Query-Focused Retrieval Heads Improve Long-Context Reasoning and Re-ranking

Wuwei Zhang[♠] **Fangcong Yin**[♢] **Howard Yen**[♠] **Danqi Chen**[♠] **Xi Ye**[♠]
[♠] Princeton Language and Intelligence, Princeton University
[♢] The University of Texas at Austin
[♠] {wuwei.zhang,hyen,danqic}@cs.princeton.edu xi.ye@princeton.edu
[♢] fangcongyin@utexas.edu

Abstract

Recent work has identified retrieval heads (Wu et al., 2025b), a subset of attention heads responsible for retrieving salient information in long-context language models (LMs), as measured by their copy-paste behavior in Needle-in-a-Haystack tasks. In this paper, we introduce QRHEAD (Query-Focused Retrieval Head), an improved set of attention heads that enhance retrieval from long context. We identify QRHEAD by aggregating attention scores *with respect to the input query*, using a handful of examples from real-world tasks (e.g., long-context QA). We further introduce QRRETRIEVER, an efficient and effective retriever that uses the accumulated attention mass of QRHEAD as retrieval scores. We use QRRETRIEVER for long-context reasoning by selecting the most relevant parts with the highest retrieval scores. On multi-hop reasoning tasks LongMemEval and CLIPPER, this yields over 10% performance gains over full context and outperforms strong dense retrievers. We also evaluate QRRETRIEVER as a re-ranker on the BEIR benchmark and find that it achieves strong zero-shot performance, outperforming other LLM-based re-rankers such as RankGPT. Further analysis shows that both the query-context attention scoring and task selection are crucial for identifying QRHEAD with strong downstream utility. Overall, our work contributes a general-purpose retriever and offers interpretability insights into the long-context capabilities of LMs.[1]

1 Introduction

Retrieving salient information from long context serves as a foundation for language models (LMs), enabling a wide range of downstream applications, such as long document understanding and passage re-ranking. Prior work has identified a subset of attention heads in transformers (Vaswani et al., 2017) that are responsible for retrieving relevant information, known as *retrieval heads* (Wu et al., 2025b).

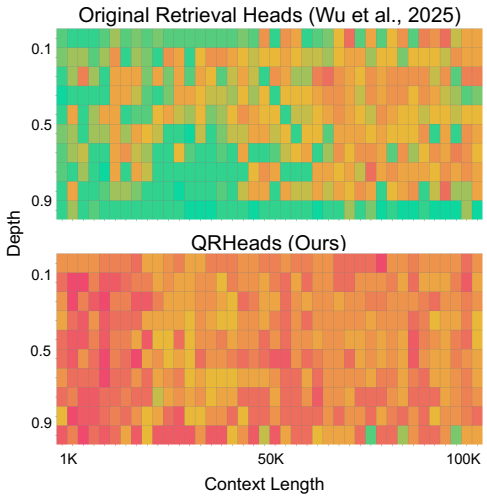

Figure 1: **Top**: Masking the top 32 original retrieval heads (Wu et al., 2025b) of Llama-3.1-8B. **Bottom**: Masking the top 32 QRHeads of the same model, which has a more pronounced impact on Needle-in-a-Haystack.

However, these retrieval heads are identified based on the frequency of their copy-paste operations in a simple synthetic task—Needle-in-a-Haystack (NIAH; Kamradt, 2024). Although they exhibit significance on certain downstream tasks, such as extractive question answering, we argue that the copy-paste objective and synthetic data used to identify them are misaligned with how language models retrieve pertinent information in real-world settings.

To this end, we propose a more effective approach for identifying retrieval heads and introduce QRHEAD, a distinct subset of attention heads whose attention mass plays a more critical role in retrieving relevant information from long context. Compared to original retrieval heads, our method incorporates two key changes: (1) a query-context scoring function that measures attention mass allocated to pertinent context spans with respect to an

[1] Code: https://github.com/princeton-pli/QRHead.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 23791–23805
November 4-9, 2025

input query, and (2) the use of more natural data from real-world tasks, such as question answering over long texts. Our method only requires a small amount of data to be effective. As shown in Figure 1, we detect QRHEAD using 70 examples from a natural long-context QA task, LongMemEval, and find masking them out results in more severe degradation in NIAH compared to original retrieval heads detected from in-domain data.

Furthermore, we build QRRETRIEVER on top of QRHEAD as a general-purpose retriever for improving LMs on diverse long-context downstream applications. Given a query and a set of passages (e.g., a claim and a book consisting of multiple chapters), QRRETRIEVER scores each passage using the aggregated attention mass from the QRHEAD of a language model, and returns the top-ranked passages. We detect QRHEAD for multiple LMs of different scales (3B–70B) and families (Llama-3.1, Llama-3.2, and Qwen), and build QRRETRIEVER with these LLMs.

We evaluate QRRETRIEVER on two long-context, multi-hop reasoning tasks: LongMemEval (Wu et al., 2025a) and CLIPPER (Pham et al., 2025). Using QRRETRIEVER to select top-ranked documents yields substantial improvements in retrieval recall and downstream task performance. For example, with Llama-3.1-8B-Instruct, QRRETRIEVER outperforms dense retrievers and improves performance by over 10% on both datasets, compared to full-context generation. We further evaluate QRRETRIEVER as a re-ranker on the standard BEIR benchmark (Thakur et al., 2021). It exhibits strong zero-shot performance across diverse domains and outperforms other LLM-based re-rankers, such as RankGPT (Sun et al., 2024).

Finally, we provide extensive analyses of the effectiveness of QRHEAD. First, using QRHEAD outperforms both full attention heads and original retrieval heads. Second, QRHEAD **generalizes across different tasks and input lengths**—the heads identified at 32K tokens transfer well to tasks with 128K context lengths. Lastly, we show that both key modifications—our query-focused scoring objective and the use of natural data—contribute to the improved downstream performance of QRHEAD over original retrieval heads. Together, these findings highlight the practicality and robustness of QRHEAD as a foundation for long-context retrieval and suggest opportunities for further exploration of retrieval mechanisms in language models.

2 Background: Retrieval Heads

Retrieval heads are a specialized subset of attention heads that are pivotal for extracting relevant information from long input context.

Original retrieval heads. Wu et al. (2025b) first discovered a set of retrieval heads that exhibit copy-paste behavior during decoding—effectively copying tokens from the long context input context into the generated output. As shown in Figure 2 (top), the retrieval head detection method roots from the Needle-in-a-Haystack test (NIAH) with a triple (C, q, a) of context, question, and answer: the answer span a (the "needle") is embedded within a long context sequence $C = d_1...a...d_N$ where $d_1, ..., d_N$ are N irrelevant passages (the "haystack"). The LM is tasked to generate an answer to q based on the provided context. Successful generation of a demonstrates effective copy-paste behavior by extracting a from the haystack and copying it over to the output. We say an attention head h copies a token t if, during the generation of t in the answer, h assigns the maximum attention score to the same token t in the needles. To quantify this behavior, the retrieval score of an attention head h is defined as the fraction of tokens copied from a by the head h during decoding:

$$\text{Retrieval\_Score}(h) = \frac{|g_h \cap a|}{|a|}, \qquad (1)$$

where g_h denotes the set of tokens copied by head h to the output. Attention heads with the highest retrieval scores are selected as retrieval heads.

Shortcomings. The scoring mechanism described above focuses only on attention heads that perform strict copy-paste operations, potentially missing heads involved in semantic-based retrieval, such as paraphrasing or reasoning over relevant context. Moreover, recent work has shown that heads identified through copy-paste metrics exhibit limited cross-domain generalizability (Zhao et al., 2025). This suggests that the simplified formulation may not fully capture the complexity of in-context retrieval behavior in LLMs and has limited relevance for downstream applications.

3 QRHEAD: Identifying Query-Focused Retrieval Heads

In this section, we introduce a new approach for detecting retrieval heads that significantly improves upon prior retrieval head detection. For clarity,

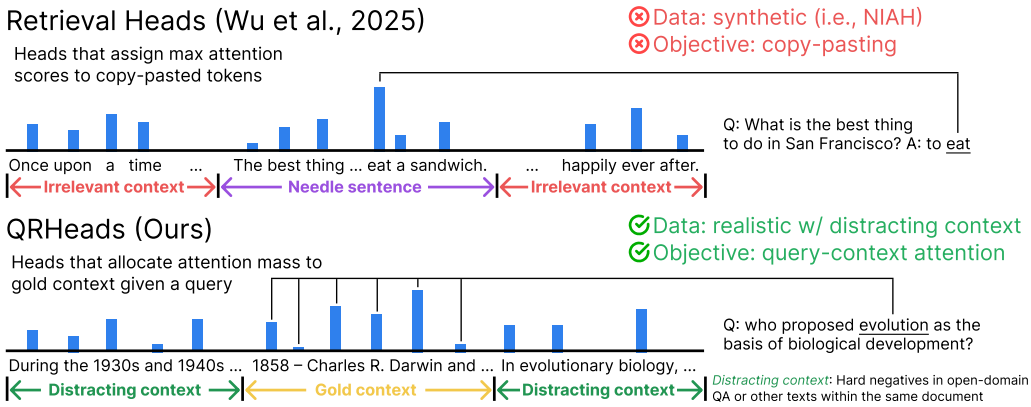

Figure 2: Comparison between Retrieval Heads (Wu et al., 2025b) and QRHEAD (Ours).

we refer to our heads as Query-Focused Retrieval Head (QRHEAD) and the original retrieval head as RETHEAD. See Figure 2 (bottom), our approach introduces two key improvements. First, we propose a query-focused retrieval score (QRscore), which captures query-context attention rather than relying solely on copy-paste behavior (§3.1). Second, we leverage realistic tasks that require in-context retrieval to identify effective heads (§3.2). We also present a comparison between QRHEAD and RETHEAD (§3.3).

Task formulation: LMs for in-context retrieval. Our study focuses on the task of in-context retrieval with LMs, i.e., identifying relevant information from given context. Formally, let Σ denote the vocabulary. Given an input query $q \in \Sigma^*$ and a context $D \in \Sigma^*$, the objective is to retrieve the most relevant information from the context with respect to q, denoted as $D_{[q]} \subseteq D$. Typically, the context D consists of a sequence of passages (or chunks), represented as $D = \{d_1, d_2, \ldots, d_N\}$. With both q and D jointly fed into an LM as input, we assign a score $\mathcal{R}(q, d_i)$ to each passage d_i with respect to q. We measure the effectiveness of the retriever by evaluating whether the top-scored passages align with the ground-truth relevant documents $D^*_{[q]}$.[2]

3.1 Scoring Heads with Query-Context Attention

Instead of scoring attention heads based on their activations in copy-paste operations, we propose to evaluate them based on their effectiveness in realistic in-context retrieval tasks. This offers a more general and realistic measure of retrieval capability, as it captures semantic relevance rather than relying solely on verbatim copying.

Query-focused retrieval score (QRscore). We use QRscore as a measure of the retrieval capability of an attention head in response to a specific query. Formally, let $h \in \mathcal{H}$ be an attention head within the language model, and let A_h denote the attention weights (post-softmax) of head h over a query prompt $\{D, q\}$, such as a prompt with a book followed by a question over its contents. The query-focused attention scores of head h towards a document d_i is calculated as follows:

$$\text{QRscore}_h(q, d_i) = \frac{1}{|q|} \sum_{t_q \in q} \sum_{t_d \in d_i} A_h^{t_q \to t_d}, \qquad (2)$$

where t_q denotes tokens in the query q, t_d represents tokens in the document d_i, and $A_h^{t_q \to t_d}$ is the attention weight of h from t_q to t_d. This formulation quantifies the degree to which head h focuses on document d_i in response to q. Lastly, we aggregate the scores for all documents d_i within the gold document set $D^*_{[q]}$, resulting in the final QRscore for head h with respect to the query q:

$$\text{QRscore}_h(q) = \frac{1}{|q|} \sum_{d_i \in D^*} \sum_{t_q \in q} \sum_{t_d \in d_i} A_h^{t_q \to t_d} \qquad (3)$$

3.2 Detecting QRHEAD on Real-World Tasks

With the QRscore defined in Eq. 3, we can now quantify the retrieval capabilities of each attention head over a given set of documents in response to a query. To achieve this, we leverage a head detection dataset $\mathcal{T} = \{(q, D, D^*_{[q]})\}$, which consists of a query q, a set of candidate documents D, and the corresponding gold documents $D^*_{[q]}$. Notably, our approach does not require explicit answers to the queries—only the annotations of the gold document. Using this detection dataset $\mathcal{T}$, we compute

[2] We note NIAH task can also be viewed as a special case of this formulation, where the gold document set only contains one document (the needle).

the effectiveness of an attention head h for retrieval as follows:

$$\text{QRscore}_{h,\mathcal{T}} = \frac{1}{|\mathcal{T}|} \sum_{(q,D,D^*)\in\mathcal{T}} \text{QRscore}_h(q) \quad (4)$$

As shown in Figure 2, instead of synthetic needle-in-a-haystack task (NIAH) (Kamradt, 2023), we use more realistic in-context retrieval task for head detection (e.g., claim verification over books). We argue that more natural and realistic distractors provide more effective supervision that allows identifying heads that are better at differentiating relevant context from distracting context. We also note that even a small amount (< 100) of realistic data points can be sufficient, allowing us to find QRHEAD heads that contribute to improved downstream performance.

3.3 Comparing QRHEAD and Original Retrieval Head

We have introduced our method for detecting QRHEAD. Here, we compare the QRHEAD with original retrieval head (RETHEAD) within the same model, using Llama-3.1-8B-Instruct (Llama-3 Team, 2024) as a case study.

First, following the analysis setup of Wu et al. (2025b), we measure the impact of pruning by the *performance drop on NIAH test*. Specifically, we prune the top 32 heads (roughly 3% of all attention heads in LLaMA-3.1-8B), following the commonly reported 5% sparsity level of retrieval heads in Wu et al. (2025b); Zhao et al. (2025). As shown in Figure 1, pruning the top 32 QRHEAD results in near-complete failure on the NIAH performance, whereas pruning the top 32 RETHEAD yields a much smaller performance decline.[3] In addition, we find **substantial divergence** between the two sets. Among the top 32 and top 64 heads, only 8 and 32 overlap, respectively. This less than 25% overlap in the top 32 highlights the distinct roles of QRHEAD and RETHEAD.

4 Using QRHEAD to Build A General-Purpose Retriever

In this section, we describe how the detected QRHEAD can be used in downstream applications. Specifically, we find the attention mass of QRHEAD provides highly reliable signals for in-context retrieval.

[3]See Appendix B for results on Qwen-2.5-7B-Instruct.

4.1 The Method

Given a selected set of QRHEAD $\mathcal{H}^{\text{select}}$, a query q, and a collection of passages D, we compute the retrieval score for each passage d_i by aggregating the QRscore across all heads in $\mathcal{H}^{\text{select}}$:

$$\mathcal{R}(q, d_i) = \frac{1}{|\mathcal{H}^{\text{select}}|} \sum_{h\in\mathcal{H}^{\text{select}}} \text{QRscore}_h(q, d_i). \quad (5)$$

Passages are then ranked using their retrieval scores. We call our retrieval system QRRETRIEVER. It offers several advantages: (1) *General-purpose:* applicable across diverse domains without training, unlike traditional retrievers that often are often limited in generalizing out of domain (2) *Model-agnostic:* compatible with any transformer-based LMs without modification, (3) *Efficient:* leverages attention patterns to process long context simultaneously without expensive generation or pairwise comparisons.

Calibration. To mitigate intrinsic biases in LMs' attention weights, we adopt the score calibration method proposed by Chen et al. (2025). Instead of directly using $R(q, d_i)$ as the score, we additionally compute baseline score, $R(q_{null}, d_i)$, using a context-free null query q_{null} (`"N/A"`). For each d_i, we use calibrated the score $R(q, d_i) - R(q_{null}, d_i)$ as the final retriever score.

4.2 Applications

Long-context reasoning. Long-context language models often struggle with performance degradation when processing long context (Yen et al., 2025; Ye et al., 2025; Liu et al., 2024a). To address this, we integrate QRRETRIEVER within a retrieval-augmented generation (RAG) framework. Given a long-context input and a query, we segment the input into smaller chunks and use QRRETRIEVER to score and subsequently extract the most relevant ones. The extracted context are concatenated to create a reduced context, that is then given to the LM for generating the final answer in another forward pass.

Passage re-ranking. Text retrieval powers many retrieval-augmented downstream applications (Lewis et al., 2020). A critical component in the retrieval pipeline is the re-ranker, which reorders the passages returned by a first-stage retriever to enhance top passage relevance (Nogueira and Cho, 2020; Ma et al., 2024). QRRETRIEVER can naturally be used as a re-ranker as part of any

Retriever	LongMemEval				CLIPPER			
	Retrieval Recall@k		End-to-End Performance		Retrieval Recall@k		End-to-End Performance	
	k = 5	k = 10	Top-5	Top-10	k = 3	k = 5	Top-3	Top-5
Base LM: Llama-3.2-3B-Instruct								
Full context	-	-	28.1		-	-	25.2	
BM25	57.5	67.5	46.1	44.9	74.6	83.7	20.0	22.8
Contriever	62.7	79.2	**48.6**	46.5	60.2	78.9	12.6	18.4
Stella	63.9	77.6	44.9	**47.7**	83.3	90.0	21.3	25.1
RankGPT	1.8	3.4	23.5	23.3	16.8	27.3	3.6	8.8
RankGPT$^{\text{Bubble}}$	2.1	3.8	24.0	24.4	17.0	27.4	3.8	8.8
ICR	68.7	78.8	46.5	45.1	72.8	83.6	19.4	23.6
QRRetriever (Ours)	**77.6**	**86.6**	47.4	**47.7**	**85.5**	**93.4**	23.4	**26.9**
Base LM: Llama-3.1-8B-Instruct								
Full context	-	-	46.5		-	-	31.3	
BM25	57.5	67.5	48.8	50.9	74.6	83.7	37.9	37.9
Contriever	62.7	79.2	52.6	55.4	60.2	78.9	28.2	31.1
Stella	63.9	77.6	50.9	58.4	83.3	90.0	38.8	39.6
RankGPT	2.1	4.0	26.7	24.2	30.0	39.4	15.9	19.4
RankGPT$^{\text{Bubble}}$	8.3	9.0	28.1	27.0	36.7	44.3	19.7	20.4
ICR	77.0	84.4	59.3	56.1	89.3	94.7	43.8	**42.5**
QRRetriever (Ours)	**85.5**	**91.7**	**59.8**	**60.2**	**93.8**	**96.9**	**47.6**	41.9
Base LM: Llama-3.1-70B-Instruct								
Full context	-	-	34.2		-	-	63.9	
BM25	57.5	67.5	52.8	53.0	74.6	83.7	60.1	66.5
Contriever	62.7	79.2	53.7	60.5	60.2	78.9	38.5	49.7
Stella	63.9	77.6	56.3	62.3	83.3	90.0	65.9	71.2
RankGPT	1.8	3.5	21.2	27.4	57.0	63.4	44.7	50.4
RankGPT$^{\text{Bubble}}$	47.9	49.0	44.0	42.6	74.3	78.8	58.4	61.5
ICR	32.1	46.5	36.5	39.8	86.2	93.1	68.9	71.6
QRRetriever (Ours)	**80.4**	**88.5**	**66.7**	**67.7**	**95.0**	**98.2**	**74.2**	**73.3**

Table 1: Results on LongMemEval and CLIPPER. The base model denotes the LM used for both the retriever and end-to-end generation. QRHEAD used for CLIPPER are found through using LongMemEval.

retrieval pipeline without any fine-tuning by simply concatenating the retrieved passages in the input and scoring their relevance directly.

5 Experiments

We evaluate QRRETRIEVER on two tasks: long-context reasoning (§5.2) and re-ranking (§5.3).

5.1 Base Models and Baselines

Base LMs. We experiment with open-weight, instruction-tuned LMs from two families across different sizes, including Llama-3.2 (3B), Llama-3.1 (8B and 70B) of Llama family (Llama-3 Team, 2024), and Qwen2.5 (7B) of Qwen family (Yang et al., 2024). With QRRETRIEVER, we use 16 heads for models with fewer than 10B parameters, and 32 heads for Llama-3.1-70B. This corresponds to approximately 1–2% of the total attention heads, given the sparsity of retrieval heads.

Baselines. We compare our methods against several strong baselines. Following Wu et al. (2025a), we compare against dense retrievers, including Contriever (Izacard et al., 2022) and 1.5B Stella V5 (Zhang et al., 2025), two popular strong dense retrievers. For Contriever, we truncate the input to 512 tokens according to its maximum context length. We also compare against existing LLM-based re-rankers, including:

- **RankGPT** (Sun et al., 2024) is a generative re-ranker that instructs LLMs to output the ranking order of a given set of documents based on a query. We experiment with two variants of RankGPT: (1) **RankGPT without sliding window**, which directly inputs all documents into the model prompt simultaneously, and (2) **RankGPT with sliding window** (RankGPT$^{\text{Bubble}}$), which leverages bubble sort to rank smaller subsets of documents incrementally.
- **In-Context-Re-ranking** (ICR; Chen et al., 2025) is a re-ranker that also leverages the attention for relevance scoring. ICR uses full attention heads for scoring relevace, whereas we only use the attention weights of selected QRHEAD.

5.2 Long-Context Multi-Hop Reasoning

Datasets. We use 1) **LongMemEval** (Wu et al., 2025a), which evaluates the long-term memory capabilities of LLM-driven chat assistants, and 2) **CLIPPER** (Pham et al., 2025), which evaluate claim-verification over books. Both datasets feature long-context (90K to 120K) and require multi-hop reasoning over several pieces of evidences. We

	NQ	COVID	NFCorpus	FiQA	Scifact	Scidocs	FEVER	Climate	DBPedia	Robust04	News	Avg
BM25	30.5	59.5	32.2	23.6	67.9	14.9	65.1	16.5	31.8	40.7	39.5	38.4
Base LM: Llama-3.2-3B-Instruct												
RankGPT	30.0	59.5	32.2	23.6	67.9	14.9	65.9	17.1	31.8	40.7	39.5	38.5
RankGPT$^{\text{Bubble}}$	33.2	61.8	32.0	22.4	66.1	14.8	65.8	17.1	34.8	40.5	40.2	39.0
ICR	49.2	72.3	33.8	31.8	73.3	17.4	82.6	24.2	34.7	47.2	44.7	46.5
QRRETRIEVER (Ours)	**54.9**	**77.4**	**35.1**	**35.1**	**74.7**	**18.3**	**83.7**	**24.5**	**36**	**49.7**	**45.1**	**48.6**
Base LM: Llama-3.1-8B-Instruct												
RankGPT	30.0	59.5	32.2	23.6	67.9	14.9	65.9	16.8	31.8	40.7	39.5	38.4
RankGPT$^{\text{Bubble}}$	53.7	75.5	34.3	31.4	69.3	17.4	67.5	23.8	**42.9**	47.8	**46.2**	46.3
ICR	54.0	73.3	34.8	35.6	75.5	19.0	**85.8**	**24.8**	36.9	49.0	44.5	48.5
QRRETRIEVER (Ours)	**58.6**	**77.5**	**35.3**	**39.1**	**76.2**	**19.4**	85.3	23.9	37.2	**51.4**	**46.2**	**50.0**
Base LM: Qwen-2.5-7B-Instruct												
RankGPT	30.0	59.5	32.2	23.6	67.9	14.9	65.9	16.8	31.8	40.7	39.5	38.4
RankGPT$^{\text{Bubble}}$	42.7	**70.5**	**34.1**	**29.5**	69.3	**16.6**	70.5	19.7	**37.1**	**46.4**	**43.6**	43.6
ICR	43.1	66.1	32.7	27.0	**71.1**	16.4	79.2	19.6	35.3	43.0	40.0	43.0
QRRETRIEVER (Ours)	**49.9**	67.7	33.1	29.2	71	15.3	**80.7**	**20.1**	35.7	43.7	39.8	**44.2**
Base LM: Llama-3.1-70B-Instruct												
RankGPT	45.4	62.7	33.6	28.6	71.3	16.1	74.2	18.9	37.6	41.3	39.8	42.7
RankGPT$^{\text{Bubble}}$	58.4	**81.2**	**36.1**	41.0	76.1	20.2	80.0	**25.1**	**45.5**	**59.0**	**48.5**	**51.9**
ICR	57.9	72.3	34.2	38.9	74.6	19.4	**86.4**	22.0	38.3	42.6	40.4	47.9
QRRETRIEVER (Ours)	**62.1**	73.9	34.7	**43.8**	**76.8**	**20.4**	85.9	23.1	35.7	51.6	43.5	50.1
Dual Encoder and Cross Encoder												
Contriever	44.6	67.5	32.8	28.4	67.1	18.9	64.2	28.0	39.5	45.7	41.7	43.5
GTR-t5-base	51.4	74.8	32.5	34.7	62.1	15.8	72.9	26.8	37.1	46.1	42.8	45.2
BGE-Reranker-base	55.2	66.4	31.0	31.7	70.8	15.7	88.6	36.5	42.5	39.9	37.0	46.8
msmarco-MiniLM	55.8	74.3	35.2	35.1	68.5	17.5	80.4	25.5	45.3	47.9	43.0	48.0

Table 2: Performance comparison (nDCG@10) on BEIR benchmarks across LMs. QRRETRIEVER generally outperforms other baselines across all models. With Llama-3.1-70B, QRRETRIEVER underperforms RankGPT with (Bubble sort), which requires substantial amount of LLM generation calls. We also report the performance of popular dual encoders and cross encoders for reference (gray box).

segment each dataset according to its natural structure (e.g., message in multi-turn conversation or chapters in a book). For evaluation, we measure retrieval performance using recall and assess downstream task performance with accuracy. Please refer to Appendix A for more details.

Data for head detection. We detect QRHEAD using a small subset of data from LongMemEval, specifically the single-session-user subset (which only requires single-hop reasoning) consisting of 70 examples, which we exclude from downstream evaluation. We use the set of heads for both LongMemEval and CLIPPER, testing generalization to multi-hop reasoning.

QRRETRIEVER achieves strong retrieval performance for long context, leading to improved end-to-end performance. Table 1 demonstrates the strong performance of QRRETRIEVER on both LongMemEval and CLIPPER: it outperforms other baselines regarding both retrieval recall and end-to-end performance. For instance, Llama-3.1-8B-Instruct as the base LM, we see end-to-end performance improvements of over 10% on both tasks with Llama-3.1-8B-Instruct.

QRRETRIEVER generalizes across domains. QRRETRIEVER outperform off-the-shelf dense retrievers (Contriever and Stella) by a large margin on LongMemEval and CLIPPER. In particular, none of these methods are trained or calibrated on CLIPPER. The better performance of QRRETRIEVER suggests its stronger cross-domain generalization capabilities than dense retrievers.

It is worth noting that all test questions in LongMemEval are multi-hop, yet QRRETRIEVER performs well on them despite only using single-hop questions to detect QRHEAD.

QRRETRIEVER scales with model sizes. We note that LM-based re-rankers show inconsistent performance patterns across model scales: RankGPT achieves near-zero retrieval recall with small models, and retrieval performance of ICR sees significant degradation when scaling up model size from 8B to 70B. At the same time, the performance of QRRETRIEVER generally improves as the model size scales up.

Compact LMs exhibit strong retrieval capabilities despite their limited generation abilities. As shown in Table 1, on LongMemEval, Llama-3.2-3B-Instruct achieves a Recall@10 of 86.6, closely matching the 88.5 score of the much larger LlamA-3.1-70B. However, Llama-3.2-3B only achieves a final end-to-end performance of 47.7, largely lagging 70B's performance of 67.7. We hypothesize that the long-context limitations of compact models stem more from their generation capabilities than from their retrieval abilities. These findings open up promising future directions. Compact LMs could serve as efficient long-context retrievers, paired with larger models for the actual generation.

5.3 Passage Re-Ranking

To test the general applicability of QRRETRIEVER, we evaluate our method on BEIR benchmark (Thakur et al., 2021) consisting of diverse domains. We compare against zero-shot LLM-based re-rankers, RankGPT and ICR. We also report the performance of popular dual encoders and cross encoders from sentence transformer for references (please refer to Appendix C for details about these models).

Setting. Our setting largely follows prior work (Chen et al., 2025). We re-rank 200 passages retrieved using BM25, resulting a overall context length ranging from 16K to 64K depending on the average document length of domains. We report the performance on the set of tasks used in Chen et al. (2025), we sub-sampled 512 random questions for each domain for evaluation.

Data for head detection. For BEIR, we utilize the 256 (held-out) data points from NQ and use them for on all other domains zero-shot.

Results. Table 2 summarizes the BEIR results, demonstrating the strong effectiveness of QRRETRIEVER as a general-purpose retriever. For models under 10B parameters, QRRETRIEVER consistently outperforms other baselines. With Llama-3.1-8B, it achieves an average score of 50.0, outperforming RankGPT by 3.7 points and ICR by 1.5 points. For the larger Llama-3.1-70B model, QRRETRIEVER significantly surpasses ICR, though it generally lags RankGPT$^{\text{Bubble}}$ (which require over 200 generation calls). Nevertheless, QRRETRIEVER achieves the best performance on several domains, such as SciFact and FiQA. In addition, we find that QRRETRIEVER directly benefits from the general-purpose capabilities of strong LLMs, outperforming popular dual encoders and cross-encoders that fine-tune various base models to improve retrieval performance.

		BEIR$_{\text{SHUFFLED}}$ NDCG@10	LONGMEMEVAL RECALL
Llama-8B	RANDOM HEADS	37.5	59.8
	FULL HEADS	42.8	77.0
	RETRIEVAL HEAD	43.4	81.5
	QRHEAD	**47.5**	**85.5**
Qwen-7B	RANDOM HEADS	19.9	57.2
	FULL HEADS	22.6	67.1
	RETRIEVAL HEAD	27.4	70.7
	QRHEAD	**31.9**	**83.2**

Table 3: Comparison across head selection strategies. Using QRHEAD substantially outperforms using all heads or using original retrieval heads.

6 Analysis

In this section, we analyze the key advantages of QRHEAD (e.g., length generalization) and examine the factors underlying its effectiveness. Additional analyses on the impact of varying the number of heads (Appendix F) and inference latency (Appendix E) are provided in the appendix.

6.1 Impact of Head Selection

We provide further ablations on head selection, the core idea behind QRRETRIEVER. We experiment with different sets of heads, including (1) using our QRHEAD, (2) using all the attention heads, (3) using original retrieval head, and (4) using randomly selected heads. We use 16 heads for all settings. Table 3 presents the retrieval performance on LongMemEval re-ranking performance on BEIR (aggregated across tasks).[4] The performance gaps between different strategies demonstrate the importance of using the right heads for retrieval. Using original retrieval heads is effective, compared to using random heads or full heads. Using our improved QRHEAD consistently outperforms using original retrieval heads.

6.2 Generalizability Across Lengths

We test the length generalization of QRHEAD: if we detect QRHEAD on relatively short context length (32K), can the heads generalize to longer context lengths (128K)?

We test such short-to-long generalization by controlling the number of documents in context.

[4]Here, we use BEIR where input documents are randomly shuffled rather than ranked by BM25. This setup allows uniform evaluation of retrieval across the full context.

	Model: LLama-3.1-8B-Instruct			
	NQ+FEVER		LongMemEval	
	32K	128K	32K	128K
ICR	66.7	56.5	85.2	78.2
QRRETRIEVER^{32K}	**70.1**	63.9	89.2	85.2
QRRETRIEVER^{128K}	68.8	**67.2**	**89.2**	**85.6**
	Model: Qwen-2.5-7B-Instruct			
	NQ+FEVER		LongMemEval	
	32K	64K	32K	64K
ICR	40.0	17.4	83.4	67.1
QRRETRIEVER^{32K}	51.9	25.3	**90.2**	**77.9**
QRRETRIEVER^{64K}	**54.1**	**29.1**	90.1	77.0

Table 4: Results on short-to-long generalization of QRHEAD. QRHEAD detected with relative short-context data can be used for retrieval on longer context.

	Data	BEIR NDCG@10	LONGMEM RECALL
	Model: LLama-3.1-8B-Inst		
QRHEAD	NQ	**47.5**	83.9
QRHEAD	LongMem	47.1	**85.6**
QRHEAD	NIAH	46.8	83.4
RETRIEVAL HEAD	NIAH	43.4	81.5
	Model: Qwen-2.5-7B-Inst		
QRHEAD	NQ	31.9	80.2
QRHEAD	LongMem	**32.1**	**83.2**
QRHEAD	NIAH	30.9	79.7
RETRIEVAL HEAD	NIAH	27.4	70.7

Table 5: Analysis of factors contributing to improved head selection. Applying QRScore (§3.1) on NIAH results in more effective heads than the original retrieval heads. Using QRScore on realistic tasks yields the most effective head selection overall.

This results in datasets of different lengths ranging from 32K to 128K tokens. We detect QRHEAD from both short and long datasets and test their performance on re-ranking tasks (using two representative subsets: NQ and FEVER) and LongMemEval. For Qwen-2.5-7B, we set the longer context length to 64K due to its original 32K limit. As shown in Table 4, QRHEAD detected using short-context data can generalize to longer-context settings, though heads detected from longer data generally yield better long-context performance.

6.3 Ablations on Scoring Function and Task Selection for Head Detection

In §3, we describe two key factors for head detection: using query-context attention objective, and using realistic data. To assess the importance of these factors, we experiment with detecting heads on NIAH using QRscore (§3). As shown in Table 5, applying QRscore on NIAH leads to improved performance compared to using the original retrieval heads detected from the same task. However, using realistic tasks such as NQ and LongMemEval with QRscore yields the best overall performance. These results highlight the importance of both the scoring method and head detection data.

	Overlap (Top 64)			BEIR
	Set0	Set1	Set2	nDCG@10
	Model: LLama-3.1-8B-Inst			
QRHEAD^{Set0}	64	51	51	49.8
QRHEAD^{Set1}	51	64	53	49.7
QRHEAD^{Set2}	51	53	64	49.9
	Model: Qwen-2.5-7B-Inst			
QRHEAD^{Set0}	64	50	53	44.2
QRHEAD^{Set1}	50	64	57	44.4
QRHEAD^{Set2}	53	57	64	44.5

Table 6: **Left**: Overlap in QRHEAD identified using three disjoint sets of 128 random samples from NQ. **Right**: BEIR performance (nDCG@10) using QRHEAD detected from each sample set.

6.4 Sensitivity of QRHEAD Detection to Variation in Detection Data

In Section 5.3, we show using a small number of samples from NQ is sufficient to identify effective QRHEAD for BEIR re-ranking tasks. We assess the robustness of this head detection process to different random samples of detection set, by experimenting with three disjoint random subsets of NQ, each containing 128 examples. Table 6 presents the overlap among the top-64 heads selected from these subsets and their performance on BEIR benchmark. Across two LLMs from different model families (Llama and Qwen), we observe a high degree of consistency with over 50 heads overlapping among the top 64 across subsets. Furthermore, the downstream performance remains stable across these variations. These results indicate that QRRETRIEVER can be reliably identified using a small sample of data.

7 Related Work

LM-based retrieval and re-ranking. LMs are widely used in retrieval, including embedding-based methods (Muennighoff, 2022; Lee et al., 2021) and generative approaches (Tay et al., 2022; Cao et al., 2021; Sun et al., 2023). For re-ranking, instruction-tuned LMs been adapted as re-rankers in various ways (Sun et al., 2024; Drozdov et al., 2023; Sachan et al., 2023; Ma et al., 2023; Pradeep et al., 2023), leveraging their generation capabilities. Similar to our approach, recent work has explored using logits (Reddy et al., 2024) or aggregated attention scores (Chen et al., 2025) for re-ranking. In contrast, we identify a specialized set of attention heads responsible for retrieval, offering improved performance and interpretability.

Localizing model behavior. Interpretability studies have shown that many core behaviors of LMs, including in-context learning (Olsson et al., 2022; Todd et al., 2024; McDougall et al., 2023) and retrieval (Wu et al., 2025b), can be traced to specialized transformer modules (Meng et al., 2022; Dai et al., 2022; Stolfo et al., 2024). Techniques have been proposed to localize such modules with a small amount of data (Meng et al., 2022; Geiger et al., 2024; Bhaskar et al., 2024), and to intervene on them for control (Li et al., 2023; Yin et al., 2024; Huang et al., 2025) or efficiency (Tang et al., 2025; Xiao et al., 2025; Liu et al., 2024b). However, only a few works (Zhao et al., 2025) have examined attention head specialization in long-context settings, where attention is known to be not robust (Liu et al., 2024a; Xiao et al., 2024a), and it is an open question if intervening the localized modules is crucial in practical settings (Hase et al., 2023; Wang and Veitch, 2024). Our work contributes to this line of research by finding better specialized set of attention heads that explain the model behavior for query-focused long-context retrieval, and that can be practically useful for zero-shot efficient retrieval.

8 Conclusion

We introduced Query-Focused Retrieval Heads (QRHEAD), a set of attention heads specialized in identifying query-relevant information in long-context inputs. Detected using query-context attention scores on realistic data, QRHEAD are better aligned with practical retrieval tasks than original retrieval heads. Built on top of QRHEAD, our retrieval method QRRETRIEVER achieves strong performance on both long-context reasoning and re-ranking tasks, outperforming dense retrievers and other LLM-based re-rankers in many settings. These findings highlight the practical utility of QRHEAD and offer insights for further improving retrieval with LMs.

Limitations

Our work detects improved retrieval heads and builds general-purpose retrievers based on them. We do not explore techniques that involve updating model parameters, as our goal is to develop flexible methods that can directly use off-the-shelf models as retrievers. Consequently, we leave to future work the investigation of parameter-updating techniques that leverage insights from QRHEAD.

While our method finds that QRHEAD can enhance downstream performance, and shows the importance of two factors leading to selection of better heads. We lack a complete understanding of the internal mechanism accounting for QRHEAD's effectiveness. Future work could apply circuit analysis techniques (e.g., Bhaskar et al. (2024); Shi et al. (2024)) to dissect the fine-grained behaviors and roles of these heads.

Our evaluation primarily targets passage re-ranking and long-context multi-hop reasoning tasks. Although our approach is conceptually applicable to broader long-context tasks—such as long-document summarization (Shaham et al., 2023; Laban et al., 2024)—it remains unclear whether it generalizes to such tasks without thorough empirical validation.

Finally, our experiments are limited to English datasets. As LMs may exhibit different behaviors across languages, the cross-lingual robustness of our approach remains an open question.

9 Acknowledgments

This work is gratefully supported by an NSF CAREER award (IIS-2239290) and a grant from Intel. Howard Yen is supported by the William A. Dippel' 50 * 55 Graduate Fellowship.

References

Adithya Bhaskar, Alexander Wettig, Dan Friedman, and Danqi Chen. 2024. Finding transformer circuits with edge pruning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2021. Autoregressive entity retrieval. *Preprint*, arXiv:2010.00904.

Shijie Chen, Bernal Jiménez Gutiérrez, and Yu Su. 2025. Attention in large language models yields efficient zero-shot re-rankers. *Preprint*, arXiv:2410.02642.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8493–8502, Dublin, Ireland. Association for Computational Linguistics.

Andrew Drozdov, Honglei Zhuang, Zhuyun Dai, Zhen Qin, Razieh Rahimi, Xuanhui Wang, Dana Alon, Mohit Iyyer, Andrew McCallum, Donald Metzler, and Kai Hui. 2023. Parade: Passage ranking using demonstrations with large language models. *Preprint*, arXiv:2310.14408.

Atticus Geiger, Zhengxuan Wu, Christopher Potts, Thomas Icard, and Noah Goodman. 2024. Finding alignments between interpretable causal variables and distributed neural representations. In *Proceedings of the Third Conference on Causal Learning and Reasoning*, volume 236 of *Proceedings of Machine Learning Research*, pages 160–187. PMLR.

Peter Hase, Mohit Bansal, Been Kim, and Asma Ghandeharioun. 2023. Does localization inform editing? surprising differences in causality-based localization vs. knowledge editing in language models. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Lei Huang, Xiaocheng Feng, Weitao Ma, Yuchun Fan, Xiachong Feng, Yangfan Ye, Weihong Zhong, Yuxuan Gu, Baoxin Wang, Dayong Wu, Guoping Hu, and Bing Qin. 2025. Improving contextual faithfulness of large language models via retrieval heads-induced optimization. *arXiv preprint arXiv:2501.13573*.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised dense information retrieval with contrastive learning. *Preprint*, arXiv:2112.09118.

Garrett Kamradt. 2024. Needle in a haystack - pressure testing llms.

Gregory Kamradt. 2023. Needle In A Haystack - pressure testing LLMs.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Philippe Laban, Alexander Fabbri, Caiming Xiong, and Chien-Sheng Wu. 2024. Summary of a haystack: A challenge to long-context LLMs and RAG systems. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9885–9903. Association for Computational Linguistics.

Jinhyuk Lee, Alexander Wettig, and Danqi Chen. 2021. Phrase retrieval learns passage retrieval, too. *Preprint*, arXiv:2109.08133.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2023. Inference-time intervention: Eliciting truthful answers from a language model. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024a. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Wenhan Liu, Xinyu Ma, Yutao Zhu, Ziliang Zhao, Shuaiqiang Wang, Dawei Yin, and Zhicheng Dou. 2024b. Sliding windows are not the end: Exploring full ranking with long-context large language models. *Preprint*, arXiv:2412.14574.

Llama-3 Team. 2024. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. 2024. Fine-tuning llama for multi-stage text retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, page 2421–2425, New York, NY, USA. Association for Computing Machinery.

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

Callum McDougall, Arthur Conmy, Cody Rushing, Thomas McGrath, and Neel Nanda. 2023. Copy Suppression: Comprehensively Understanding an Attention Head. *arXiv preprint arXiv:2310.04625*.

Kevin Meng, David Bau, Alex J Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems*.

Niklas Muennighoff. 2022. Sgpt: Gpt sentence embeddings for semantic search. *Preprint*, arXiv:2202.08904.

Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernandez Abrego, Ji Ma, Vincent Zhao, Yi Luan, Keith Hall, Ming-Wei Chang, and Yinfei Yang. 2022. Large dual encoders are generalizable retrievers. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*.

Rodrigo Nogueira and Kyunghyun Cho. 2020. Passage re-ranking with bert. *Preprint*, arXiv:1901.04085.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, and 7 others. 2022. In-context Learning and Induction Heads. *arXiv preprint arXiv:2209.11895*.

Chau Minh Pham, Yapei Chang, and Mohit Iyyer. 2025. Clipper: Compression enables long-context synthetic data generation. *Preprint*, arXiv:2502.14854.

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

Revanth Gangi Reddy, JaeHyeok Doo, Yifei Xu, Md Arafat Sultan, Deevya Swain, Avirup Sil, and Heng Ji. 2024. First: Faster improved listwise reranking with single token decoding. *Preprint*, arXiv:2406.15657.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Devendra Singh Sachan, Mike Lewis, Mandar Joshi, Armen Aghajanyan, Wen tau Yih, Joelle Pineau, and Luke Zettlemoyer. 2023. Improving passage retrieval with zero-shot question generation. *Preprint*, arXiv:2204.07496.

Uri Shaham, Maor Ivgi, Avia Efrat, Jonathan Berant, and Omer Levy. 2023. ZeroSCROLLS: A zero-shot benchmark for long text understanding. In *Findings of the Conference on Empirical Methods in Natural Language Processing (EMNLP Findings)*.

Claudia Shi, Nicolas Beltran-Velez, Achille Nazaret, Carolina Zheng, Adrià Garriga-Alonso, Andrew Jesson, Maggie Makar, and David Blei. 2024. Hypothesis testing the circuit hypothesis in LLMs. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Alessandro Stolfo, Ben Peng Wu, Wes Gurnee, Yonatan Belinkov, Xingyi Song, Mrinmaya Sachan, and Neel Nanda. 2024. Confidence regulation neurons in language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Weiwei Sun, Lingyong Yan, Zheng Chen, Shuaiqiang Wang, Haichao Zhu, Pengjie Ren, Zhumin Chen, Dawei Yin, Maarten de Rijke, and Zhaochun Ren. 2023. Learning to tokenize for generative retrieval. *Preprint*, arXiv:2304.04171.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 2024. Is chatgpt good at search? investigating large language models as re-ranking agents. *Preprint*, arXiv:2304.09542.

Hanlin Tang, Yang Lin, Jing Lin, Qingsen Han, Danning Ke, Shikuan Hong, Yiwu Yao, and Gongyi Wang. 2025. Razorattention: Efficient KV cache compression through retrieval heads. In *The Thirteenth International Conference on Learning Representations*.

Yi Tay, Vinh Q. Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Gupta, Tal Schuster, William W. Cohen, and Donald Metzler. 2022. Transformer memory as a differentiable search index. *Preprint*, arXiv:2202.06991.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Eric Todd, Millicent Li, Arnab Sen Sharma, Aaron Mueller, Byron C Wallace, and David Bau. 2024. Function vectors in large language models. In *The Twelfth International Conference on Learning Representations*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Zihao Wang and Victor Veitch. 2024. Does editing provide evidence for localization? In *ICML 2024 Workshop on Mechanistic Interpretability*.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. 2025a. Longmemeval: Benchmarking chat assistants on long-term interactive memory. *Preprint*, arXiv:2410.10813.

Wenhao Wu, Yizhong Wang, Guangxuan Xiao, Hao Peng, and Yao Fu. 2025b. Retrieval head mechanistically explains long-context factuality. In *The Thirteenth International Conference on Learning Representations*.

Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, junxian guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. 2025. Duoattention: Efficient long-context LLM inference with retrieval and streaming heads. In *The Thirteenth International Conference on Learning Representations*.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024a. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations*.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2024b. C-pack: Packed resources for general chinese embeddings. In *Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval*, pages 641–649.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 39 others. 2024. Qwen2 technical report. *ArXiv*, abs/2407.10671.

Sohee Yang and Minjoon Seo. 2020. Is retriever merely an approximator of reader? *ArXiv*, abs/2010.10999.

Xi Ye, Fangcong Yin, Yinghui He, Joie Zhang, Howard Yen, Tianyu Gao, Greg Durrett, and Danqi Chen. 2025. Longproc: Benchmarking long-context language models on long procedural generation. In *arXiv*.

Howard Yen, Tianyu Gao, Minmin Hou, Ke Ding, Daniel Fleischer, Peter Izsak, Moshe Wasserblat, and Danqi Chen. 2025. Helmet: How to evaluate long-context language models effectively and thoroughly.

Fangcong Yin, Xi Ye, and Greg Durrett. 2024. Lofit: Localized fine-tuning on LLM representations. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Dun Zhang, Jiacheng Li, Ziyang Zeng, and Fulong Wang. 2025. Jasper and stella: distillation of sota embedding models. *Preprint*, arXiv:2412.19048.

Xinyu Zhao, Fangcong Yin, and Greg Durrett. 2025. Understanding synthetic context extension via retrieval heads. In *Proceedings of ICML*.

A Details about Evaluation Datasets

We use LongMemEval (Wu et al., 2025a) and CLIPPER (Pham et al., 2025) for evaluating our systems on long-context reasoning.

LongMemEval evaluates the long-term memory capabilities of LLM-driven chat assistants across five fundamental abilities: information extraction, multi-session reasoning, temporal reasoning, knowledge updates, and abstention. We segment the LongMemEval-S dataset ($\sim$115k tokens/question) at the round level, where each round is a document consisting of a single user message paired with the corresponding assistant response.

CLIPPER targets narrative claim verification—a challenging long-context reasoning task that requires verifying claims over entire books, with an average length of 90K tokens and 23 chapters. In CLIPPER, data is split at the chapter level, with each chapter treated as an individual document during retrieval.

Evaluation Process For each question, we first feed the entire context (e.g., all chapters or dialogue rounds) into the language model without using any first-stage retriever. We compute a retrieval score for each document or segment using our method described in §4. We then select the top-k documents based the scores, concatenate them, and feed them together with the query into the language model in a second pass to generate the final answer. We choose $k = 5, 10$ for LongMemEval and $k = 3, 5$ for Clipper. We report retrieval performance using recall and downstream task performance using accuracy.

B NIAH Test on Qwen-2.5-7B-Instruct

We evaluate Qwen-2.5-7B-Instruct on the NIAH test by masking selected attention heads. As shown in Figure 3 and Figure 4, pruning the top 16 QRHEAD leads to a more substantial degradation in NIAH performance compared to pruning the top 16 RETHEAD, indicating the greater functional importance of QRHEAD. When pruning the top 32 heads, the performance gap between QRHEAD and RETHEAD narrows, suggesting that QRHEAD achieves better efficiency and effectiveness with fewer heads for retrieval in NIAH task.

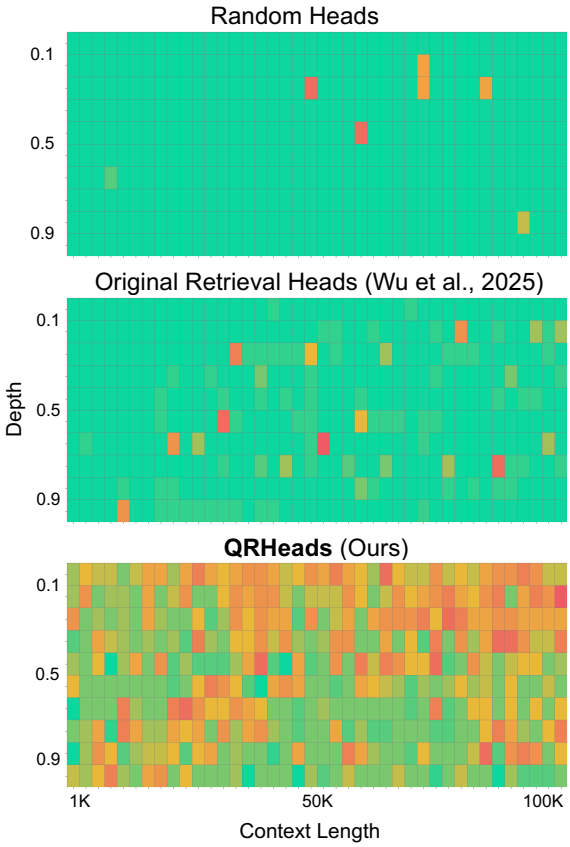

Figure 3: **Top**: Masking 16 random heads of Qwen2.5-7B-Instruct. **Middle**: Masking the top 16 original retrieval heads (Wu et al., 2025b). **Bottom**: Masking the top 16 QRHeads.

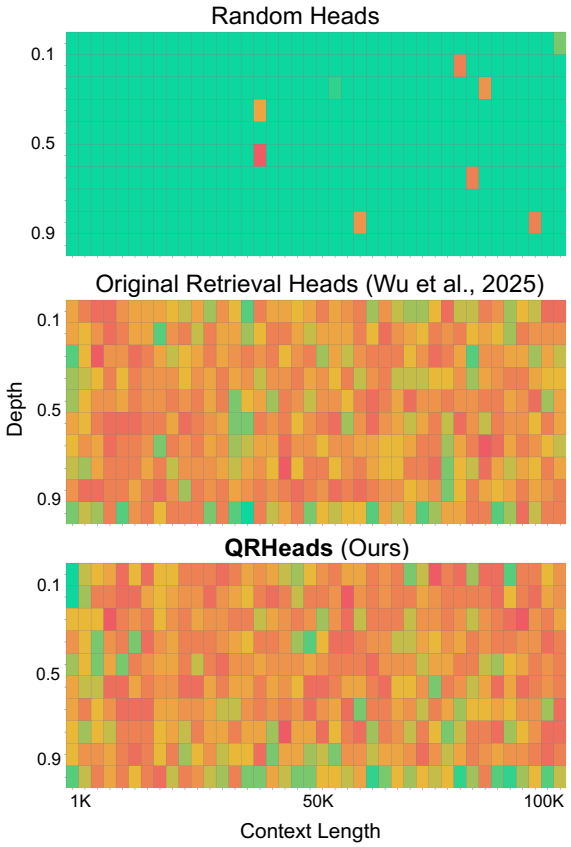

Figure 4: **Top**: Masking 32 random heads of Qwen2.5-7B-Instruct. **Middle**: Masking the top 32 original retrieval heads (Wu et al., 2025b). **Bottom**: Masking the top 32 QRHeads.

C Details about Dual Encoder and Cross Encoder Baselines on BEIR

In §5.3, we report the performance of several traditional retrievers, including dual encoders (Karpukhin et al., 2020) and cross-encoders (Yang and Seo, 2020). We use popular models from the SentenceTransformers library (Reimers and Gurevych, 2019).[5] Specifically, we include the following dual encoders:

- **Contriever**(Izacard et al., 2022): We use the checkpoint from `https://huggingface.co/facebook/contriever-msmarco`, which is fine-tuned on MS MARCO.
- **GTR-T5-Base**(Ni et al., 2022): We use the checkpoint from `https://huggingface.co/sentence-transformers/gtr-t5-base`.

We also include two cross-encoders fine-tuned on MS MARCO:

- **BGE-Reranker-Base**(Xiao et al., 2024b): We use the checkpoint from `https://huggingface.co/BAAI/bge-reranker-base`.
- **MSMARCO-MiniLM**(Reimers and Gurevych, 2019): We use the checkpoint from `https://huggingface.co/cross-encoder/ms-marco-MiniLM-L6-v2`.

[5] `https://sbert.net/`

D Prompt Templates

We provide prompt templates used in our experiments for ICR and QRRETRIEVER in Figure 5 and rankGPT in Figure 6.

E Inference Time Comparison between QRRETRIEVER, ICR, and RankGPT

We compare the latency of QRRETRIEVER with other LLM-based re-rankers (ICR and RankGPT). In Table 7, we report both latency and performance of Llama-3.1-8B and Llama-3.1-70B on NQ dataset. Compared to RankGPT, QRRETRIEVER is

```
{prompt_prefix} Here are some paragraphs:

[1] {Title 1 (if available)}
{Paragraph text 1}

[2] {Title 2 (if available)}
{Paragraph text 2}

...

Please find information that is relevant
to the following query in the paragraphs
above.

Query: {query}{prompt_suffix}
```

Figure 5: Prompt used for ICR and QRRETRIEVER.

```
{prompt_prefix} This is an intelligent
assistant that can rank passages based on
their relevancy to the query.

The following are {N} passages, each
indicated by a numbered identifier [i]. I
can rank them based on their relevance to
the query: "{query}"

[1] {Title 1 (if available)}
{Paragraph text 1}

[2] {Title 2 (if available)}
{Paragraph text 2}

...

The search query is: "{query}". I will
rank the {N} passages above based on their
relevance to the search query. The passages
will be listed in descending order using
identifiers, the most relevant passages
should be listed first and the output format
should be [] > [] > etc, e.g., [1] > [2] >
etc. Be sure to list all {N} ranked passages
and do not explain your ranking until after
the list is done. {prompt_suffix} Ranked
Passages: [
```

Figure 6: Prompt used for rankGPT.

significantly more time-efficient. This is primarily because (1) QRRETRIEVER avoids autoregressive generation, and (2) it does not rely on bubble sort, which requires multiple rounds of generation to compare elements sequentially within each bubble.

F Determine the Number of Heads Used

We select the number of heads based on the sparsity level reported in prior work on retrieval heads (Wu et al., 2025b). In Table 8, we include additional results on BEIR NQ using varying numbers of heads.

	Avg Latency (s/sample)	NQ NDCG@10
	Model: LLama-3.1-8B-Inst	
RankGPT$^{\text{Bubble}}$	14.5	53.7
ICR	2.2	54.0
QRRETRIEVER	2.2	58.6
	Model: Llama-3.1-70B-Inst	
RankGPT$^{\text{Bubble}}$	63.9	58.4
ICR	13.7	57.9
QRRETRIEVER	13.7	62.1

Table 7: Inference time and retrieval performance comparison on NQ.

Model / #Heads	NQ (NDCG@10)
Llama-3.1-8B (#heads = 1024)	
16 (∼1.5%)	58.6
64 (∼6%)	57.4
256 (∼20%)	56.6
Llama-3.1-70B (#heads = 5120)	
32 (∼0.5%)	62.1
128 (∼2%)	61.9
512 (∼10%)	61.0

Table 8: Performance with different numbers of selected heads.

The results show that performance remains consistently strong when using a small, top-ranked subset of heads, but degrades as more heads are included. This trend aligns with the sparsity property observed in retrieval heads.

G License of Datasets

The licenses datasets used in our work include:

- LongMemEval (Wu et al., 2025a) under MIT License.
- Clipper (Pham et al., 2025) under Apache license 2.0.
- NQ (Kwiatkowski et al., 2019) under Creative Commons Attribution Share Alike 3.0.
- BEIR (Thakur et al., 2021) under Creative Commons Attribution Share Alike 4.0 Read on choosealicense.com

H Computational Resources and Model Sizes

We use Llama-3.2 (3B), Llama-3.1 (8B and 70B) (Llama-3 Team, 2024), and Qwen2.5 (7B) (Yang et al., 2024). 8B models were run using a single NVIDIA A100 GPU with 80GB of memory, and 70B models were run using 4 A100 GPUs. All experiments were conducted on A100-based infrastructure.

I Potential Risks of Our Work

N/A. Our work investigates the capabilities of existing language models, without proposing new model architectures or training procedures. While large language models pose well-known risks—including potential misuse, generation of harmful content, and encoding of societal biases—our study does not introduce new risks beyond those already covered in the broader literature. As such, we do not believe any specific risk mitigation measures are necessary for the scope of this work.