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.*, 3B 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.*, 3B 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 across diverse retrieval settings, including long narrative QA (NarrativeQA, DetectiveQA), long-term dialogue memory (LoCoMo), wikipedia multi-hop QA (MuSiQue, HotpotQA), and reasoning-intensive retrieval (BRIGHT), demonstrate the effectiveness of QRRanker. As a versatile reranking framework, QRRanker outperforms strong general-purpose pointwise and listwise rerankers, such as Qwen-Reranker and ReasonRank, and also improves over domain-specific retrieval or memory methods, including HippoRAG-v2 (Guti'errez et al., 2025) for Wikipedia QA and recent memory-enhanced approaches (Li et al., 2025b; Rasmussen et al., 2025) on LoCoMo. Additional comparisons with retrieval agents (Li et al., 2026b; NVIDIA, 2026) further suggest that QRRanker can serve as an efficient one-step alternative to iterative retrieval pipelines.

2 Related Work

Reranking Reranking methods are commonly built on bi-encoder or cross-encoder architectures (Koch et al., 2015; Thakur et al., 2021). Bi-encoders retrieve candidates using reusable document embeddings (Zhang et al., 2025b), but suffer from the "geometric bottleneck" and cannot fully model fine-grained query-document interactions. Cross-encoders address this by jointly encoding each query-document pair with cross-attention, but their high cost limits them to reranking the top-n candidates retrieved by bi-encoders. LLM-based rerankers are typically categorized into pointwise and listwise methods. Pointwise methods score documents independently and are widely used in practice (Qin et al., 2024; Sun et al., 2023; Liu et al., 2025a; Zhuang et al., 2025), *e.g.*, Qwen3 (Zhang et al., 2025b), Jina, mGTE (Zhang et al., 2024), and BGE-m3 (Chen et al., 2024); however, independent scoring limits their ability to capture global information across the candidate list. Listwise methods concatenate multiple documents and generate rankings directly (Pradeep et al., 2023a,b), with recent variants using reasoning or RL for stronger performance (Sun et al., 2023; Liu et al., 2025a; Qin et al., 2025; Sun et al., 2025). However, they often require ranking-specific supervision and suffer from unstable generation formats, especially when reasoning traces are introduced.

Recent studies show that LLMs inherently possess retrieval ability, and retrieval-related attention heads can be extracted or modulated for retrieval (Wu et al., 2024; Zhang et al., 2025a; Lee et al., 2025). SEAL (Lee et al., 2025) learns lightweight scaling factors for attention heads or channels to enhance long-context retrieval. In contrast, QRRanker trains selected query-focused retrieval heads as an explicit listwise reranking scorer, producing fine-grained document scores with query-positive-negative supervision.

Memory Utilization Memory construction and utilization have been widely studied to alleviate long-context processing challenges. Existing work builds global memories for long-story understanding (Zhou et al., 2025; Kočiskỳ et al., 2018; Xu et al., 2025b; Wu et al., 2025; Li et al., 2025a), and designs 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., 2026a), or memory systems (Chhikara et al., 2025; Li et al., 2025b; Nan et al., 2025; Tao et al., 2026; Zou et al., 2026) to retrieve relevant dialogue histories, events, personas, and chunks. In contrast to increasingly complex memory management, we show that stronger retrieval over simply constructed memories can be an effective alternative.

3 Preliminaries: QR-head

We first introduce Query-Focused Retrieval heads (QR-heads). Prior studies show that some self-

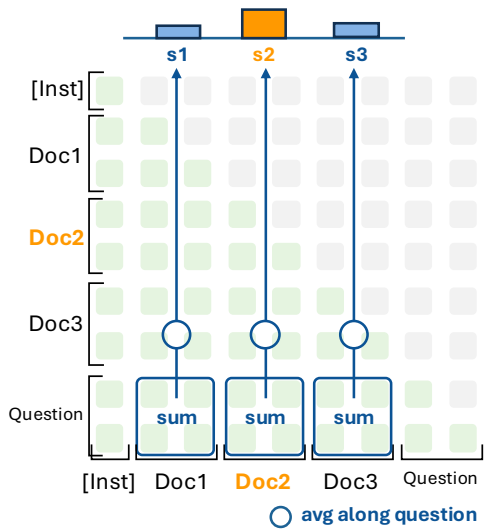

Figure 1: The retrieval score and QR score are computed based on the attention score of a (QR) attention head. In this figure, Doc2 is the gold document (chunk).

attention heads act as retrievers by attending to context spans relevant to the question during question encoding (Wu et al., 2024; Zhang et al., 2025a). Zhang et al. (2025a) define such heads as QR-heads and identify them using the QR score.

Formally, given a question Q and its context $C = [c_0, c_1, \ldots, c_n]$, let $G = [c_{g0}, \ldots, c_{gm}]$ denote the gold chunks. For an attention head h, we denote its attention from Q to chunk c_i as $A_h^{Q \to c_i} \in \mathbb{R}^{|Q| \times |c_i|}$. The QR score of h is computed by aggregating its attention to gold chunks:

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

where w_q and w_c denote tokens in Q and c_i, respectively. A higher QR score indicates stronger attention to gold chunks. Following Zhang et al. (2025a), we compute the average QR score of each head on a seed set, rank all heads, and select the top 16 as QR-heads H_{QR}. In this work, we select QR-heads for `Qwen3-4B-Instruct-2507` (Yang et al., 2025) using 1,000 random samples from NarrativeQA training split. We also apply the same selection procedure to `Llama-3.2-3B-Instruct` (Grattafiori et al., 2024) on 1,000 random samples from MuSiQue in Appx. G, showing that the selected heads are effective beyond a single model or data.

Given the selected QR-heads, the score of a chunk c_i is computed by aggregating its question-to-chunk attention over H_{QR}, analogous to Eq. 1.

4 Method

QRRanker is a listwise reranking method that processes all candidate documents in a single inference pass. Unlike generation-based listwise rerankers, it only prefills the question-document prompt and derives document scores from attention patterns, avoiding generation-format errors and reducing inference cost. Since precomputed QR-heads may vary across tasks (Zhang et al., 2025a), we further train them with a dedicated pipeline: constructing listwise training instances and optimizing the selected 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.

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 Appx. 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

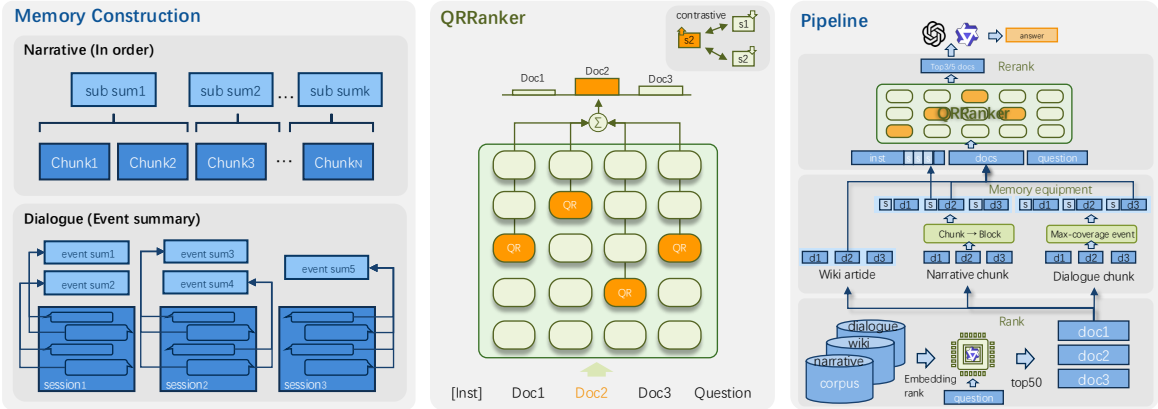

Figure 2: The structure of QRRanker is illustrated in the middle, where the highlighted heads are QR heads for document scoring. As QRRanker can be aware of memory enhancement to capture more contextual information, we can construct memories for narratives and dialogues, which is shown on the left. The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

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 Appx. J.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. (Details in Appx. J.2).

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 Appx. J.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:

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

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 adopt a group version of contrastive loss to simultaneously optimize them:

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

where τ denotes the exponential function. The objective above treats each 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 $\mathtt{P} = \text{Inst}(M, C, Q)$.

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 (Sun et al., 2025), ReasonRank (Liu et al., 2025a), 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., 2026a), Synapse (Jiang et al., 2026), Membox (Tao et al., 2026), CompassMem (Hu et al., 2026b), and ES-Mem (Zou et al., 2026); SimpleMem (Liu et al., 2026). Detailed baseline descriptions are provided in Appx. E.

Additional Comparisons. Beyond the main benchmarks, we include two additional comparisons to examine QRRanker's scope. First, we evaluate QRRanker on BRIGHT (SU et al., 2025), a reasoning-intensive retrieval benchmark, and compare it with reasoning rerankers: ReasonRank and GroupRank. Second, we compare one-step QRRanker reranking with recent retrieval agents, DCI-Agent (Li et al., 2026b) and Nemo Retriever Agent (NVIDIA, 2026), to contrast lightweight reranking with iterative agentic retrieval. Details are provided in Appx C and D.

5.3 Implementation Details

QRRanker is trained on `Qwen3-4B-Instruct-2507`. Unless otherwise stated, all results are based on this backbone. We fine-tune the full backbone using a contrastive ranking loss computed from the attention scores of selected QR heads. To verify the generalizability of head selection and cross-architecture transferability, we additionally experiment with `Llama-3.2-3B Instruct`. QR head selection details for both models are provided in Appx. F. During training, the $scale$ factor in the max-min norm is set to 8, the batch size is 1, the gradient accumulation step is 4, and the learning rate is 1×10^{-5}. We adopt the DeepSpeed ZeRO Stage 2 strategy and use 8 NVIDIA H20 GPUs.

For downstream QA evaluation, we use task-specific prompting for generation; the full prompt templates for NarrativeQA, DetectiveQA, and Lo-

Methods	Wikipedia QA						Story QA						Overall		
	Musique			HotpotQA			NarrativeQA			DetectiveQA			Avg@k		
	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
ReasonRank-32B	57.33	64.58	72.26	92.90	95.65	96.85	28.58	37.11	50.71	**36.00**	**42.84**	51.83	53.70	60.05	67.91
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}. The first-stage retriever R@50 ceiling is 86.17% (Musique), 97.85% (HotpotQA), 79.90% (NarrativeQA), and 63.85% (DetectiveQA). '–' 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. Bold and underline denote the best and second-best results, respectively. * 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
Qwen-Reranker-4B (out-of-box)	76.15	83.02	90.15
Qwen-Reranker-4B (trained)	79.17	85.51	90.74
GroupRank-32B	77.99	82.94	88.14
ReasonRank-32B	82.49	86.83	92.45
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. The first-stage retriever R@50 ceiling is 97.87%

CoMo are provided in Appx. J. 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. In our main evaluation, QRRanker is built upon the `Qwen3-4B-Instruct-2507` backbone, while we also provide results for a `Llama-3.2-Instruct` variant in Appx. G to demonstrate that QRRanker is robust across different backbones. 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 effectiveness of QRRanker when applied to rerank the candidates retrieved by the first-stage retriever.

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). The performance gap is particularly evident in the story domain, where context tracking is critical. Under our long-context reranking evaluation, QRRanker achieves competitive or stronger average recall than much larger 32B rerankers, such as GroupRank (Sun et al., 2025) and ReasonRank (Liu et al., 2025a). We further compare QRRanker with these listwise rerankers on BRIGHT in Appx. C, demonstrating its effec-

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., 2026a)	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.

tiveness in reasoning-intensive retrieval. Furthermore, QRRanker maintains this advantage on LoCoMo, despite using a substantially smaller backbone. These results suggest that QR-head training is an effective lightweight alternative for long-context reranking.

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, complementing 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 Heads from Different Layer- Levels

QRRanker uses a fixed set of precomputed QR-heads, raising the question of which layer ranges provide suitable initialization for QR training. To study this, we use a semi-automatic variant that selects 16 heads from a continuous layer range l_s–l_e for each sample. We evaluate QRRanker and its variants on NarrativeQA, with implementation details provided in Appx. I.

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 Table 6, selecting heads from lower layers (10–17) leads to a clear performance drop, while selecting from middle layers (17–24) or higher layers (28–35) achieves performance close to QRRanker. This suggests that QR training is more effective when initialized from middle-to-high layers, where retrieval-relevant behavior is more likely to emerge. Since the 17–24 variant uses only middle-layer heads while preserving comparable performance, it also motivates a truncated middle-layer variant for more efficient inference.

Interestingly, the heads selected by the 17–24 variant have low overlap with the original QR-heads. This indicates that QR training can activate retrieval ability in diverse mid-layer heads, rather than relying exclusively on the originally discovered QR-heads. Nevertheless, QRRanker still performs slightly better, suggesting that precomputed QR-heads provide a stronger initialization prior. Further head-level comparisons are provided in Appx. H. We report the inference efficiency of the truncated middle-layer variant in § 6.4.

6.4 Inference Efficiency

We compare inference efficiency on 20 random samples. Table 7 shows that `QRRanker` reduces P50/P95 latency, TFLOPs, and peak memory compared with `Qwen3-Reranker-4B`. `QRRanker(middle)` further truncates the model after layer 24, achieving the lowest latency and resource cost. For `Qwen3-Reranker-4B`, we report both *batch=50*, which processes all chunk–query pairs in one forward pass, and *batch=1*, which processes them separately. These results show that QRRanker provides a more efficient reranking interface, especially with middle-layer truncation.

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, compute, and peak GPU memory. All models are evaluated under the same hardware and inference settings. `QRRanker(middle)` truncates the model after layer 24.

6.5 Further Analysis

We provide further analyses to better understand QRRanker's behavior and generality. QRRanker remains robust when the same top-50 candidate set is randomly shuffled on HotpotQA and LoCoMo, suggesting that its scores primarily reflect content-level relevance rather than positional cues (Appx. B). Moreover, we compare QRRanker with advanced retrieval agents, showing that it generalizes beyond standard reranking settings while retaining a simple one-step retrieval–reranking pipeline (Appx. D).

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 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, although we validate QRRanker on two backbone architectures , its behavior on larger-scale models (e.g., 14B+) remains unexplored. Moreover, larger models with more attention heads per layer could offer a richer pool of retrieval-sensitive candidates, potentially improving both head selection quality and final reranking performance. We leave this exploration for future work. 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.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

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.

Changhun Lee, Minsang Seok, Jun-gyu Jin, YoungHyun Cho, and Eunhyeok Park. 2025. Seal: Scaling to emphasize attention for long-context retrieval. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 28942–28955.

Kai Li, Xuanqing Yu, Ziyi Ni, Yi Zeng, Yao Xu, Zheqing Zhang, Xin Li, Jitao Sang, Xiaogang Duan, Xuelei Wang, and 1 others. 2026a. 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*.

Zhuofeng Li, Haoxiang Zhang, Cong Wei, Pan Lu, Ping Nie, Yi Lu, Yuyang Bai, Shangbin Feng, Hangxiao Zhu, Ming Zhong, Yuyu Zhang, Jianwen Xie, Yejin Choi, James Zou, Jiawei Han, Wenhu Chen, Jimmy Lin, Dongfu Jiang, and Yu Zhang. 2026b. Beyond semantic similarity: Rethinking retrieval for agentic search via direct corpus interaction. *arXiv preprint arXiv:2605.05242*.

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.

Meixiu Long, Duolin Sun, Dan Yang, Junjie Wang, Yue Shen, Jian Wang, Peng Wei, Jinjie Gu, and Jiahai Wang. 2025. Diver: A multi-stage approach for reasoning-intensive information retrieval. *Preprint*, arXiv:2508.07995.

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*.

NVIDIA. 2026. NeMo Retriever: Agentic Retrieval Pipeline for BRIGHT. GitHub repository.

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*.

Hongjin SU, Howard Yen, Mengzhou Xia, Weijia Shi, Niklas Muennighoff, Han yu Wang, Liu Haisu, Quan Shi, Zachary S Siegel, Michael Tang, Ruoxi Sun, Jinsung Yoon, Sercan O Arik, Danqi Chen, and Tao Yu. 2025. BRIGHT: A realistic and challenging benchmark for reasoning-intensive retrieval. In *The Thirteenth International Conference on Learning Representations*.

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.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

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.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*.

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*.

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 $\mathcal{T}$

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

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 Position Bias Analysis

Since QRRanker concatenates all 50 candidates under causal attention, a natural question is whether it exploits positional cues rather than semantic relevance. To test this, we keep the same top-50 candidate set for each query and evaluate three conditions: (i) the original retriever order preserved, (ii) a randomly shuffled order used directly, and (iii) QRRanker reranking on both the original and shuffled inputs.

As shown in Table 9, the random-order baseline collapses to near-chance levels after shuffling, indicating that positional information is largely destroyed. In contrast, QRRanker remains robust under shuffled input, with a maximum drop of only 3.40 R@3 on HotpotQA. Notably, even after shuffling, QRRanker still substantially outperforms the original-order retriever, demonstrating that its ranking quality stems from content-level relevance matching rather than positional cues. We further assess ranking stability by running five independent random permutations on LoCoMo. The resulting Kendall's $\tau = 0.638 \pm 0.100$ and Spearman's $\rho = 0.808 \pm 0.096$ indicate high agreement across permutations, suggesting that QRRanker produces consistent rankings largely independent of order.

C Comparison on BRIGHT

Reasoning rerankers like GroupRank (Sun et al., 2025) and ReasonRank (Liu et al., 2025a) focus on reasoning-intensive retrieval. They are trained by a sophisticated pipeline with supervised fine-tuning and reinforcement learning to possess the think-then-rank ability. To verify QRRanker's capability of handling reasoning-intensive retrieval scenarios and compare it with these rerankers excelling in reasoning, we trained QRRanker using the open-source training set of ReasonRank (Liu et al., 2025a)[1]. The training process involves no reasoning CoT generation but only uses the positive/negative annotations for our contrastive learning.

We compare QRRanker with ReasonRank and GroupRank on the reasoning dataset BRIGHT (SU et al., 2025). Following GroupRank's setting, QRRanker also uses the original queries to rerank the top-100 passages retrieved by DIVER-Retriever-4B (Long et al., 2025) using GPT4-rewritten queries. The results are shown in Table 8. QRRanker, a 3B model, is trained with the same data as ReasonRank and produces better results than ReasonRank's 32B version on average, indicating its inner reasoning ability may be activated by our QR training without complicated thinking generation. As for GroupRank, the model is trained with closed-source and more diverse data constructed by the authors. Additionally, GroupRank's generation is performed several times to get the self-consistency results, where self-consistency is proven to improve CoT reasoning by Wang et al. (2023). QRRanker can defeat GroupRank 7B and show comparable performance against GroupRank 32B, which demonstrates QRRanker's potential in reasoning-intensive retrieval with easy training adaptation.

[1] https://huggingface.co/datasets/liuwenhan/reasonrank_data_13k

Model	Avg.	Stackoverflow							Coding		Theorem-based		
		Bio.	Earth.	Econ.	Psy.	Rob.	Stack.	Sus.	Leet.	Pony	AoPS	TheoQ.	TheoT.
ReasonRank 7B†	32.5	51.6	43.4	32.4	44.0	31.0	25.6	39.8	15.4	20.1	7.0	38.9	40.7
ReasonRank 32B†	35.6	53.9	47.6	36.3	**52.6**	36.5	34.2	**44.5**	15.2	14.8	5.5	40.6	45.3
GroupRank 7B†	34.3	52.7	51.0	33.8	44.5	32.1	33.9	38.1	16.3	17.3	8.7	**40.7**	42.4
GroupRank 32B†	**38.0**	59.0	57.5	**39.2**	50.0	**39.1**	**39.0**	42.7	14.3	14.9	**12.6**	39.0	**48.8**
QRRanker (3B)	36.2	**60.7**	**60.0**	32.3	47.1	34.2	37.8	35.6	**30.2**	19.4	8.3	34.5	34.0

Table 8: nDCG@10 results of QRRanker and reasoning rerankers on BRIGHT. † denotes the results are cited from Sun et al. (2025). QRRanker ranks in second place on average compared with reasoning rerankers.

Dataset	Setting	R@3	R@5	R@10
HotpotQA	Retriever (original)	82.85	89.05	95.15
	Random (shuffled)	7.05	11.70	20.10
	QRRanker (original)	95.05	96.90	97.70
	QRRanker (shuffled)	91.65	95.20	97.00
LoCoMo	Retriever (original)	76.01	83.10	90.15
	Random (shuffled)	6.04	9.99	18.96
	QRRanker (original)	87.34	91.32	95.01
	QRRanker (shuffled)	85.35	90.60	93.78

Table 9: Position bias analysis. "Random (shuffled)" denotes using a randomly permuted order directly as the ranking output without any reranking.

D One-step QRRanker Reranking v.s. Advanced Retrieval Agents

We also compare QRRanker with the recent advanced retrieval agents: DCI-Agent (Li et al., 2026b) and Nemo Retrieval Agent (NVIDIA, 2026). DCI-Agent discards any similarity-based retrieval tools, and iteratively utilizes only `grep`/shell commands/read tools to access texts in chunks. Constructed for QA, DCI-Agent also achieves excellent performance in retrieval tasks (*e.g.,* BRIGHT and BEIR subsets) (Li et al., 2026b). To this end, we extend DCI-Agent to StoryQA and WikipediaQA using its settings on BRIGHT. As for Nemo Agent, it is a typical embedding-retrieval-based ReAct (Yao et al., 2023) agent that iteratively recalls top-N chunks using embedding tools, and finally picks and reranks the required number of chunks among all retrieved chunks. Unlike these agents, we adapt QRRanker to a standard one-step pipeline: embedding retrieval $\rightarrow$ rerank, which can be a step of tool calling.

Following the settings of DCI-Agent, the LLM used by agents is GPT-5.4-nano with highly thinking; the context management of DCI-Agent, dealing with the over-length situation, shortens the large tool results in each turn, keeps recent turns, and replaces older tool results with placeholders (called Level3 by Li et al. (2026b)). Nemo Agent uses a different context management: no truncation; when retrieved chunks explode the context, it ends the main agent and uses a selection agent to do the final reranking. To ensure fairness, we use Qwen3-Embedding-8B as the embedding model for Nemo Agent and our method on Story/Wikipedia/Conversation QA, Diver-Retriever-4B on BRIGHT. As agents are notoriously slow and expense-unfriendly, we randomly select 100 samples for Story/Wikipedia/Conversation QA, and BRIGHT aligns with DCI-Agent settings.

The results are shown in Table 10, where Nemo Agent exhibits strong performance due to its harness engineering to fully utilize LLM and frequent retrieval. It is worth noting that QRRanker is competitive against retrieval agents as it achieves the best results on BRIGHT and StoryQA on average without any LLM interactive synergy. As for WikipediaQA, which is thoroughly studied with massive high-quality training resources, retrieval agents like Nemo Agent can significantly reason and trigger more retrieval-tool calls, which can be seen in Table 11. Therefore, they almost kill this type of task, especially indicated by the results on HotpotQA. As for Nemo Agent defeating QRRanker on LoCoMo, Nemo Agent recalls and reranks all chunks in LoCoMo, showing in Table 11. Referring to tasks like StoryQA and BRIGHT, which require integrative reasoning over dispersed evidence rather than compositional entity-hop resolution, Nemo Agent basically falls back to the same level as QRRanker. It indicates that current retrieval agents may need to enhance their capacity for holistic evidence synthesis. In the future, we will try to synergize the agent with our QRRanker to reach such a goal.

DCI-Agent shows a notable performance disparity across task types. On WikipediaQA, it achieves strong retrieval quality while introducing remarkably few chunks into the LLM context, largely be-

	Recall@3/5/10					nDCG@10				
	StoryQA		**WikipediaQA**		**Conv.**	**BRIGHT**				
Model	NarQA	DetQA	MuSiQue	HotpotQA	LoCoMo	Avg.	Bio.	Earth.	Econ.	Rob.
DCI-Agent	12.7/15.6/19.3	22.0/28.0/30.0	63.6/68.3/75.3	97.0/98.5/98.5	86.3/88.6/91.4	46.4	60.0*	50.8*	32.3*	**42.4***
Nemo Agent	27.3/**36.5/47.4**	23.0/35.0/42.0	**76.9/86.7/92.1**	**98.0/99.5/99.5**	**87.5/90.5/96.5**	45.7	55.7	51.5	**35.6**	39.8
QRRanker	**28.0**/36.4/47.2	**25.0/39.0/49.0**	67.2/75.1/78.8	93.5/95.5/97.0	86.8/90.2/96.3	**46.8**	**60.7**	**60.0**	32.3	34.2

Table 10: Performance comparison across dataset groups. StoryQA includes NarrativeQA and DetectiveQA (averaged over EN and ZH); WikipediaQA includes MuSiQue and HotpotQA; Conv. denotes LoCoMo; BRIGHT includes Biology, Earth Science, Economics, and Robotics subsets according to Li et al. (2026b). * denotes the results are cited from Li et al. (2026b).

Dataset	**#C.(T)**	**Method**	**#C.(R)**	**#Ret.**	**#Ti.(s)**	**#Tok.**
Story	831	Nemo	242	12	97	280k
		DCI	111	16	41.7	57k
		Ours	50	1	1.4	10k
LoCoMo	71	Nemo	71	13	89	205k
		DCI	37	16	32.7	27k
		Ours	50	1	1.8	13k
Wiki	11k	Nemo	401	19	115	602k
		DCI	7	15	34.6	55k
		Ours	50	1	1.1	8k
BRIGHT	73k	Nemo	239	12	61	168k
		DCI	–	–	–	–
		Ours	100	1	1.5	27k

Table 11: Average resource consumption per sample across dataset groups. All values prefixed with # denote averages. **C.(T)**: total chunks in the corpus; **C.(R)**: number of chunks retrieved; **Ret.**: number of times the retrieve function (or tool) is called; **Ti.(s)**: latency for finishing a sample in seconds; **Tok.**: total tokens consumed per sample. Because we cite the BRIGHT results of DCI-Agent from Li et al. (2026b), we do not display its consumption on BRIGHT. In the table, we can see that Nemo Agent significantly invokes more retrieval calls in Wikipedia than in others. Due to `grep` tools reading localized and partial contents in a chunk, DCI-Agent refers to dozens of chunks but consumes significantly fewer tokens than Nemo.

cause explicit entity names in the questions can often be matched precisely with `grep`. However, approximately 20% of WikipediaQA queries issue `grep` commands containing terms from the *gold answer* that are absent from the question, suggesting that the agent may leverage the LLM's parametric knowledge of Wikipedia content to guide retrieval. In contrast, on StoryQA, where evidence is distributed across long narratives and deeper comprehension is required, DCI-Agent's keyword-matching strategy degrades substantially, and it underperforms QRRanker and Nemo Agent with conventional retrieval tools. This phenomenon aligns with the findings of the DCI-Agent paper (Li et al., 2026b): `grep` verifies exact and localized spans in chunks instead of the whole chunk, leading to a lower recall but deeper exploitation of fine-grained discoveries (Li et al., 2026b). This utilization makes DCI-Agent a great question answerer. In future work, we will explore combining QRRanker with local search techniques to support deeper and fine-grained evidence mining beyond expanding retrieval coverage.

Overall, QRRanker is competitive against retrieval agents, which does not imply QRRanker should replace agents. Rather, we believe that QRRanker can be a good enhancer, acting as a tool equipped for these agents, to reach a new level of retrieval coverage and deeper evidence digging. We leave this to future work, as we have mentioned above.

E 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., 2026a): 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.

F QR Head Selection

Qwen3-4B-Instruct-2507. This model contains 36 layers of 32-head self-attention. We use 1000 random samples from the NarrativeQA training split as the seed set to compute QR scores and select the top-16 heads. The selected QR heads (denoted as l-h, where $0 \leq l < 36$ is the layer and $0 \leq h < 32$ is the head index) 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`.

Llama-3.2-3B-Instruct. To verify that QR head selection generalizes across model architectures and seed datasets, we apply the same procedure to `Llama-3.2-3B-Instruct` using 1000 random samples from the MuSiQue training split as the seed set. The selected QR heads are: `12-1`, `13-9`, `13-23`, `12-16`, `8-8`, `13-16`, `12-6`, `13-22`, `12-2`, `15-22`, `12-3`, `15-18`, `12-10`, `14-2`, `21-20`, `9-23`.

Dataset / Method	R@3	R@5	R@10
HotpotQA (out-of-domain)			
Retriever	82.85	89.05	95.15
QRHeads	89.85	93.75	96.40
QRRanker	**93.05**	**96.20**	**97.60**
LoCoMo (out-of-domain)			
Retriever	58.61	67.57	79.15
QRHeads	82.75	87.84	92.52
QRRanker	**84.27**	**88.98**	**93.65**
DetectiveQA (out-of-domain, avg EN/ZH)			
Retriever	12.84	20.03	31.17
QRHeads	20.03	27.55	39.07
QRRanker	**22.04**	**28.88**	**40.65**

Table 12: Out-of-domain results with Llama-3.2-3B-Instruct. The first-stage retriever is Qwen3-Embedding-8B. DetectiveQA scores are averaged over English and Chinese sets.

G Cross-Architecture Results with Llama-3.2-3B-Instruct

To verify cross-architecture generalization, we instantiate QRRanker on Llama-3.2-3B-Instruct. Table 12 shows that this variant consistently outperforms both the base retriever and pretrained QRHeads across out-of-domain dataset: HotpotQA, LoCoMo, and DetectiveQA, demonstrating effective transfer of the QR training objective to different model families. The results indicate that QRRanker transfers well across model architectures. Compared with the first-stage retriever, the Llama-3.2-3B-Instruct variant brings consistent gains on all three out-of-domain datasets, especially on LoCoMo and DetectiveQA where the initial retrieval quality is relatively weaker and reranking provides larger improvements. QRRanker also consistently surpasses QRHeads at all recall cutoffs. These results suggest that the QR training objective is not tied to a specific backbone architecture, and can effectively improve both multi-hop retrieval and long-context narrative retrieval under out-of-domain transfer.

H Validation of QR Head Selection

In § 6.3, we show that heads from middle-to-top layers can reach comparable performance after QR training using the semi-auto selection variant. This raises a natural question: does the QR head discovery itself matter? Table 13 addresses this by comparing different head selection strategies on HotpotQA using the pretrained Qwen3-4B model, without any QR training.

Method	R@1	R@3	R@5
Retriever (Qwen3-Emb-8B)	45.45	82.85	89.05
Pretrained (no QR training)			
Random mid-layer (seed 42)	43.45	84.85	92.55
Random mid-layer (seed 123)	41.70	84.10	93.35
Random mid-layer (seed 999)	40.75	81.15	91.60
Discovered QR Heads	**45.50**	**90.20**	**94.80**
After QR training			
QRRanker	**48.35**	**95.05**	**96.90**

Table 13: Head selection validation on HotpotQA (Qwen3-4B). Random mid-layer heads are sampled from layers 17–24.

Without any QR training, the discovered QR heads already outperform random mid-layer heads by +6.83 R@3 on average, indicating that the QR score selects heads with inherently stronger retrieval behavior. After QR training, performance further increases to 95.05 R@3, highlighting the complementary benefits of informed head selection and subsequent training.

I 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:

$$S_{l_i} = q_{l_i} \cdot W_{l_i}, \quad (5)$$

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

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(·) is concatenating all query states along the head, mean(·, 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.

J Prompt Templates

J.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**:
   - 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.

J.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]
    }
  ]
}
```

J.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 }`

J.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:

J.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:**

J.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.