i) \rightarrow G(p_i)$.

Unlike Yang et al. (2025b), which optimizes only pointwise embedding distances, RCEM also preserves pairwise structure during training. The hyperparameter λ controls the weight of the pairwise structure preservation term, and is set to 1.0 unless otherwise specified. Figure 2 provides an overview of RCEM and contrasts it with Lin et al. (2021). A more comparison between RCEM and Yang et al. (2025b) is presented in Appendix D.

This design offers several advantages. First, RCEM eliminates expensive LLM-based rewriting during inference, substantially reducing latency

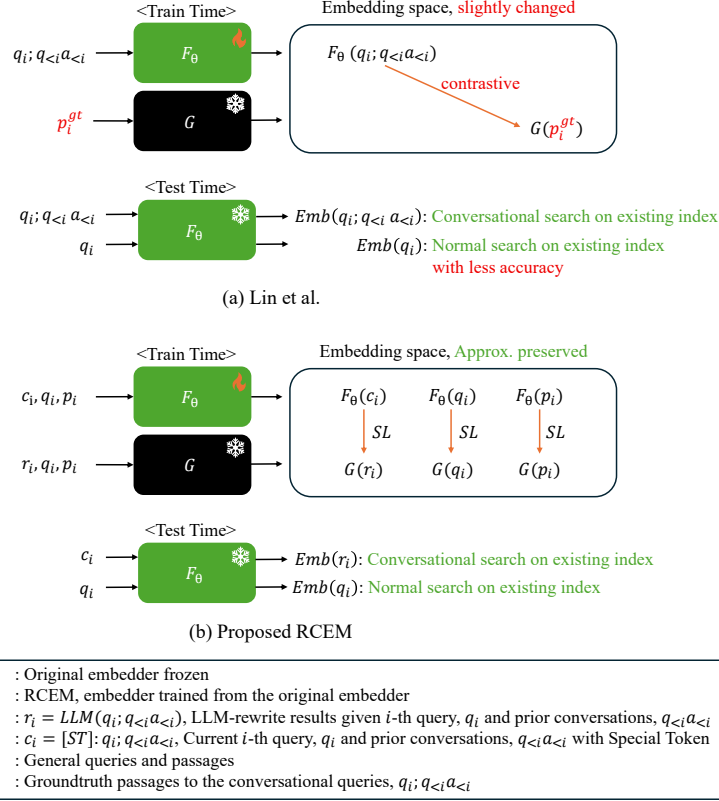


Figure 2: **RCEM overview and comparison with prior work.** RCEM first precomputes embeddings of the LLM-rewritten query, $r_i = LLM(q_i; q_{<i} a_{<i})$, as well as general queries q_i and passages p_i using the original frozen embedder G . RCEM F_θ is then trained to map conversational queries prepended by a special token, $F_\theta(c_i)$, to the embedding of the rewritten query produced by the frozen embedder $G(r_i)$ using the proposed Structure Learning (SL) loss. During training, RCEM also aligns general queries and passages with their precomputed embeddings, i.e., $F_\theta(q_i) \rightarrow G(q_i)$ and $F_\theta(p_i) \rightarrow G(p_i)$, to preserve the original embedding space. This design enables a single model to support both conversational search, triggered by the special token, and standard search over the index built by the original embedder. Attempting to preserve the embedding space also serves as a regularizer, making the model more robust to distributional shift. In contrast, Lin et al. (2021) trains the model to map the conversational query embedding $F_\theta(q_i; q_{<i} a_{<i})$ directly to the ground-truth passage $G(p_i^{gt})$, which is a more complicated training task, resulting in overfitting. Unlike Lin et al. (2021); Yang et al. (2025b), RCEM does not require conversational-to-passage relevance annotations for training, which are often difficult to obtain accurately. This makes RCEM more scalable, as it reduces the need for expensive and high-quality relevance annotations.

and computational cost. Second, RCEM does not require conversational-query-to-passage relevance supervision, making training more scalable. Finally, the approximately preserved original embedding space serves as a regularizer during training that increases robustness in distributional shift and remains compatible with existing indexes built by the original embedder.

4 Experiments

Experiment setting. To evaluate RCEM, we use widely adopted conversational search benchmarks: QReCC (Anantha et al., 2021), TopiOCQA (Adlakha et al., 2022), and TREC CAsT (Dalton et al., 2020, 2021). QReCC and TopiOCQA provide both train and test splits, whereas TREC

CAsT is primarily used for zero-shot evaluation. QReCC contains approximately 14K conversations and 80K question-answer pairs, with retrieval performed over a corpus of roughly 54M passages. TopiOCQA comprises 3.9K conversations and 51K question-answer pairs, paired with a corpus of approximately 25M passages. Together, these benchmarks enable evaluation of RCEM in both in-domain and distribution-shifted conversational retrieval settings. For consistency with prior work (Lai et al., 2025a; Yang et al., 2025b; Lin et al., 2021), we report Recall@10, MRR, and NDCG@3 using pytreval (Van Gysel and de Rijke, 2018).

We compare RCEM against state-of-the-art CDR (Lin et al., 2021; Yang et al., 2025b) and

Type	Method	QReCC			TopiOCQA		
		R@10	MRR	NDCG@3	R@10	MRR	NDCG@3
CQR	ConvGQR (Mo et al., 2023)	63.5	42.0	39.1	41.8	25.6	24.3
CQR	LLM4CS, $N = 16$ (Mao et al., 2023b)	64.6	43.2	40.7	55.1	35.4	34.5
CQR	CHIQ-Fusion (Mo et al., 2024a)	70.7	47.2	44.2	61.6	38.0	37.0
CQR	AdaCQR+Expansion (Lai et al., 2025b)	67.3	45.8	42.9	58.4	38.5	37.6
CQR	AdaRewriter, $N = 16$ (Lai et al., 2025a)	69.8	47.5	44.7	61.9	40.3	39.7
CQR	Query Rewrite (GPT-4.1) → Embedding	73.3	47.3	45.4	64.6	38.1	37.6
CDR	Lin et al. (2021)	67.5	39.3	37.4	66.3	40.3	40.1
CDR	Yang et al. (2025b)	–	36.8	–	–	42.2	–
CDR	Proposed RCEM	75.8	49.9	48.3	70.4	43.2	42.9

Table 1: **In-domain retrieval accuracy comparison.** RCEM consistently outperforms both the conversational query reformulation (CQR) and conversational dense retrieval (CDR) methods, achieving $\sim 10\%$ and $\sim 13\%$ higher MRR than Lin et al. (2021) and Yang et al. (2025b) on QReCC respectively.

Method	Test dataset	Train dataset	R@10	MRR	NDCG@3
Query Rewrite (GPT-4.1) → Embedding	QReCC	–	73.3	47.3	45.4
Lin et al. (2021)		TopiOCQA	68.0	42.2	40.5
Proposed RCEM		TopiOCQA	76.3	51.3	49.8
Query Rewrite (GPT-4.1) → Embedding	TopiOCQA	–	64.6	38.1	37.6
Lin et al. (2021)		QReCC	44.1	24.3	23.4
Proposed RCEM		QReCC	65.3	38.6	37.8
Query Rewrite (GPT-4.1) → Embedding	TREC CAsT 2019	–	90.2	73.4	46.7
Lin et al. (2021)		QReCC	76.3	48.7	26.7
Proposed RCEM		QReCC	93.1	77.4	51.8
Lin et al. (2021)		TopiOCQA	89.0	63.4	38.1
Proposed RCEM		TopiOCQA	93.6	76.2	50.5
Query Rewrite (GPT-4.1) → Embedding	TREC CAsT 2020	–	87.0	65.9	37.8
Lin et al. (2021)		QReCC	64.9	40.1	19.7
Proposed RCEM		QReCC	86.5	64.3	39.3
Lin et al. (2021)		TopiOCQA	79.8	52.1	26.4
Proposed RCEM		TopiOCQA	88.0	68.1	39.5

Table 2: **Retrieval accuracy comparison in distributional shift.** Models are trained on one dataset and evaluate them on another. The proposed RCEM method maintains strong performance under distributional shift, whereas the performance of Lin et al. (2021) drops noticeably. In particular, when trained on QReCC and tested on TREC CAsT 2019, RCEM achieves $\sim 30\%$ higher MRR than Lin et al. (2021).

CQR (Mo et al., 2023; Mao et al., 2023b; Mo et al., 2024a; Lai et al., 2025b,a) methods. Unless otherwise specified, both RCEM and Lin et al. (2021) use Qwen3-8B (Yang et al., 2025a) as the original embedder. RCEM uses GPT-4.1 to generate rewritten queries for training with a simple query-rewriting prompt (Appendix B). For a fair comparison, we also evaluate a Query Rewrite → Embedding pipeline using the same LLM (prompt) and embedding model as RCEM. All models are trained with a batch size of 32 on eight H100 GPUs. Unlike prior methods (Lin et al., 2021; Yang et al., 2025b), RCEM does not require conversation-to-passage relevance labels for training in any of the experiments. Additional implementation details are provided in Appendix E.

In domain experiments. We conduct in-domain evaluations on the QReCC and TopiOCQA datasets. For a fair comparison with Lin et al. (2021), we keep the experimental setup largely identical, including the hardware configuration, adaptation architecture, and number of training epochs. The primary difference lies in the training objective. We also compare against a Query Rewrite → Embedding baseline that uses the same GPT-4.1 query rewriter and Qwen3-8B embedder as RCEM. As shown in Table 1, RCEM consistently outperforms prior methods, achieving $\sim 10\%$ and $\sim 13\%$ higher MRR than Lin et al. (2021) and Yang et al. (2025b) on QReCC, respectively.

Notably, RCEM also surpasses the Query Rewrite → Embedding baseline despite using the same LLM and embedding models, which hypo-

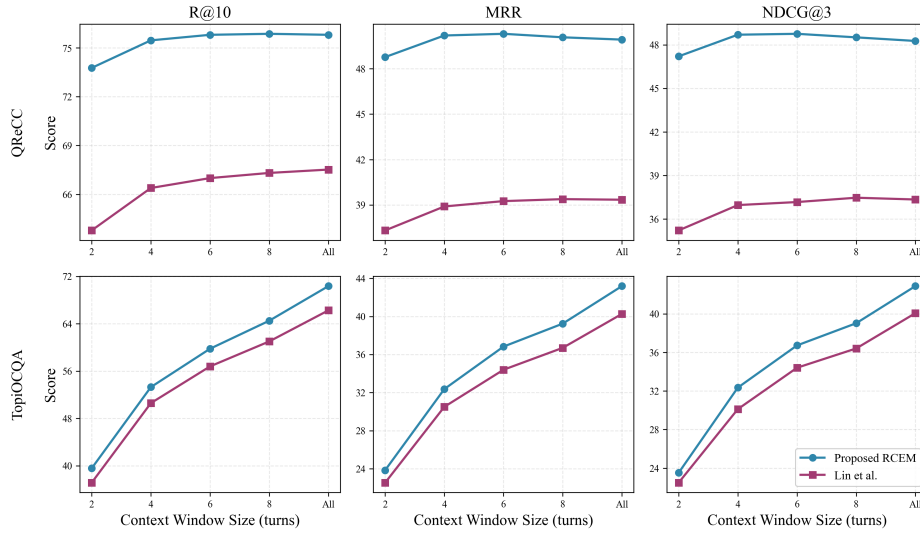


Figure 3: **Accuracy comparison VS context window size.** On the TopiOCQA benchmark, the retrieval accuracy of RCEM improves more with additional conversational turns, suggesting that RCEM is more effective at identifying and leveraging relevant information from long conversation histories.

thetically suggests that treating the rewritten query as an intermediate textual representation may introduce information loss. In contrast, RCEM directly aligns conversational queries with the embedding representation of rewritten queries, producing a more faithful and effective retrieval representation.

Distributional shift experiments. In the distribution-shift experiments, Lin et al. (2021) exhibits substantial performance degradation, whereas RCEM remains robust under distribution shift. Table 2 reports retrieval performance when models are trained on one dataset and evaluated on another under an identical experimental setup. RCEM consistently outperforms both Lin et al. (2021) and the Query Rewrite $\rightarrow$ Embedding baseline across all dataset combinations and evaluation metrics. In particular, when trained on QReCC and evaluated on TREC CAsT 2019, Lin et al. (2021) performs substantially worse than RCEM, with accuracy gaps often ranging from 10% to 30%. Similar trends are observed when the same experiments are conducted using a smaller base embedder, Qwen3-0.6B (Appendix C).

This performance degradation is likely attributable to the contrastive objective used by Lin et al. (2021), which directly aligns long conversational queries with long passages. Such a learning objective is inherently more complex and can cause the model to overfit to the training dataset. In contrast, RCEM maps conversational inputs to shorter rewritten queries, providing a simpler learning tar-

get while preserving the original embedding space. Attempting to preserving the embedding space also acts as an effective regularizer, improving robustness to distribution shift.

Accuracy VS context window. We further conduct an ablation study to examine the impact of conversational history length on retrieval performance. Figure 3 shows that RCEM consistently outperforms Lin et al. (2021) across different context window sizes, with accuracy gains increasing as more conversational history is provided. In particular, RCEM benefits more from longer contexts on TopiOCQA, suggesting that it is better able to capture and leverage salient information distributed across multi-turn conversations.

Structure learning vs L_2 . A key differentiator of RCEM from Yang et al. (2025b) is the proposed Structure Learning (SL) loss, which preserves both pointwise embedding similarity and pairwise structure during training. Table 4 shows that the SL loss consistently achieves better retrieval performance than the L_2 loss used in Yang et al. (2025b) on both QReCC and TopiOCQA.

Approximately preserving original embedding space. Figure 4 shows whether RCEM preserves the original embedding space. Specifically, we encode general queries (the GT rewrites query in QReCC) using RCEM and perform dense retrieval over a QReCC index built by the original embedder. We then compare the retrieval accuracy of

Type	Method	R@10	MRR	NDCG@3	Latency
CQR	AdaRewriter, $N = 16$ (Lai et al., 2025a)	69.8	47.5	44.7	3,656 ms
CQR	Query Rewrite (GPT-4.1) with baseline prompt $\rightarrow$ Embedding	73.3	47.3	45.4	1,275 ms
CQR	Query Rewrite (GPT-4.1) with Mao et al. (2023a) $\rightarrow$ Embedding	75.1	49.3	47.5	2,495 ms
CDR	Lin et al. (2021)	67.5	39.3	37.4	69 ms
CDR	Proposed RCEM with baseline prompt	75.8	49.9	48.3	69 ms
CDR	Proposed RCEM with Mao et al. (2023a)	78.1	52.3	50.8	69 ms

Table 3: **Effect of rewrite quality on RCEM and latency.** When RCEM is trained with a stronger query rewriting method (Mao et al., 2023a), retrieval accuracy on QReCC dataset is better than baseline prompt setting, while maintaining the latency in inference.

Benchmark	Loss	R@10	MRR	NDCG@3
QReCC	SL	75.8	49.9	48.3
	L_2	74.4	47.2	45.2
TopiOCQA	SL	70.4	43.2	42.9
	L_2	53.9	30.4	29.7

Table 4: **Loss comparison.** Structural Learning (SL) loss proposed in RCEM consistently better performs than L_2 loss used in Yang et al. (2025b) on both QReCC and TopiOCQA datasets.

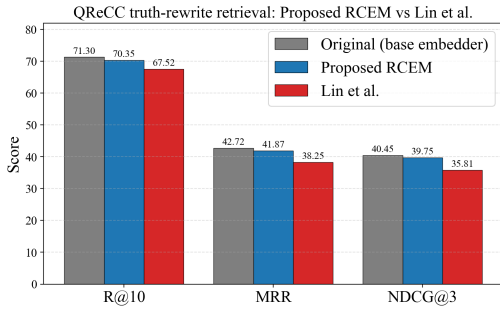


Figure 4: **Approximately preserving original embedding space.** General query is embedded by RCEM and run retrieval on index built from the original embedding. RCEM better preserves retrieval accuracy than Lin et al. (2021) compared with the original embedder. The experiment is performed with Qwen3-0.6B base embedder.

RCEM against the original embedder and Lin et al. (2021). The results show that RCEM experiences a small performance drop compared to the original embedder, whereas Lin et al. (2021) suffers a larger degradation. We attribute this difference to the fact that Lin et al. (2021) does not explicitly preserve the original embedding space during training.

Effect of better rewrite quality and latency. Table 3 compares RCEM with the Query Rewrite $\rightarrow$ Embedding baseline on QReCC using either a simple prompting strategy or a stronger query rewriting method (Mao et al., 2023a). The improved rewriting method enhances the retrieval accuracy of both

approaches, indicating that rewrite quality has a direct impact on retrieval effectiveness. Nevertheless, RCEM consistently outperforms the two-step pipeline under the same rewriting strategy. Notably, RCEM uses query rewriting only during training so that better prompting strategy adds up better retrieval accuracy with inference latency unchanged. In contrast, the two-step baseline requires query rewriting at inference time, meaning that improved rewrite quality often comes at the cost of higher computational overhead and increased latency.

Table 3 also compares inference latency. In general, conversational dense retrieval (CDR) methods exhibit lower latency than conversational query reformulation (CQR) methods because CDR do not require LLM calls during inference. RCEM achieves latency comparable to existing CDR methods while delivering higher accuracy than both CDR and CQR baselines. Additional details on computational cost are provided in Appendix A.

5 Conclusion

We present RCEM, a conversational search embedder that implicitly maps conversational queries to LLM-rewritten queries and generates their embeddings without requiring explicit LLM-based query rewriting during inference. Unlike prior methods that address the challenging task of directly aligning long conversational contexts with long passages, RCEM maps conversational queries to shorter LLM-rewritten queries while attempting to preserve the original embedding space, thereby improving generalization and remaining compatible with existing retrieval indexes built by the original embedder. In addition, RCEM does not require expensive conversation-to-passages relevance supervision. Extensive experiments demonstrate that RCEM achieves strong retrieval accuracy and improved robustness under distribution shift while substantially reducing inference latency.

Limitations

The primary limitation of RCEM lies in the LLM’s query rewriting capability, as the model is designed to distill LLM-based rewriting behavior into the embedder. Although the results in Table 1 show that RCEM consistently outperforms the two-step pipeline (Query Rewrite → Embedding), hypothetically because it learns a richer latent representation than explicit rewritten text, its accuracy still depends on the quality of the rewritten queries. As shown in Table 3, stronger query rewriting methods lead to higher retrieval accuracy for RCEM. Therefore, further improvements in query rewriting, such as incorporating advanced rewriting techniques (Lai et al., 2025a) or leveraging multiple query rewrites as additional conversation-to-rewrite training pairs, could further enhance RCEM’s retrieval accuracy.

References

- Vaibhav Adlakha, Shehzaad Dhuliawala, Kaheer Suleman, Harm de Vries, and Siva Reddy. 2022. [Top-iOCQA: Open-domain conversational question answering with topic switching](#). *Transactions of the Association for Computational Linguistics*, 10:468–483.
- Unsloth AI. 2024. Unsloth: Fast and memory-efficient fine-tuning for large language models. <https://github.com/unslothai/unsloth>.
- Raviteja Anantha, Svitlana Vakulenko, Zhucheng Tu, Shayne Longpre, Stephen Pulman, and Srinivas Chappidi. 2021. [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*, pages 520–534, Online. Association for Computational Linguistics.
- Yiruo Cheng, Kelong Mao, and Zhicheng Dou. 2024. [Interpreting conversational dense retrieval by rewriting-enhanced inversion of session embedding](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2879–2893, Bangkok, Thailand. Association for Computational Linguistics.
- Jeffrey Dalton, Chenyan Xiong, and Jamie Callan. 2020. Cast 2019: The conversational assistance track overview. In *In Proceedings of TREC*.
- Jeffrey Dalton, Chenyan Xiong, and Jamie Callan. 2021. Cast 2020: The conversational assistance track overview. In *In Proceedings of TREC*.
- Jianfeng Gao, Chenyan Xiong, Paul Bennett, and Nick Craswell. 2023. *Neural approaches to conversational information retrieval*, volume 44. Springer.
- Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. 2017. Self-normalizing neural networks. *Advances in neural information processing systems*, 30.
- Yilong Lai, Jialong Wu, Zhenglin Wang, and Deyu Zhou. 2025a. [AdaRewriter: Unleashing the power of prompting-based conversational query reformulation via test-time adaptation](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 3889–3905, Suzhou, China. Association for Computational Linguistics.
- Yilong Lai, Jialong Wu, Congzhi Zhang, Haowen Sun, and Deyu Zhou. 2025b. [Adacqr: Enhancing query reformulation for conversational search via sparse and dense retrieval alignment](#). In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 7698–7720, Abu Dhabi, UAE. Association for Computational Linguistics.
- Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. 2021. [Contextualized query embeddings for conversational search](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1004–1015, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Kelong Mao, Chenlong Deng, Haonan Chen, Fengran Mo, Zheng Liu, Tetsuya Sakai, and Zhicheng Dou. 2024. [Chatretriever: Adapting large language models for generalized and robust conversational dense retrieval](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1227–1240, Miami, Florida, USA. Association for Computational Linguistics.
- Kelong Mao, Zhicheng Dou, Bang Liu, Hongjin Qian, Fengran Mo, Xiangli Wu, Xiaohua Cheng, and Zhao Cao. 2023a. [Search-oriented conversational query editing](#). In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4160–4172, Toronto, Canada. Association for Computational Linguistics.
- Kelong Mao, Zhicheng Dou, Fengran Mo, Jiewen Hou, Haonan Chen, and Hongjin Qian. 2023b. [Large language models know your contextual search intent: A prompting framework for conversational search](#). In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 1211–1225. Association for Computational Linguistics.
- Fengran Mo, Abbas Ghaddar, Kelong Mao, Mehdi Rezagholizadeh, Boxing Chen, Qun Liu, and Jian-Yun Nie. 2024a. [Chiq: Contextual history enhancement for improving query rewriting in conversational search](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2268, Miami, Florida, USA. Association for Computational Linguistics.

- Fengran Mo, Kelong Mao, Yutao Zhu, Yihong Wu, Kaiyu Huang, and Jian-Yun Nie. 2023. [Convqqr: Generative query reformulation for conversational search](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4998–5012, Toronto, Canada. Association for Computational Linguistics.
- Fengran Mo, Chen Qu, Kelong Mao, Tianyu Zhu, Zhan Su, Kaiyu Huang, and Jian-Yun Nie. 2024b. [History-aware conversational dense retrieval](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 13366–13378, Bangkok, Thailand. Association for Computational Linguistics.
- Christophe Van Gysel and Maarten de Rijke. 2018. `Py trec_eval`: An extremely fast python interface to `trec_eval`. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 873–876.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. 2025a. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.
- Seunghan Yang, Juntae Lee, Jihwan Bang, Kyuhong Shim, Minsoo Kim, and Simyung Chang. 2025b. [Learning contextual retrieval for robust conversational search](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 11991–12003, Suzhou, China. Association for Computational Linguistics.
- Fanghua Ye, Meng Fang, Shenghui Li, and Emine Yilmaz. 2023. [Enhancing conversational search: Large language model-aided informative query rewriting](#). In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5985–6006, Singapore. Association for Computational Linguistics.

Type	Method	Latency	Cost/1M Qry.	MRR
CQR	Query Rewrite (GPT4.1) $\rightarrow$ Embedding (Qwen3-8B)	1,275 ms	\$570.4	47.3
CQR	Query Rewrite (GPT4.1) + Mao et al. (2023a) $\rightarrow$ Embedding (Qwen3-8B)	2,495 ms	\$840.4	49.3
CQR	AdaRewriter, $N = 16$ (Lai et al., 2025a) (Llama-3.1-8B)	3,656 ms	\$20.8	47.5
CDR	Lin et al. (2021) (Qwen3-8B)	69 ms	\$1.77	39.3
CDR	Yang et al. (2025b) (Qwen2.5-7B)	41 ms	\$1.77	36.8
CDR	Proposed RCEM (Qwen3-8B)	69 ms	\$1.77	49.9
CDR	Proposed RCEM + Mao et al. (2023a) (Qwen3-8B)	69 ms	\$1.77	52.3

Table 5: **Inference efficiency and accuracy on QReCC.** Latency, estimated recurring serving cost, and retrieval performance for conversational query reformulation (CQR) and conversational dense retrieval (CDR) methods.

A Latency and Computational Cost Analysis

We evaluate inference latency and estimate recurring serving cost in Table 5. Following Yang et al. (2025b), latency is measured over the complete QReCC test set including all conversations. NVIDIA H100 GPU is used for the evaluation. We use the following public serving prices:

- GPT-4.1: \$2 per million input tokens and \$8 per million output tokens from Azure OpenAI Service pricing¹ (accessed August 2, 2026).
- Llama-3.1-8B-Instruct: \$0.02 per million input tokens and \$0.04 per million output tokens from OpenRouter (DeepInfra)² (accessed August 2, 2026).
- Qwen2.5-7B-Instruct: \$0.01 per million tokens from OpenRouter (Phala)³ (accessed August 2, 2026).
- Qwen3-Embedding-8B: \$0.01 per million input tokens from OpenRouter (DeepInfra)⁴ (accessed August 2, 2026).

RCEM achieves higher retrieval accuracy than existing CDR methods while maintaining comparable latency and serving cost. Compared with the Query Rewrite $\rightarrow$ Embedding pipeline, RCEM reduces latency from 1,275 ms to 69 ms and lowers the estimated serving cost from \$570.4 to \$1.77 per million queries, while improving MRR from 47.3 to 49.9. RCEM also outperforms Lai et al. (2025a) in retrieval accuracy while requiring substantially lower latency and serving cost.

Stronger offline query rewriting methods (Mao et al., 2023a) further improve RCEM’s MRR from 49.9 to 52.3 without increasing inference latency or serving cost. This is possible because the improved query rewriting is used only during offline training-data generation and is never invoked at inference time.

¹<https://azure.microsoft.com/en-us/pricing/details/azure-openai/>

²<https://openrouter.ai/meta-llama/llama-3.1-8b-instruct#providers>

³<https://openrouter.ai/qwen/qwen-2.5-7b-instruct>

⁴<https://openrouter.ai/qwen/qwen3-embedding-8b#providers>

B Baseline Prompt

We use a simple prompt to convert each conversational query into a rewritten query for generating the training dataset, as shown below:

Given the recent conversation:
{conversation_history}

Rewrite the following ambiguous query
into a self-contained and more clear query.

This is the ambiguous query:
{query}

We enforce GPT4.1 to follow schema to reject redundant output token such as

The rewritten query is ...

The clear query is ...

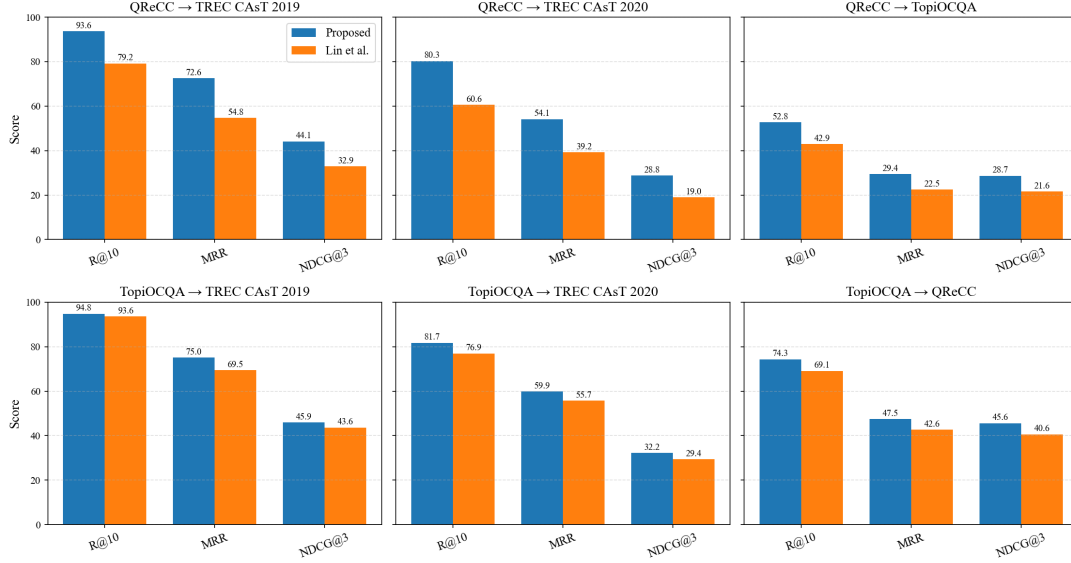


Figure 5: **Retrieval accuracy comparison in distributional shift with 0.6B Qwen3 base embedder:** We train the models on one dataset and evaluate them on another under the same experimental setup to ensure a fair comparison. The proposed RCEM method maintains strong performance under distributional shift, whereas the performance of Lin et al. (2021) drops noticeably. In particular, when trained on QReCC and tested on TREC CAsT 2020, RCEM achieves approximately 20% higher Recall@10 than Lin et al. (2021). Across all dataset combinations and evaluation metrics, RCEM consistently outperforms Lin et al. (2021).

λ	R@10	MRR	NDCG@3
0	53.9	30.4	29.7
0.25	69.8	42.2	41.8
0.5	69.9	42.4	42.2
1.0	70.4	43.2	42.9
2.0	68.8	42.1	41.7
4.0	57.9	33.5	33.2

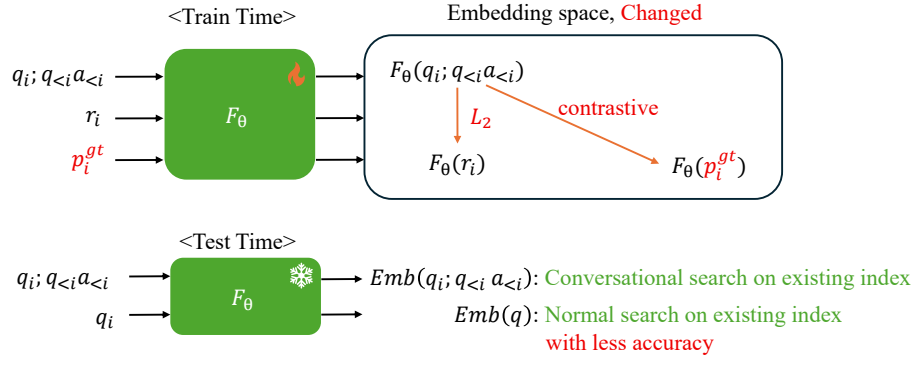
Table 6: **Sensitivity to the structure-learning weight.** TopiOCQA retrieval performance as the pairwise structure-preservation weight λ varies. The setting used in the main experiments is shown in bold.

C More Experiments

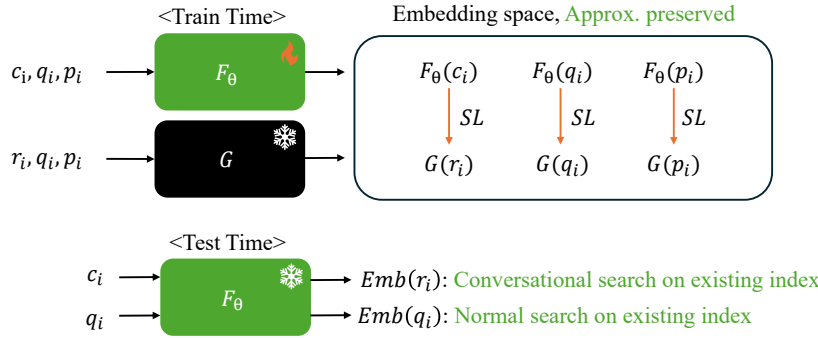
We perform the distributional shift experiments with 0.6B Qwen3 base model. The proposed RCEM method maintains strong performance under distributional shift, whereas the performance of Lin et al. (2021) drops noticeably with 0.6B Qwen3 base model experiments. See Figure 5.

We analyze the sensitivity of RCEM to λ , which controls the weight of the pairwise structure preservation objective relative to the pointwise alignment objective. Table 6 reports retrieval performance on TopiOCQA. Performance remains stable over the broad range $\lambda \in [0.25, 2.0]$: R@10 varies by only 1.5 points, MRR by 1.1 points, and NDCG@3 by 1.2 points. The value used in the main experiments, $\lambda = 1.0$, achieves the best result on all three metrics.

The extreme settings further clarify the role of pairwise structure preservation. Setting $\lambda = 0$ removes the pairwise objective and reduces R@10, MRR, and NDCG@3 by 16.5, 12.8, and 13.2 points, respectively, relative to $\lambda = 1.0$. Conversely, $\lambda = 4.0$ overemphasizes pairwise geometry at the expense of pointwise conversational-to-rewrite alignment. These results indicate that pairwise structure preservation is important, while RCEM is not sensitive to moderate choices around the selected value.



(a) Yang et al.



(b) Proposed RCEM

G	: Original embedder frozen
F_θ	: RCEM, embedder trained from the original embedder
r_i	: $r_i = LLM(q_i; q_{<i} a_{<i})$, LLM-rewrite results given i -th query, q_i and prior conversations, $q_{<i} a_{<i}$
c_i	: $c_i = [ST]; q_i; q_{<i} a_{<i}$, Current i -th query, q_i and prior conversations, $q_{<i} a_{<i}$ with Special Token
q_i, p_i	: General queries and passages
p_i^{gt}	: Groundtruth passages to the conversational queries, $q_i; q_{<i} a_{<i}$

Figure 6: **RCEM, comparison with prior work.** Different from RCEM, Yang et al. (2025b) trains the model to map the conversational query embedding $F_\theta(q_i; q_{<i} a_{<i})$ directly to the ground-truth passage $G(p_i^{gt})$ with contrastive loss and also maps to rewritten query $F_\theta(r_i)$ with L_2 loss. Yang et al. (2025b) requires conversational queries and relevant passages for their training. Because Yang et al. (2025b) does not use embeddings produced by the original encoder G as training anchors, the learned representations can drift from the original embedding space. This embedding-space shift may degrade compatibility with existing document indexes encoded by G , whereas RCEM is designed to preserve embedding-space compatibility while adapting conversational queries.

D More Comparison with Prior Works

We compare RCEM with Yang et al. (2025b) in this Section. See Figure 6 for more details.

E More Implementation Details

RCEM is trained using the Unsloth training framework (AI, 2024). The head projects the base embedding into an intermediate representation of size 3×1024 , followed by a projection back to the 1024-dimensional embedding space. Note that we only use the first 1024 embeddings from the base model. We use LoRA rank=32 and apply adapters to all attention and feed-forward projection layers, including q proj, k proj, v proj, o proj, gate proj, up proj, and down proj. Training is performed with the Adam optimizer using a learning rate of 3×10^{-4} , no weight decay, and a cosine learning-rate schedule with a warmup ratio of 0.1. To preserve compatibility with the original retrieval embedding space, we jointly optimize the embedding preservation objective using a randomly sampled 1% subset of passages, standalone queries, and conversational queries from the training data.