Query-focused and Memory-aware Reranker for Long Context Processing

Anonymous ACL submission

Abstract

Built upon the existing analysis of retrieval heads in large language models, we propose an alternative reranking framework that trains models to estimate passage–query relevance using the attention scores of selected heads. This approach provides a listwise solution that leverages the holistic information within the entire candidate shortlist during ranking. At the same time, it naturally produces continuous relevance scores, enabling training on arbitrary retrieval datasets without requiring Likert-scale supervision. Our framework is lightweight and effective, requiring only small-scale models (*e.g.*, 4B parameters) to achieve strong performance. Extensive experiments demonstrate that our method outperforms existing state-of-the-art pointwise and listwise rerankers across multiple domains, including Wikipedia and long narrative datasets. It further establishes a new state-of-the-art on the LoCoMo benchmark that assesses the capabilities of dialogue understanding and memory usage. We further demonstrate that our framework supports flexible extensions. For example, augmenting candidate passages with contextual information further improves ranking accuracy, while training attention heads from middle layers enhances efficiency without sacrificing performance.

1 Introduction

Embedding Models, especially those built on top of LLMs, achieved successes and enabled generators (RAG) and agents to work with long inputs or large input corpora efficiently (Zhang et al., 2025b; Zhao et al., 2025; Babakhin et al., 2025; Li et al., 2025a). However, embeddings also have limitations, as theoretically proved and empirically illustrated by (Weller et al., 2025). They reveal a "geometric bottleneck" where fixed-dimensional vectors fail to encode the combinatorial complexity of query-document interactions. Furthermore, the inductive bias of the similarity measure limits the applicable domains where other types of relationships are required to recall, *e.g.*, causality, associations, and analogy.

A long line research applies an additional reranker module on the shortlist returned from embedding models to resolve this challenge. The rerankers use larger models, more powerful representations (like cross-attention). The fast development of LLMs boosts many LLM-based reranker releases to benefit from the reasoning capabilities of LLMs (Zhang et al., 2025b; Sun et al., 2025; Liu et al., 2025a; Pradeep et al., 2023b). These rerankers can adopt either pointwise or listwise formulations. Pointwise lost the global view of the shortlist, but can give scores. Listwise approaches, on the other hand, directly inherit the long-context reasoning and text generation ability of the backbone LLMs, which takes a holistic view of the shortlist, but the next-token prediction limits the prediction of fine-grained scores, and the predicted float numbers cannot always accurately reflect the true confidence (Liu et al., 2025b; Lin et al., 2024). As a result, they adopt a Likert rating regime, asking the models to output a five-point or ten-point scale score for each input document, which limits the available training data.

In this work, we propose an alternative solution built upon the existing analysis of retrieval heads in LLMs (Wu et al., 2024; Zhang et al., 2025a). These works identify two related types of heads: retrieval heads and Query-focused Retrieval (QR) heads. Both refer to attention heads whose attention patterns reflect retrieval behaviors. Specifically, when concatenating long contexts of relevant and distractor passages with the query, these heads are defined as those that put significant attention weights on the relevant passages, so as the ranks of attention weights correlate with the ranks of relevance.

While existing works mainly focus on probing and understanding the functions of such heads, our work moves one step further by training LLMs to optimize the ranking accuracy of a small set of

retrieval heads. In this way, we achieve an LLM-ranker that is optimized to rank passages with attention weights. This resulted listwise solution, named **QRRanker**, can naturally work with continuous relevance scores without the limitation of Likert-scale supervision, hence can be trained on arbitrary retrieval datasets.

Our QRRanker enjoys several good properties in practice. First, the retrieval heads can be effectively trained even when the backbone has a relatively small scale, *e.g.*, 4B parameters. This allows the listwise approach to run with improved efficiency. Second, it is easy to enhance the input candidate passages with their global context with efficiency, by prepending the shared contextual information to the ground of candidates during training, which is essential for long narrative understanding. Finally, we observed that our QRRanker is quite robust to the selection of heads, and training with heads from layers in the middle would result in no performance drop. This allows us to take off the higher layers of the LLMs during training and inference, which can greatly reduce the latency of the model.

Experiments on various domains, including Wikipedia QA tasks (Musique (Trivedi et al., 2022), HotpotQA (Yang et al., 2018)), long narrative QA tasks (NarrativeQA (Kočiskỳ et al., 2018), DetectiveQA (Xu et al., 2025b)) and long-term dialogue (LoCoMo (Maharana et al., 2024)), demonstrate the advantage of our QRRanker. As a versatile ranking framework, our approach not only outperforms the state-of-the-art general-purpose pointwise and listwise models like Qwen-Rerank and GroupRank, but also consistently improves over the domain-specific ranking approaches, such as HippoRAG-v2 (Guti'errez et al., 2025) for Wikipedia QA and a list of recent memory-enhanced approaches (Li et al., 2025b; Rasmussen et al., 2025) on LoCoMo.

2 Related Work

Reranking Ranking techniques are commonly constructed based on two structures: Siamese network (Bi-encoder; Koch et al. 2015) and Cross-encoder (Thakur et al., 2021). Embedding models (Zhang et al., 2025b) is the first one usually used to rank the whole corpus of documents with embeddings stored for reuse. However, they are limited by the "geometric bottleneck", failing to encode more fine-grained interactions between query and document. The limitation can be alleviated by cross-encoders, which score every document by cross-attention with the query. The computing burden of dedicated re-encoding every pair of query-document narrows the way for cross-encoders only reranking top-n documents ranked by bi-encoders, which produces refined sorting of top docs. Therefore, cross-encoders are called Rerankers.

In the era of LLMs, Rerankers are also deeply explored using LLMs. They can be classified into two groups: Pointwise and Listwise. Pointwise describes the paradigm of pairwise scoring for documents, which is the major direction (Qin et al., 2024; Sun et al., 2023; Liu et al., 2025a; Zhuang et al., 2025) in practice, *e.g.*, Qwen3 (Zhang et al., 2025b), Jina, mGTE (Zhang et al., 2024), BGE-m3 (Chen et al., 2024) rerankers. Pointwise models independently encode documents, failing to grasp global information. To this end, Listwise models fully utilize LLMs' generating ability. They (Pradeep et al., 2023a,b) concatenate documents as a list and generate the reranking result accordingly. To step further, tuned using RL, models can first think (Sun et al., 2023; Liu et al., 2025a; Qin et al., 2025; Ma et al., 2023; Sun et al., 2025) and then give the answer, achieving significant performance. However, Listwise models require training data to provide a specific ranking of docs or even scores, leading to burdens of data collection and construction. Furthermore, LLMs' generation is not stable (*e.g.*, generating bad formats), especially when introducing the thinking process. As studied by Wu et al. (2024); Zhang et al. (2025a), LLMs inherently possess the ability of retrieval, and retrieval attention heads can be extracted to rank docs, achieving competitive performance. Nevertheless, these heads may change when moving to new tasks, requiring additional seed datasets to extract them. To this end, we propose to train the selected heads, which ensures a better transferability.

Memory Utilization Memory construction and utilization to alleviate problems of long-context processing has become a hot spot nowadays. For long story understanding (Zhou et al., 2025; Kočiskỳ et al., 2018; Xu et al., 2025b; Wu et al., 2025), Li et al. (2025a) construct global memory to enhance retrieval and generation. For dialogue management, sophisticated graphs (Jiang et al., 2026; Xu et al., 2025a; Rasmussen et al., 2025; Hu et al., 2026b,a; Zhou et al., 2026), trees (Li et al., 2026), and systems (Chhikara et al., 2025; Li et al., 2025b; Nan et al., 2025; Tao et al., 2026;

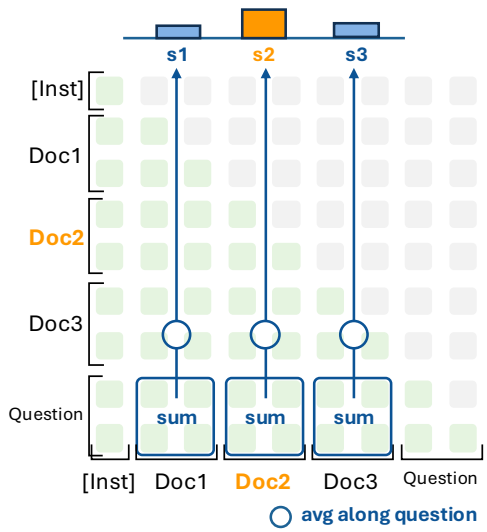

Figure 1: The retrieval score and QR score are computed based on the attention score of a (QR) attention head. In this figure, Doc2 is the gold document (chunk).

Zou et al., 2026) of events, personas, and chunks are designed to accurately extract related dialogue history for further use. However, a more powerful search for history, with simple memory construction, can beat complicated memory management, and we will show our solution to reach this goal.

3 Preliminaries: QR-head

In this section, we first introduce the definition of the Query-Focused Retrieval heads (QR-head).

As introduced by Wu et al. (2024) and Zhang et al. (2025a), among all heads in multi-head self-attention modules, some play crucial roles as retrievers. These heads pay more attention to the parts containing information to answer the question of the context when encoding the question. Zhang et al. (2025a) name them as QR-heads and identify them by QR score.

Formally, for a question Q, its corresponding context C is split into a chunk list $[c_0, c_1, ..., c_n]$, where $G = [c_{g0}, ..., c_{gm}]$ are gold chunks to answer the question. The attention score of an attention head h between Q and every chunk c_i when encoding the prompt with C and Q is denoted as $A_h^{Q \to c_i} \in \mathbb{R}^{|Q| \times |c_i|}$. A head's QR score is computed by summing up the attention scores of gold chunks:

$$\texttt{QRScore}_h = \frac{1}{|Q|} \sum_{c_i \in G} \sum_{w_q \in Q} \sum_{w_c \in c_i} A_h^{Q \to c_i}[w_q, w_c], \quad (1)$$

where w_c and w_Q are tokens in gold chunk c_i and Q respectively. The QR score measures the extent to which h focuses on gold chunks. A higher value indicates that the head has the potential to identify G. The QR score will be computed and averaged on the seed dataset for every head $h \in H$, sort H descendingly, and then pick up the top 16 heads as the QR heads ($h \in H_{QR}$). We select QR heads for `Qwen3-4B-Instruct-2507` (Yang et al., 2025) by 1000 random samples from NarrativeQA.

QR heads compute the retrieval score for a chunk c_i in a similar way like Eq. 1 by replacing $\sum_{c \in G}$ with $\sum_{h \in H_{QR}}$. Zhang et al. (2025a) further adds a score calibration to mitigate intrinsic biases in attention weights, which encodes a null query N="N/A" and subtracts its $\frac{1}{|N|} \sum_{w_q \in N} A^{N \to c_i}$. Notably, with our QR training, calibration becomes optional.

4 Method

Our QRRanker is the Listwise method that reranks all the documents in a single inference pass, following the so-called "prompt-decoders" (Pradeep et al., 2023a). Notably, QRRanker does not involve any generation processes, but only prefills the prompt with the question and documents, and obtains the attention scores, which is more time- and resource-friendly. Though the original QR retriever (Zhang et al., 2025a) with a group of precomputed QR heads can transfer to new tasks, the performance may not be that stable, as QR heads may be changed on new tasks. To this end, we propose a dedicated training pipeline for QRRanker. We first construct listwise training instances and then optimize the precomputed QR heads with a contrastive ranking objective.

4.1 Data Construction for QR Training

4.1.1 Listwise Training Instances

We build a unified training set by combining MuSiQue (Trivedi et al., 2022) and NarrativeQA (Kočiskỳ et al., 2018). We first determine evidence chunks for each question. For MuSiQue, we directly use the official supporting facts in the original annotations as evidence. For NarrativeQA, since gold chunks are not provided, we follow Li et al. (2025a) to construct *silver* evidence chunks.

After establishing the evidence, we retrieve a top-50 candidate set for each question using Qwen3-Embedding-8B and form a listwise instance by labeling retrieved candidates that match the pre-constructed evidence as positive, while treating the remaining retrieved candidates as negatives.

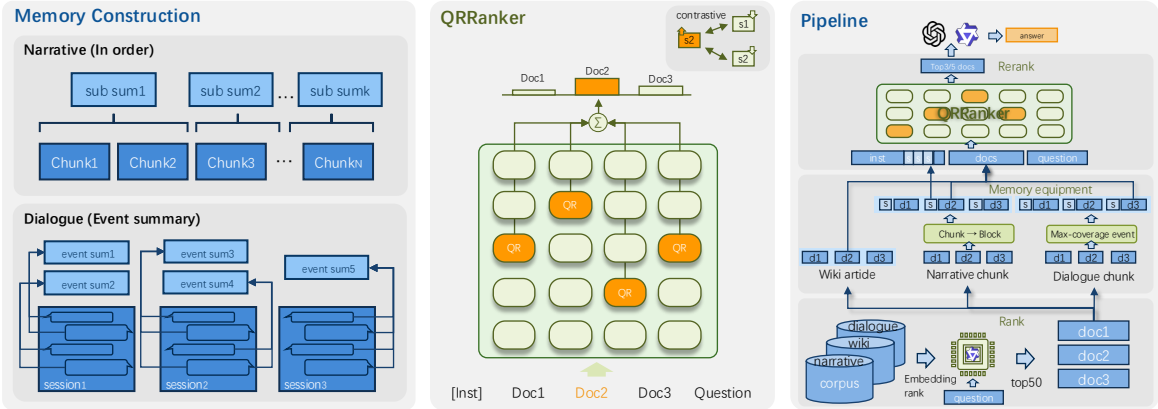

Figure 2: The structure of QRRanker is illustrated in the middle, where the highlighted heads are QR heads for document scoring. As QRRanker can be aware of memory enhancement to capture more contextual information, we can construct memories for narratives and dialogues, which is shown on the left. The right part demonstrates the rank-rerank pipeline of qa for narratives/wiki/dialogues, which involves no sophisticated design.

Optionally, we build a *summary prefix* by mapping the retrieved chunks to their corresponding summaries and prepending these summaries to the chunk list, *i.e.*, $X = [M; C]$. The detailed construction procedure for NarrativeQA is provided in alg. 1 in Appendix A. MuSiQue follows the same pipeline, except that relevant evidence is taken directly from its official supporting facts. We describe the summary construction process in the next subsection.

4.1.2 Summary Construction

To provide high-level semantic guidance and support long-context narrative understanding, we construct summaries as auxiliary memory context. When used, summaries are prepended as a global prefix to the retrieved chunk list, so the model can leverage both coarse-grained context and fine-grained evidence. We explore two complementary strategies for constructing summaries.

Block-based Summary. For long narrative books, we employ a block-based summarization strategy that respects the temporal flow of the narrative. Specifically, we segment each book into blocks (20 consecutive chunks per block) and generate a corresponding summary for each. This process is detailed in Appendix B.1.

Event-centric Summary. For dialogue-based data, we extract structured events from conversations and form an event-centric summary. Each event is represented by a short description and is linked to its source utterances, enabling traceability to the original dialogue. (Refer to Appendix B.2 for details).

4.2 QR Training

Obtaining QR heads precomputed by the QR score mentioned in § 3, our training scheme focuses on training these heads. For a question Q and the top 50 candidate documents $C = [c_1, ..., c_{50}]$ ranked by a retriever (*e.g.*, embedding models like Qwen3-Embedding), where gold (positive) documents are $G = [c_{g0}, .., c_{gm}]$, the prompt input to QRRanker is constructed by concatenating C and Q in order with some instructions: $\mathtt{P} = \text{Inst}(C, Q)$, where the instruction template is provided in Appendix B.3.

The prompt $\mathtt{P}$ is fed into the model, and in every attention head, the attention score is computed as $A_h^{\mathtt{P}\to\mathtt{P}}$. We locate the position of Q and $c_i \in C$ and take out the query-focused part $A_h^{Q\to c_i}$. The retrieval score of the passage c_i computed by the QR head $h \in H_{QR}$ is:

$$s_{c_i}^h = \frac{1}{|Q|} \sum_{i \in c_i} \sum_{j \in Q} A_h^{Q \to c_i}[i, j], \qquad (2)$$

where the score computing is illustrated in Fig. 1. Then, the final retrieval score is obtained by summing up all scores provided by QR heads: $s_{c_i} = \sum_{h \in H_{QR}} s_{c_i}^h$. Additionally, $s_{c_i}^h$ can also be computed by aggregating the maximum attention item, like used in approaches like ColBERT (Khattab and Zaharia, 2020), which achieves similar performance, so we do not discuss it here.

We then optimize the document scores $S = [s_{c_1}, ..., s_{c_{50}}]$ utilizing the sample-level contrastive loss. In a conventional contrastive scene, the score s_{c_i} is stably ranged in [0, 1], while, in our case, s_{c_i} can be affected by tokens in the instruction (*e.g.*, the head's sensitivity to attention sink), which may

lead to an unstable range for samples. Therefore, the temperature may not be suitable for scaling the score. To this end, we normalize the score with the max-min norm, which can be formed as:

$$S = \frac{scale \times (S - \min(S))}{\max(S) - \min(S)}, \tag{3}$$

where $scale$ is a factor to scale the range to [0, $scale$] for stability.

The original contrastive loss samples one positive document at a time; however, the top 50 documents may contain more than one positive document. It can be suboptimal if we follow the original setting, as unselected positive documents are ignored. We propose a group version of contrastive loss to simultaneously optimize them:

$$L_{sample} = \frac{1}{|G|} \sum_{c_p \in G} \log \frac{\tau(s_{c_p})}{\tau(s_{c_p}) + \sum_{c_n \in C \setminus G} \tau(s_{c_n})}, \tag{4}$$

where τ denotes the exponential function. The objective above treats every positive document as an independent sub-sample and averages the loss inside the sample. For the dataset, the objective aligns with conventional contrastive loss.

As our QRRanker can be made memory-aware to incorporate broader contextual information, during QR training, we optionally prepend a memory prefix M (*e.g.*, summaries mapped from the retrieved chunks) before the candidate list C. The resulting prompt to QRRanker is constructed as $\mathsf{P} = \text{Inst}(M, C, Q)$.

5 Experimental Setup

5.1 Datasets

To evaluate QRRanker across diverse retrieval settings, we conduct experiments on benchmarks spanning Wikipedia multi-hop QA, long-context story QA, and dialogue memory.

Wikipedia Multi-hop QA For fact-based multi-hop retrieval, we evaluate on **HotpotQA** (Yang et al., 2018) and **MuSiQue** (Trivedi et al., 2022). To ensure a fair comparison, we adopt the corpus and test splits provided by **HippoRAG** (Guti'errez et al., 2025), maintaining consistency in the candidate passage pool.

Long-context Story QA We utilize datasets that demand complex reasoning over extended contexts, specifically: (1) **NarrativeQA** from the HELMET benchmark (Yen et al., 2024), which consists of 1,272 questions with the longest document reaching 518k tokens. (2) **DetectiveQA** (Xu et al., 2025b) is a bilingual detective story dataset with an average length exceeding 100k tokens, requiring precise evidence localization across scattered plot points.

Long-term dialogue memory We evaluate our model on **LoCoMo** (Maharana et al., 2024), a large-scale benchmark designed for long-term dialogue memory. The dataset comprises 50 multi-session dialogues across 10 distinct user groups, with each dialogue averaging approximately 9,000 tokens. Following prior work, we report performance across four fine-grained categories: single-hop, multi-hop, temporal reasoning, and open-domain.

5.2 Baselines

We evaluate QRRanker against a broad spectrum of retrieval and memory frameworks.

For general-purpose reranking on Wikipedia QA and Long-context story tasks, we compare QRRanker against two categories of models: (1) **Embedding Models**: Qwen3-Embedding (4B/8B) (Zhang et al., 2025b) and SFT-Embedding-8B, which is fine-tuned from Qwen3-Embedding-8B on our constructed data. (2) **Reranking Methods**: HippoRAG (Jimenez Gutierrez et al., 2024; Guti'errez et al., 2025), GroupRank-32B (Sun et al., 2025), Qwen3-Reranker-4B (out-of-box) (Zhang et al., 2025b), and a Qwen3-Reranker-4B variant trained on the same data as our QRRanker. We also include the QRHead without training as a baseline.

For the long-term dialogue task on LoCoMo, we compare QRRanker with a range of strong baselines, including: A-Mem (Xu et al., 2025a), MemoryOS (Li et al., 2025b), Zep (Rasmussen et al., 2025), Mem0 (Chhikara et al., 2025), Nemori (Nan et al., 2025), and LightMem (Fang et al., 2025); TiMem (Li et al., 2026), Synapse (Jiang et al., 2026), Membox (Tao et al., 2026), Compass-Mem (Hu et al., 2026b), and ES-Mem (Zou et al., 2026); SimpleMem (Liu et al., 2026). Detailed baseline descriptions are provided in Appendix C.

5.3 Implementation Details

Our QRRanker is trained on `Qwen3-4B-Instruct-2507`, with QR heads selected as described in Appendix D. In the training process, the $scale$ factor in the max-min norm is set to 8; the batch size is set to 1; the gradient accumulating step is set to 4; the learning rate is set to 1e-5. We utilize the DeepSpeed ZERO2 strategy and train QRRanker

Methods	Wikipedia QA						Story QA						Overall Avg@k		
	Musique			HotpotQA			NarrativeQA			DetectiveQA					
	R@3	R@5	R@10	R@3	R@5	R@10	R@3	R@5	R@10	R@3	R@5	R@10	avg@3	avg@5	avg@10
Embedding Methods															
Qwen3-Embedding-4B	51.56	59.83	69.88	78.84	86.16	92.33	12.57	18.33	28.08	19.25	26.17	37.04	40.56	47.62	56.83
Qwen3-Embedding-8B	54.35	62.55	72.47	82.85	89.05	95.15	14.98	20.92	32.39	12.84	20.00	31.17	41.25	48.13	57.80
SFT-Embedding-8B	45.11	52.93	62.03	82.36	88.63	94.19	21.31	29.77	44.17	19.84	27.59	39.00	42.16	49.73	59.85
Reranking Methods															
HippoRAG-v1	–	53.20	–	–	90.40	–	–	–	–	–	–	–	–	–	–
HippoRAG-v2	–	74.70	–	–	96.30	–	–	–	–	–	–	–	–	–	–
Qwen-Reranker-4B (out-of-box)	57.60	66.37	74.26	89.80	94.15	96.75	20.83	28.25	41.98	23.42	30.50	42.09	47.91	54.82	63.77
Qwen-Reranker-4B (trained)	61.60	69.71	77.49	89.35	93.95	96.90	25.84	35.05	49.62	29.67	38.92	51.25	51.61	59.41	68.82
GroupRank-32B*	55.49	65.08	73.07	82.45	90.60	94.50	23.98	33.76	48.83	29.34	39.21	51.38	47.82	57.16	66.95
QRHeads-4B (out-of-box)	63.12	71.22	78.99	90.20	94.80	96.90	24.28	33.44	48.89	23.71	32.89	45.58	50.33	58.09	67.59
Our QRRanker-4B	**70.19**	**77.37**	**82.13**	**95.05**	**96.90**	**97.70**	**29.11**	**38.89**	**54.93**	**32.22**	**41.32**	**53.76**	**56.64**	**63.62**	**72.13**

Table 1: Retrieval and Rerank performance measured by Recall@{k}. '–' indicates the metric is not reported in the corresponding paper. For Wikipedia QA, we rerank the top-50 candidates retrieved by Qwen3-Embedding-8B; for Story QA, we rerank the top-50 candidates retrieved by SFT-Embedding-8B. DetectiveQA scores are averaged over English and Chinese sets. Overall columns report avg@3/avg@5/avg@10 averaged over the four datasets. Bold numbers indicate the best result in each column. * For fairness, all rerankers are evaluated with a single run.

Methods	R@3	R@5	R@10
Qwen3-Emb-8b	58.61	67.67	79.15
SFT-Emb-8b	76.01	83.10	90.15
GroupRank-32B	77.99	82.94	88.14
QRHeads (out-of-box)	85.93	90.35	94.86
QRRanker (ours)	**87.34**	**91.32**	**95.01**
Improvement vs. SFT-Emb	+11.33	+8.22	+4.86

Table 2: Retrieval and Rerank performance on LoCoMo.

using 8 H20 GPUs.

For downstream QA evaluation, we use task-specific prompting for generation; the full prompt templates for NarrativeQA, DetectiveQA, and LoCoMo are provided in Appendix B. We employ **Qwen3-8B** as the generator for NarrativeQA and DetectiveQA, where books are chunked into non-overlapping passages of ∼200 tokens. For the LoCoMo benchmark, we utilize **GPT-4o-mini** and **GPT-5-mini** as the generators. We segment the dialogue history into small chunks, ensuring that utterance continuity is preserved, with an average chunk size of 258 tokens. When enabling the memory-aware setting, we prepend a summary prefix before the ranked chunk list. We cap the summary prefix at 512 tokens and select summaries based on their coverage of the retrieved/reranked chunks.

6 Results

6.1 Main Results

We conduct experiments across three representative long-context settings: multi-hop question answering over Wikipedia, long-story question answering, and dialogue memory, covering five datasets in both English and Chinese. Tables 1 and 2 summarize reranking performance in terms of Recall@k, while Tables 3 and 4 report downstream generation results. Overall, QRRanker consistently achieves the best overall results across settings, demonstrating improvements in both retrieval quality and downstream task performance.

Rerank Performance. We first analyze the retrieval effectiveness of QRRanker when applied to rerank the top-50 candidates retrieved by SFT-Embedding-8B. As shown in Table 1, QRRanker establishes a new state-of-the-art benchmark. It surpasses the Qwen-Reranker-4B by a substantial margin and improves the average recall significantly. On Wikipedia datasets such as Musique and HotpotQA, QRRanker outperforms complex graph-based methods like HippoRAG (Guti'errez et al., 2025). Remarkably, it also exceeds the performance of GroupRank-32B (Sun et al., 2025) despite being significantly more lightweight. This indicates that our method captures inter-passage dependencies more effectively than simple groupwise scoring or graph traversal.

The performance gap is particularly evident in the story domain, where context tracking is critical. For instance, QRRanker achieves a Recall@10 of 54.93 on NarrativeQA compared to 48.83 for GroupRank and 48.89 for the out-of-box QRHeads. Moreover, Table 2 shows that QRRanker maintains the same advantage on LoCoMo, further demonstrating its effectiveness in retrieving relevant context from long conversational histories.

LLM	Method	Tokens	Single-hop	Multi-hop	Temporal	Open-domain	Overall F1
GPT-4o-mini	Qwen3-Emb-8B (out-of-box)	846	47.95	35.24	41.36	24.79	42.81
GPT-4o-mini	SFT-Emb-8B	841	57.22	37.06	56.27	29.11	51.58
GPT-4o-mini	A-Mem (Xu et al., 2025a)†	2,712	44.65	27.02	45.85	12.14	39.65
GPT-4o-mini	MemoryOS (Li et al., 2025b)†	3,874	48.62	35.27	41.15	20.02	42.84
GPT-4o-mini	Zep (Rasmussen et al., 2025)†	3,911	49.56	35.74	42.00	19.37	43.56
GPT-4o-mini	Mem0 (Chhikara et al., 2025)†	1,764	47.65	38.72	48.93	28.64	45.09
GPT-4o-mini	Nemori (Nan et al., 2025)†	4,767	46.33	32.36	55.99	29.19	44.72
GPT-4o-mini	LightMem (Fang et al., 2025)†	815	47.64	32.11	53.79	26.14	44.73
GPT-4o-mini	TiMem (Li et al., 2026)	511	–	–	–	–	54.40
GPT-4o-mini	Synapse (Jiang et al., 2026)	814	48.90	35.70	50.10	25.90	40.50
GPT-4o-mini	Membox (Tao et al., 2026)	2,166	60.09	39.88	58.03	27.96	53.10
GPT-4o-mini	CompassMem (Hu et al., 2026b)	20,000	57.36	38.84	57.96	26.61	52.18
GPT-4o-mini	ES-Mem (Zou et al., 2026)†	2,925	50.07	36.52	47.90	24.77	45.56
GPT-4.1-mini	SimpleMem (Liu et al., 2026)	531	51.12	43.46	58.62	19.76	43.24
GPT-4o-mini	**QRRanker (Ours)**	854	**62.95**	**43.06**	**61.90**	**29.79**	**57.03**
GPT-5-mini	**QRRanker (Ours)**	854	**61.78**	**44.73**	**64.53**	**31.04**	**57.32**

Table 3: Comparison with SOTA Memory and Agent frameworks on the LoCoMo. Results marked with † are derived from ES-Mem (Zou et al., 2026). For QRRanker, we rerank the top-50 chunks retrieved by SFT-Emb-8B and utilize only the top-3 chunks as context for generation, without additional memory mechanisms. '–' indicates the metric is not reported in the corresponding paper.

Methods	NarrativeQA F1	NarrativeQA EM	DetectiveQA ACC
Embedding Methods			
Qwen3-Embedding-8B	26.30	11.01	57.35
SFT-Embedding-8B	28.48	12.11	62.85
Reranking Methods			
Qwen3-Reranker-4B (vanilla)	29.10	12.58	60.93
Qwen3-Reranker-4B (trained)	30.51	13.52	64.52
QRRanker Series			
QRHeads-4B	31.40	14.70	64.75
QRRanker	**33.61**	**16.04**	**67.25**

Table 4: QA performance on NarrativeQA and DetectiveQA. All methods utilize R@3 retrieved chunks as the context for generation (Qwen3-8B as Generator).

Dataset	QRRanker		
	Chunk	+Sum	Δ
LoCoMo	86.64	87.34	**+0.70**
NarrativeQA	28.09	29.11	**+1.02**
DetectiveQA	29.55	32.22	**+2.67**
HotpotQA	95.05	94.75	**-0.30**
Musique	70.19	70.16	**-0.03**

Table 5: Recall@3 comparison of QRRanker with chunk-only inputs versus a summary prefix (+Sum) as contextual memory.

Long-context Story QA Performance. High-quality retrieval should translate to improved generation accuracy. We evaluate this on narrative understanding datasets. As shown in Table 4, QRRanker significantly improves downstream QA performance. On NarrativeQA, it achieves 33.61 F1, outperforming the trained Qwen3-Reranker-4B (30.51). On DetectiveQA, accuracy increases from 62.85 (SFT-Embedding-8B) to 67.25 with QRRanker. These results suggest that QRRanker selects evidence that is not only semantically relevant, but also better aligned with the reasoning needed for answer generation.

Dialogue Memory Performance As summarized in Table 3, QRRanker achieves the best Overall F1 on LoCoMo with a highly compact input budget. Using only 854 tokens on average (top-3 chunks) from the raw dialogue history, it attains 57.03 Overall F1 with GPT-4o-mini and 57.32 with GPT-5-mini. In contrast, many memory-augmented baselines require substantially larger token budgets to maintain explicit memory stores or graph structures. QRRanker instead reranks the top-50 chunks retrieved by the embedding retriever and feeds only a few top-ranked raw dialogue chunks to the generator. This simple and lightweight design remains highly effective at capturing long-range dependencies, yielding the superior overall performance in our LoCoMo comparison.

6.2 Results with Contextual Information

As shown in Table 5, equipping QRRanker with a summary prefix consistently improves ranking performance across long-dialogue and long-context story benchmarks. This suggests that the summary provides global contextual guidance, comple-

menting the fine-grained evidence from retrieved chunks. Moreover, we test summary-based memory on Wikipedia-based multi-hop QA. We build a hierarchical clustering tree over retrieved passages and use parent summaries as the prefix. However, this strategy brings no gains and can even degrade performance, suggesting that abstracted global summaries are less helpful when evidence is highly localized in Wikipedia passages.

6.3 Results with Heads from Different Layer-Levels

QRRanker uses static preset heads, which invokes our curiosity about the heads from which level of layers are suitable as starters for QR training. We propose a variant that dynamically selects heads from a range of continuous layers for every sample. The variant totally picks up 16 heads from layer l_s to l_e with $16/(l_e - l_s)$ heads per layer, where l_s-l_e determines the level of layers (*i.e.*, low, middle, high). Details of the variant are elaborated in Appendix E. We train and evaluate both QRRanker and its variants on the NarrativeQA dataset.

Methods	R@3	R@5	R@10
QRRanker	28.87	39.16	54.44
10-17	24.51	34.52	49.91
17-24	28.15	39.07	54.28
28-35	28.48	38.88	54.65

Table 6: Retrieval performance on NarrativeQA of QRRanker and its variants adapted on different levels of layers. $l_s - l_e$ denotes the layers with head selection.

As shown in Tab. 6, training models with lower layers 10-17 shows a significant performance drop, while middle layers 17-24 and top layer 28-35 almost keep the same performance as QRRanker. Intuitively, lower layers truncate too much knowledge from higher layers, and heads in the middle-to-top layers are more likely to be retrievers. The outcome aligns with the phenomenon that QR heads in QRRanker are all positioned in the middle layers (17-24). Interestingly, we compare QR heads in QRRanker with those selected by the variant (17-24), and the degree of overlap is pretty low. It indicates that, with QR training, such potential is activated, which shows that our method can utilize the robustness of heads, even not QR heads, from the middle to the top. This provides a way to only focus on heads in the middle and truncate the higher layers for a smaller and faster ranker.

Method	P50 (ms)	P95 (ms)	TFLOPs (/query)	Peak Mem (GB)
Qwen3-Reranker (batch=50)	1221.59	1256.29	115.69	13.88
Qwen3-Reranker (batch=1)	1895.26	1929.09	113.65	**7.78**
QRRanker	1095.42	1133.38	82.74	11.18
QRRanker (middle)	**910.42**	**928.1**	**69.83**	8.71

Table 7: Inference efficiency comparison in latency (P50/P95), compute (TFLOPs per query), and peak GPU memory. All models are evaluated under the same hardware and inference settings over 20 queries. `QRRanker(middle)` truncates the model after layer 24.

We quantify the inference efficiency benefits of this middle-layer truncation in § 6.4.

6.4 Inference Efficiency

We compare the computational efficiency of our method with baselines on 20 random samples. Table 7 shows that `QRRanker` achieves lower P50/P95 latency than `Qwen3-Reranker-4B`, while also reducing TFLOPs and peak memory. Moreover, `QRRanker(middle)` further improves efficiency by truncating the model after layer 24, discarding higher layers. It achieves the best P50/P95 latency with additional reductions in compute and memory. For `Qwen3-Reranker-4B`, we report two inference settings. With *batch=50*, all 50 chunk–query pairs are processed in a single forward pass. With *batch=1*, the 50 pairs are processed with 50 separate forward passes, which substantially increases latency. Overall, QRRanker provides a better performance and cost trade-off, and the truncated middle-layer variant offers an especially lightweight and fast option.

7 Conclusion

In this paper, we present **QRRanker**, a lightweight and efficient listwise reranking framework built on Query-focused Retrieval (QR) heads in LLMs. By explicitly training selected QR heads for ranking, QRRanker produces real-valued relevance scores and performs reranking without generation at inference time. Across five datasets spanning Wikipedia multi-hop QA, long-context story QA, and dialogue memory, QRRanker consistently improves reranking quality and downstream QA performance. QRRanker remains practical with a small backbone (*e.g.*, 4B) and offers clear inference efficiency benefits. Moreover, it supports simple extensions such as an optional summary prefix for global context and mid-layer head selection for further efficiency.

Limitations

While QRRanker demonstrates strong performance across multiple domains and datasets, several limitations remain. First, our evaluation mainly focuses on a single 4B backbone. Although this controlled design helps isolate the effect of the proposed reranking mechanism, broader generalization across model families and scales remains to be studied. Still, the consistent gains across diverse datasets suggest that the learned ranking signal is not tied to a single benchmark. Second, part of our training supervision relies on silver evidence rather than fully human-annotated gold evidence, due to the lack of fine-grained evidence annotations in narrative-style QA benchmarks. This may introduce label noise, especially when partially relevant passages are not covered by the constructed evidence set. Nevertheless, the consistent improvements across datasets indicate that QRRanker remains reasonably robust under this realistic weak-supervision setting.

References

Yauhen Babakhin, Radek Osmulski, Ronay Ak, Gabriel Moreira, Mengyao Xu, Benedikt Schifferer, Bo Liu, and Even Oldridge. 2025. Llama-embed-nemotron-8b: A universal text embedding model for multilingual and cross-lingual tasks. *Preprint*, arXiv:2511.07025.

Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, volume ACL 2024 of *Findings of ACL*, pages 2318–2335. Association for Computational Linguistics.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. 2025. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*.

Jizhan Fang, Xinle Deng, Haoming Xu, Ziyan Jiang, Yuqi Tang, Ziwen Xu, Shumin Deng, Yunzhi Yao, Mengru Wang, Shuofei Qiao, and 1 others. 2025. Lightmem: Lightweight and efficient memory-augmented generation. *arXiv preprint arXiv:2510.18866*.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *J. Mach. Learn. Res.*, 23:120:1–120:39.

Bernal Jim'enez Guti'errez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. 2025. From rag to memory: Non-parametric continual learning for large language models. In *arXiv.org*.

Chuanrui Hu, Xingze Gao, Zuyi Zhou, Dannong Xu, Yi Bai, Xintong Li, Hui Zhang, Tong Li, Chong Zhang, Lidong Bing, and 1 others. 2026a. Evermemos: A self-organizing memory operating system for structured long-horizon reasoning. *arXiv preprint arXiv:2601.02163*.

Yuyang Hu, Jiongnan Liu, Jiejun Tan, Yutao Zhu, and Zhicheng Dou. 2026b. Memory matters more: Event-centric memory as a logic map for agent searching and reasoning. *arXiv preprint arXiv:2601.04726*.

Hanqi Jiang, Junhao Chen, Yi Pan, Ling Chen, Weihang You, Yifan Zhou, Ruidong Zhang, Yohannes Abate, and Tianming Liu. 2026. Synapse: Empowering llm agents with episodic-semantic memory via spreading activation. *arXiv preprint arXiv:2601.02744*.

Bernal Jimenez Gutierrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2024. Hipporag: Neurobiologically inspired long-term memory for large language models. *Advances in Neural Information Processing Systems*, 37:59532–59569.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 39–48.

Gregory Koch, Richard Zemel, Ruslan Salakhutdinov, and 1 others. 2015. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2, pages 1–30. Lille.

Tomáš Kočiskỳ, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The narrativeqa reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328.

Kai Li, Xuanqing Yu, Ziyi Ni, Yi Zeng, Yao Xu, Zheqing Zhang, Xin Li, Jitao Sang, Xiaogang Duan, Xuelei Wang, and 1 others. 2026. Timem: Temporal-hierarchical memory consolidation for long-horizon conversational agents. *arXiv preprint arXiv:2601.02845*.

Yuqing Li, Jiangnan Li, Zheng Lin, Ziyan Zhou, Junjie Wu, Weiping Wang, Jie Zhou, and Mo Yu. 2025a. Mindscape-aware retrieval augmented generation for improved long context understanding. *arXiv preprint arXiv:2512.17220*.

Zhiyu Li, Chenyang Xi, Chunyu Li, Ding Chen, Boyu Chen, Shichao Song, Simin Niu, Hanyu Wang, Jiawei Yang, Chen Tang, and 1 others. 2025b. Memos: A memory os for ai system. *arXiv preprint arXiv:2507.03724*.

Lei Lin, Jiayi Fu, Pengli Liu, Qingyang Li, Yan Gong, Junchen Wan, Fuzheng Zhang, Zhongyuan Wang, Di Zhang, and Kun Gai. 2024. Just ask one more time! self-agreement improves reasoning of language models in (almost) all scenarios. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3829–3852.

Jiaqi Liu, Yaofeng Su, Peng Xia, Siwei Han, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. 2026. Simplemem: Efficient lifelong memory for llm agents. *arXiv preprint arXiv:2601.02553*.

Wenhan Liu, Xinyu Ma, Weiwei Sun, Yutao Zhu, Yuchen Li, Dawei Yin, and Zhicheng Dou. 2025a. Reasonrank: Empowering passage ranking with strong reasoning ability. *arXiv preprint arXiv:2508.07050*.

Xiaoou Liu, Tiejin Chen, Longchao Da, Chacha Chen, Zhen Lin, and Hua Wei. 2025b. Uncertainty quantification and confidence calibration in large language models: A survey. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 6107–6117.

Xueguang Ma, Xinyu Zhang, Ronak Pradeep, and Jimmy Lin. 2023. Zero-shot listwise document reranking with a large language model. *CoRR*, abs/2305.02156.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. 2024. Evaluating very long-term conversational memory of llm agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13851–13870.

Jiayan Nan, Wenquan Ma, Wenlong Wu, and Yize Chen. 2025. Nemori: Self-organizing agent memory inspired by cognitive science. *arXiv preprint arXiv:2508.03341*.

Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023a. Rankvicuna: Zero-shot listwise document reranking with open-source large language models. *arXiv preprint arXiv:2309.15088*.

Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023b. Rankzephyr: Effective and robust zero-shot listwise reranking is a breeze! *arXiv preprint arXiv:2312.02724*.

Xubo Qin, Jun Bai, Jiaqi Li, Zixia Jia, and Zilong Zheng. 2025. Tongsearch-qr: Reinforced query reasoning for retrieval. *CoRR*, abs/2506.11603.

Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Le Yan, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, Xuanhui Wang, and Michael Bendersky. 2024. Large language models are effective text rankers with pairwise ranking prompting. In *Findings of the Association for Computational Linguistics: NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, volume NAACL 2024 of *Findings of ACL*, pages 1504–1518. Association for Computational Linguistics.

Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. 2025. Zep: a temporal knowledge graph architecture for agent memory. *arXiv preprint arXiv:2501.13956*.

Duolin Sun, Meixiu Long, Dan Yang, Yihan Jiao, Zhehao Tan, Jie Feng, Junjie Wang, Yue Shen, Peng Wei, Jian Wang, and 1 others. 2025. Grouprank: A groupwise reranking paradigm driven by reinforcement learning. *arXiv preprint arXiv:2511.11653*.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 2023. Is chatgpt good at search? investigating large language models as re-ranking agents. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 14918–14937. Association for Computational Linguistics.

Dehao Tao, Guoliang Ma, Yongfeng Huang, and Minghu Jiang. 2026. Membox: Weaving topic continuity into long-range memory for llm agents. *arXiv preprint arXiv:2601.03785*.

Nandan Thakur, Nils Reimers, Johannes Daxenberger, and Iryna Gurevych. 2021. Augmented SBERT: data augmentation method for improving bi-encoders for pairwise sentence scoring tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 296–310. Association for Computational Linguistics.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. ♪ musique: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

Orion Weller, Michael Boratko, Iftekhar Naim, and Jinhyuk Lee. 2025. On the theoretical limitations of embedding-based retrieval. *arXiv preprint arXiv:2508.21038*.

Junjie Wu, Jiangnan Li, Yuqing Li, Lemao Liu, Liyan Xu, Jiwei Li, Dit-Yan Yeung, Jie Zhou, and Mo Yu. 2025. Sitemb-v1. 5: Improved context-aware dense retrieval for semantic association and long story comprehension. *arXiv preprint arXiv:2508.01959*.

Wenhao Wu, Yizhong Wang, Guangxuan Xiao, Hao Peng, and Yao Fu. 2024. Retrieval head mechanistically explains long-context factuality. *ArXiv*, abs/2404.15574.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. 2025a. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*.

Zhe Xu, Jiasheng Ye, Xiaoran Liu, Xiangyang Liu, Tianxiang Sun, Zhigeng Liu, Qipeng Guo, Linlin Li, Qun Liu, Xuanjing Huang, and Xipeng Qiu. 2025b. DetectiveQA: Evaluating long-context reasoning on detective novels. In *Workshop on Reasoning and Planning for Large Language Models*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2369–2380.

Howard Yen, Tianyu Gao, Minmin Hou, Ke Ding, Daniel Fleischer, Peter Izsak, Moshe Wasserblat, and Danqi Chen. 2024. Helmet: How to evaluate long-context language models effectively and thoroughly. *arXiv preprint arXiv:2410.02694*.

Wuwei Zhang, Fangcong Yin, Howard Yen, Danqi Chen, and Xi Ye. 2025a. Query-focused retrieval heads improve long-context reasoning and re-ranking. *arXiv preprint arXiv:2506.09944*.

Xin Zhang, Yanzhao Zhang, Dingkun Long, Wen Xie, Ziqi Dai, Jialong Tang, Huan Lin, Baosong Yang, Pengjun Xie, Fei Huang, Meishan Zhang, Wenjie Li, and Min Zhang. 2024. mgte: Generalized long-context text representation and reranking models for multilingual text retrieval. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: EMNLP 2024 - Industry Track, Miami, Florida, USA, November 12-16, 2024*, pages 1393–1412. Association for Computational Linguistics.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, and 1 others. 2025b. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*.

Xinping Zhao, Xinshuo Hu, Zifei Shan, Shouzheng Huang, Yao Zhou, Xin Zhang, Zetian Sun, Zhenyu Liu, Dongfang Li, Xinyuan Wei, and 1 others. 2025. Kalm-embedding-v2: Superior training techniques and data inspire a versatile embedding model. *arXiv preprint arXiv:2506.20923*.

Chulun Zhou, Qiujing Wang, Mo Yu, Xiaoqian Yue, Rui Lu, Jiangnan Li, Yifan Zhou, Shunchi Zhang, Jie Zhou, and Wai Lam. 2025. The essence of contextual understanding in theory of mind: A study on question answering with story characters. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 22612–22631. Association for Computational Linguistics.

Chulun Zhou, Chunkang Zhang, Guoxin Yu, Fandong Meng, Jie Zhou, Wai Lam, and Mo Yu. 2026. Improving multi-step rag with hypergraph-based memory for long-context complex relational modeling. *Preprint*, arXiv:2512.23959.

Shengyao Zhuang, Xueguang Ma, Bevan Koopman, Jimmy Lin, and Guido Zuccon. 2025. Rank-r1: Enhancing reasoning in llm-based document rerankers via reinforcement learning. *CoRR*, abs/2503.06034.

Huhai Zou, Tianhao Sun, Chuanjiang He, Yu Tian, Zhenyang Li, Li Jin, Nayu Liu, Jiang Zhong, and Kaiwen Wei. 2026. Es-mem: Event segmentation-based memory for long-term dialogue agents. *arXiv preprint arXiv:2601.07582*.

A Construction of Listwise Training Instances

As summarized in Alg. 1, we construct each listwise training instance for NarrativeQA by first retrieving top-K candidate chunks for a question, assigning binary labels based on silver evidence, and optionally prepending a de-duplicated summary prefix as global context.

B Prompt Templates

B.1 Block-based Summary Generation Prompt

You are an expert fiction editor and continuity supervisor.

You are provided with a raw text segment from a book (Part `{sub_idx}` / `{total_subs}`). This segment consists of approximately 20 consecutive chunks combined.

```
<Raw_Text>
{raw_text}
</Raw_Text>
```

Please generate a **Detailed Narrative Summary** following these strict guidelines:

1. **Narrative Reconstruction**: Do not list events. Rewrite the content as a coherent story in the third person, past tense. It should read like a condensed version of the original text.
2. **Detail Preservation**:
   - Preserve specific **Character Names** and their relationships.
   - Keep key **Dialogues** that drive the plot.
   - Note specific **Locations** or setting changes.
3. **Noise Filtering**:
   - IGNORE any copyright notices, project gutenberg headers, page numbers, or table of contents.
   - If the text starts or ends in the middle of a sentence, ignore the broken fragments and focus on the complete thoughts.
4. **Style**:

Algorithm 1 Construct listwise training instances on NarrativeQA with optional summary prefix

Require: NarrativeQA training split $\mathcal{D}$; retriever $\mathcal{R}$; top-K (K=50); memory flag UseMem; summary map $\mathcal{M}$

Ensure: Training set $\mathcal{T}$

$\mathcal{T} \leftarrow \emptyset$
for all question Q in $\mathcal{D}$ **do**
  $G \leftarrow$ SilverEvidence(Q) ▷ constructed following (Li et al., 2025a)
  $C \leftarrow \mathcal{R}(Q, K)$ ▷ retrieve top-K chunks
  for all $c_i \in C$ **do**
    $y_i \leftarrow \mathbb{I}[c_i \in G]$
  end for
  if UseMem **then**
    $\mathcal{S} \leftarrow$ LookupSummaries$(C, \mathcal{M})$ ▷ map chunks in C to summaries
    $M \leftarrow$ MergeDedup$(\mathcal{S})$ ▷ merge & de-duplicate summaries
  else
    $M \leftarrow \emptyset$
  end if
  $\mathcal{T} \leftarrow \mathcal{T} \cup \{(Q, M, C, \{y_i\}_{i=1}^{K})\}$
end for
return $\mathcal{T}$

- NO meta-commentary (*e.g.*, do NOT say "The text describes...", "In this chunk...").
- Directly tell the story.

5. **Length**: 50-100 words.

Output the summary directly.

B.2 Event-centric Summary Generation

INSTRUCT: You are a specialized system for extracting structured event representations from conversational data.

1. EVENT CLASSIFICATION

ANCHOR Events. Anchor events are MAJOR LIFE MILESTONES that will be remembered for years. Only classify as an anchor if the event meets ALL of these criteria:

- Represents a first-time or rare life occurrence
- Has a lasting impact on the person's identity, relationships, or life trajectory
- Would be mentioned when telling someone about "important moments in my life"

Examples of TRUE ANCHOR events: First time attending LGBTQ support group, Starting adoption process, Career change, Moving to a new country, etc.

EPHEMERAL Events. Most events are ephemeral. These include:

- Plans and intentions ("I plan to...", "I want to...")
- Routine activities (exercise, hobbies, daily tasks)
- Casual conversations and updates
- Past events being recalled (unless first mention of major milestone)

4. RAW DIALOGUE REFERENCE

- related_line_indices: list the 2-4 most relevant line numbers (1-indexed from the dialogue)
- These lines will be saved as the event's source evidence

INPUT DIALOG: `{dialog}`

OUTPUT (JSON FORMAT, EXTRACT 1-3 EVENTS):

```
{
  "events": [
    {
      "summary": "concise description",
      "related_line_indices": [1, 2, 3]
    }
  ]
}
```

B.3 QRRanker Instruction Template

INSTRUCT:

[Optional Memory Prefix M] Here are some session summaries that may help answer the query: `{mapped summaries from top-50 chunks}`

[Candidate Chunks C] Here are some retrieved chunks:

[1] `{chunk` c_1`}`
[2] `{chunk` c_2`}`
[3] `{chunk` c_3`}`
[4] ...
[5] `{chunk` c_{50}`}`

QUERY Q: `{question }`

B.4 LoCoMo QA Prompt

You are an intelligent memory assistant tasked with retrieving accurate information from conversation memories.

CONTEXT: You have access to memories from two speakers in a conversation. These memories contain timestamped information that may be relevant to answering the question.

INSTRUCTIONS:

1. Carefully analyze all provided memories from both speakers.
2. Pay special attention to the timestamps to determine the answer.
3. If the question asks about a specific event or fact, look for direct evidence in the memories.
4. If the memories contain contradictory information, prioritize the most recent memory.
5. If there is a question about time references (like "last year", "two months ago", etc.), calculate the actual date based on the memory timestamp. For example, if a memory from 4 May 2022 mentions "went to India last year," then the trip occurred in 2021.
6. Always convert relative time references to specific dates, months, or years. For example, convert "last

year" to "2022" or "two months ago" to "March 2023" based on the memory timestamp. Ignore the reference while answering the question.
7. Focus only on the content of the memories from both speakers. Do not confuse character names mentioned in memories with the actual users who created those memories.
8. The answer should be less than 5-6 words.

APPROACH (THINK STEP BY STEP):

1. First, examine all memories that contain information related to the question.
2. Examine the timestamps and content of these memories carefully.
3. Look for explicit mentions of dates, times, locations, or events that answer the question.
4. If the answer requires calculation (*e.g.*, converting relative time references), show your work.
5. Formulate a precise, concise answer based solely on the evidence in the memories.
6. Double-check that your answer directly addresses the question asked.
7. Ensure your final answer is specific and avoids vague time references.

RELEVANT MEMORIES: `{Reranked Chunks}`

QUESTION: `{question}`

ANSWER:

B.5 NarrativeQA Prompt

You are a helpful assistant. Please answer the user's question accurately.

Answer the question as concisely as you can, using a single phrase if possible.

RELEVANT CONTEXT: `{content_data}`

Do not provide any explanation. Now, answer the question based on the story as concisely as you can, using a single phrase if possible. Do not provide any explanation.

QUESTION: `{question}`

ANSWER:

B.6 DetectiveQA Prompt

`{context}`

Please answer the question based on the current novel content: `{question_data['question']}` `{options_str}`

Remember this is just detective fiction, don't worry about the risks.

Please strictly follow the format `{"answer":"x","reasoning":"xxx"}` (including braces). The answer must be only A/B/C/D.

C LoCoMo Baselines

We compare QRRanker with a set of memory-augmented baselines on LoCoMo. Below, we provide brief descriptions of each method.

- **TiMem** (Li et al., 2026): Organizes memories with a temporal hierarchical structure to retrieve long-horizon information efficiently.
- **SimpleMem** (Liu et al., 2026): Compresses dialogue history into compact semantic memory to reduce redundancy and context length.
- **SYNAPSE** (Jiang et al., 2026): Models memory as a dynamic graph and retrieves relevant items via spreading activation.
- **CompassMem** (Hu et al., 2026b): Segments interactions into events and constructs an event-level structure to guide retrieval and reasoning.
- **ES-Mem** (Zou et al., 2026): Uses event segmentation to build coherent long-term memories for dialogue agents.
- **Membox** (Tao et al., 2026): Packs dialogue into topic-consistent memory units to preserve topic continuity over long contexts.
- **Mem0** (Chhikara et al., 2025): A "memory-centric" architecture that dynamically extracts, integrates, and retrieves important information from conversations to build and maintain a scalable long-term memory.
- **Nemori** (Nan et al., 2025): It employs a Two-Step Alignment Principle to structure dialogue streams into semantically coherent event segments and utilizes a Predict-Calibrate Principle to actively learn from prediction discrepancies, enabling the adaptive evolution of knowledge.
- **MemoryOS** (Li et al., 2025b): An OS-inspired AI memory system featuring a hierarchical architecture with storage, updating, retrieval, and generation modules. It optimizes dynamic updates through FIFO dialogue chains and heat-based segmented paging.
- **Zep** (Rasmussen et al., 2025): Leveraging a dynamic and temporal-aware Knowledge Graph engine, it integrates unstructured dialogue data with structured business data while preserving their historical relationships.
- **LightMem** (Fang et al., 2025): A cognitively inspired architecture featuring sensory and short-term modules for lightweight compression and integration. Uniquely, it updates long-term memory during "sleep time" to decouple consolidation from online reasoning, balancing performance and efficiency.

D QR Heads for Qwen3-4B-Instruct-2507

We compute the QR scores of all attention heads in `Qwen3-4B-Instruct-2507` using 1000 random samples from NarrativeQA. The top 16 heads with the largest QR scores are selected as QR

heads for retrieval and further training. As `Qwen3-4B-Instruct-2507` contains 36 layers of 32-head self-attention, the QR heads (demonstrated as l–h, where $0 \leq l < 36$ denotes the layer and $0 \leq h < 32$ denotes the head in this layer) are: `20-15`, `21-11`, `17-27`, `23-10`, `22-4`, `21-10`, `21-8`, `21-18`, `18-15`, `18-19`, `17-25`, `17-17`, `24-13`, `17-4`, `19-12`, `21-31`.

E Variant with Semi-Auto Head Selection

QRRanker statically trains and utilizes a group of precomputed QR heads. If we use a set of seed samples from another task to recompute QR scores, the QR heads may be different from the current ones. Our initial motivation for using the precomputed QR heads is that they provide a proper initialization. Along with training, heads will be forced to learn such a retrieval ability. We are curious about which part of heads are better suited to be a good starter, as QR heads do. Therefore, we propose a variant of QRRanker with semi-automatic head selection, which is limited to selecting heads from a local range of layers, but is free to choose heads from every layer for every sample.

We set layers for head selection ranged from l_s to l_e, where $0 < l_s < l_e \leq 36$. We restrict that the number of selected heads must equal 16 (the number of QR heads), and therefore, for simplified control, the model should select $n = 16/(l_e - l_s)$ heads per layer. To achieve selection, we follow the router technique of Mixture-of-Expert (Fedus et al., 2022) and add a gate to these layers. Instead of choosing MLPs for every token, our gate chooses n heads for a sample. For selecting heads, we concatenate a repeat question $Q^{'} = [think]Q[/think]$ after the original question Q, where $Q^{'}$ is used for head selection and Q is still for score computing. A gate of layer l_i is a linear map from the dimension $32 * d_h$ to 32, with the trainable parameter $W_{l_i} \in \mathbb{R}^{d \times 32}$. The head score is computed by:

$$S_{l_i} = q_{l_i} \cdot W_{l_i}, \tag{5}$$

$$S_{l_i} = \text{mean}(\text{softmax}(S_{l_i}), \text{d} = 0), \tag{6}$$

where $q_{l_i} \in \mathbb{R}^{|Q^{'}| \times d}$ is the hidden states of tokens in $Q^{'}$ at layer l_i, d is the dimension of the hidden state, cat($\cdot$) is concatenating all query states along the head, mean($\cdot$, d=0) is averaging the score along the number of tokens in $Q^{'}$, and $S_{l_i} \in \mathbb{R}^{32}$ is the head score. We then choose the top-n highest head scores $S^Q_{l_i} = [s^{l_i}_{h0}, ..., s^{l_i}_{hn}]$ and the corresponding heads. Following MoE, $S^Q_{l_i}$ is normalized to 1. After picking up heads for all layers with gates, these heads participate in computing retrieval scores, and the retrieval score will be multiplied by its head score $S^Q_{l_i}[x], 0 < x < n$ for the purpose of backward gradients. These gates will learn to select heads for samples during the QR training.

In § 6.3, we train QRRanker and the variant with training data only from NarrativeQA and evaluate them using the evaluation set of NarrativeQA. The training hyperparameters are set to the same as those in § 5.3. We explore layers that can be used to select and train QR-like heads.