EviRerank: Adaptive Evidence Construction for Long-Document LLM Reranking

Minghan Li[1]*, **Eric Gaussier**[2], **Juntao Li**[1], **Guodong Zhou**[1]
[1]School of Computer Science and Technology, Soochow University, Suzhou, China
[2]Univ. Grenoble Alpes, CNRS, Grenoble INP, LIG, 38000 Grenoble, France
{mhli,ljt,gdzhou}@suda.edu.cn Eric.Gaussier@univ-grenoble-alpes.fr

Abstract

Decoder-only LLM rerankers struggle with long documents: inference is costly and relevance signals can be diluted by irrelevant context. Motivated by an attention analysis indicating a consistent degradation trend when non-relevant text is appended, we propose **EviRerank**, an evidence-based long-document reranking framework for decoder-only LLMs. EviRerank (i) scores document blocks with a lightweight selector (BM25, bi-encoder, or cross-encoder), (ii) constructs a compact reranking context under a hard token cap by dynamically budgeting evidence blocks with **Adaptive Evidence Budgeting (AEB)** and adding a global summary cue via **Summary Augmentation (SA)**, and (iii) reranks with a decoder-only LLM. Across TREC DL'19, DL'22, DL'23, and MLDR-zh, EviRerank consistently outperforms full-document LLM reranking and strong block-selection baselines while substantially reducing the required input length. On TREC DL'19, EviRerank reaches up to **0.744** nDCG@10 while substantially reducing reranking latency. Additional RankZephyr-7B experiments confirm transfer to **listwise reranking** under matched context budgets.

1 Introduction

Large language models (LLMs) (Touvron et al., 2023) have become strong rerankers (e.g., RankLLaMA (Ma et al., 2024)) for web search and other retrieval settings, thanks to advances in modeling capacity and generalization. However, real-world documents are often long, and decoder-style LLMs incur rapidly growing inference costs as input length increases. Moreover, when relevant evidence is sparse and scattered, the reranker input may fail to cover the decisive signals, leading to unstable ranking quality.

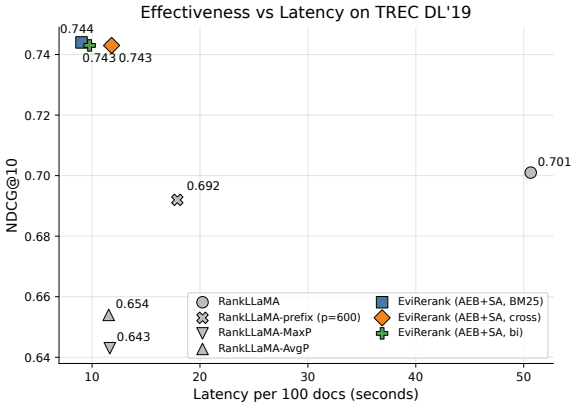


Figure 1: Effectiveness-efficiency trade-off on DL'19. We plot nDCG@10 against reranking latency per 100 documents (top-left better).

A common workaround is to decompose a document into smaller blocks and rerank based on a subset (Li et al., 2023b; Li and Gaussier, 2022). This "block-first" strategy introduces a new question: *how should we construct an evidence context under a hard token cap?* Most existing block-based pipelines (e.g., KeyB (Li et al., 2023b) / IDCM-style (Hofstätter et al., 2021) selection) primarily address *which* blocks to keep, and often rely on an implicit fixed allocation rule (e.g., always taking a fixed number of top blocks or always filling the budget). However, documents differ greatly in information density: some contain one or two highly relevant blocks followed by many weak ones, while others have many moderately relevant blocks. As a result, a fixed allocation wastes tokens on low-utility tail blocks for the former, yet may still fail to allocate enough evidence for the latter.

In this work, we propose **EviRerank**, a lightweight evidence-guided reranking framework for decoder-only LLMs that constructs a compact reranker input under a hard maximum token cap. Given a query-document pair, EviRerank first scores document blocks with a local scorer (e.g., BM25 (Robertson and Zaragoza, 2009), bi-encoder,

*Corresponding author: mhli@suda.edu.cn

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 14772–14792
October 24-29, 2026 ©2026 Association for Computational Linguistics

or cross-encoder), and then performs evidence construction by allocating the capped context budget to block evidence and global cues. Crucially, we introduce Adaptive Evidence Budgeting (AEB), a density-aware budgeting rule that determines how much block evidence to include: instead of always filling the cap, it halts further budget allocation when additional blocks provide low marginal utility, yielding a query-adaptive evidence length. In addition, we incorporate Summary Augmentation (SA) within the same cap to inject global document cues, and isolate its effect with controlled ablations. Experiments on TREC DL'19, DL'22, DL'23, and MLDR-zh show consistent gains and a better accuracy–efficiency trade-off. Additional RankZephyr-7B experiments show that the same evidence-construction layer transfers to listwise LLM reranking under matched context budgets. We focus on document reranking quality, rather than downstream answer generation.

Research questions. We study five questions: **RQ1 Mechanism:** What does attention reveal about long-document evidence construction? (Appendix C) **RQ2 Generalization:** Do gains hold across datasets and across pointwise and listwise LLM reranking? **RQ3 Selectors:** How do BM25/bi-/cross-encoder selectors affect the effectiveness-efficiency trade-off? **RQ4 SA:** Does Summary Augmentation improve effectiveness under the same cap? **RQ5 Efficiency:** What are the latency and training-memory benefits?

Contributions. Our main contributions are:

- **Evidence construction for long-document LLM reranking.** We cast long-document reranking as constructing a compact evidence context under a token cap before LLM scoring. EviRerank is compatible with different block selectors and can be used as a construction layer for both pointwise and listwise LLM rerankers.
- **Adaptive Evidence Budgeting (AEB).** We introduce a model-agnostic, normalized-score ratio stopping rule that adaptively determines how much block evidence to include.
- **Summary Augmentation (SA).** We add a compact summary cue within the same cap and evaluate it with controlled baselines to rule out "just more tokens" effects.
- **Results and analysis.** We show consistent gains on DL'19, DL'22, DL'23, and MLDR-zh, validate transfer to RankZephyr-7B listwise reranking, and provide controlled ablations and efficiency analyses.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 presents EviRerank, including AEB and SA. Section 4 and 5 describe the experimental settings and main results, followed by ablations and efficiency analyses in Section 6.

2 Related Work

2.1 Neural IR Selectors: Cross- and Bi-encoders

Neural IR with PLMs is commonly instantiated as (i) cross-encoders that jointly encode query-text pairs for accurate reranking (Devlin et al., 2019; Nogueira and Cho, 2019), (ii) bi-encoders that learn dense representations for efficient scoring or retrieval (Karpukhin et al., 2020; Reimers and Gurevych, 2019), and (iii) late-interaction models that balance token-level matching and efficiency (Khattab and Zaharia, 2020; Santhanam et al., 2022). In this work, we leverage these paradigms as lightweight local selectors (BM25/bi/cross) to score document blocks before LLM reranking.

2.2 Long-Document Reranking

Long documents challenge Transformer rerankers due to quadratic attention and diluted relevance signals. Efficient/sparse Transformers extend context length (e.g., Longformer (Beltagy et al., 2020), BigBird (Zaheer et al., 2021), and related sparse designs (Child et al., 2019)), and IR-specific variants such as TKL (Hofstätter et al., 2020) and QDS-Transformer (Jiang et al., 2020) further tailor attention patterns. A complementary line segments documents into passages and aggregates signals, using simple pooling (Dai and Callan, 2019) or learned aggregation (e.g., PARADE (Li et al., 2023a)). More recently, block selection reduces noise by identifying salient passages before applying stronger rankers: IDCM (Hofstätter et al., 2021), ICLI (Li and Gaussier, 2022), and KeyB (Li et al., 2023b) report strong results by selecting and concatenating blocks.

Different from prior work that primarily treats block selection as a pre-processing step for BERT-style rerankers or simple concatenation, we study

budget-aware evidence construction tailored to decoder-only LLM rerankers: selecting query-salient evidence blocks, composing them into a compact context under a fixed token budget, and optionally augmenting it with a lightweight summary cue.

2.3 LLM-based Reranking

Finetuned LLM rerankers such as RankLLaMA (Ma et al., 2024) demonstrate strong effectiveness, but long-document reranking remains challenging due to context limits and high inference cost. Listwise LLM rerankers such as RankGPT-style prompting (Sun et al., 2023) and RankZephyr-7B (Pradeep et al., 2023) introduce a related budget tension because a single context window is shared across multiple candidate documents. Recent work also explores progressive adaptation (Zhang et al., 2024) and context compression for efficiency (e.g., PERank (Liu et al., 2024)). In contrast to full-document LLM reranking or aggregation-only pooling, our approach explicitly constructs a compact evidence context before LLM scoring. We further evaluate whether this evidence-construction layer transfers from RankLLaMA-style pointwise reranking to RankZephyr-style listwise reranking.

3 EviRerank: Evidence-Guided LLM Reranking with Adaptive Budgeting

Fig. 2 illustrates **EviRerank**, an evidence-guided reranking framework for long documents. EviRerank decomposes each document into blocks, scores blocks with a lightweight local scorer, and then composes a compact LLM input for a decoder reranker. Two components are optional and can be toggled independently: (i) **Adaptive Evidence Budgeting (AEB)** for information-density-aware early stopping, and (ii) **Summary Augmentation (SA)** that injects a short, query-agnostic summary cue under the same total budget.

3.1 Passage Segmentation

We segment each document into blocks using the CogLTX decomposition method (Ding et al., 2020), which favors strong punctuation boundaries and uses dynamic programming to ensure each block length is bounded. Following Li et al. (2023b), we set the maximum block length to B=63. For consistency with the final reranker, block segmentation uses the reranker tokenizer (Llama2/Llama3 accordingly), and we extend the punctuation set for Chinese to better detect sentence boundaries.

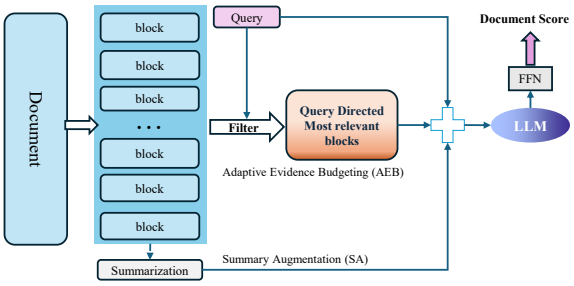


Figure 2: Architecture of **EviRerank**. The framework composes the reranker input from query-focused evidence blocks, optionally augmented with a compact query-agnostic summary augmentation (SA). Adaptive Evidence Budgeting (AEB) enables early stopping under the same budget.

3.2 Local Block Scoring

After splitting a long document into blocks, EviRerank scores each block locally and then performs global reranking with a decoder LLM. The local scoring stage is lightweight (orders of magnitude cheaper than LLM inference) and *model-agnostic*: any standard IR scorer can be plugged in. In our experiments, we instantiate three widely used options; full formulas and details are provided in Appendix B.1.

BM25 (term matching). We adopt the standard BM25 with (k_1, b) and smoothed IDF (computed with `scikit-learn` (Pedregosa et al., 2011)). BM25 provides a strong term-matching baseline and is robust for long documents. For Chinese texts we apply standard word segmentation before scoring.

Cross-encoder (interaction). A pretrained encoder (e.g., BERT (Devlin et al., 2019)) takes the concatenated `[query; block]` as input; the `[CLS]` representation is fed to a linear head to yield a relevance score. Because each block is short, per-block inference is still efficient while retaining fine-grained token-level interactions.

Bi-encoder (dense retrieval). A shared encoder maps queries and blocks to vectors; local relevance is computed by a dot product (or cosine). Block embeddings can be precomputed offline, making scoring efficient at runtime while capturing semantic similarity beyond lexical overlap.

3.3 Composing the LLM Input: Evidence Blocks with a Summary Cue

Setup. Given a query Q and a document segmented into blocks $\{b_i\}_{i=1}^{n}$, a local scorer (BM25/bi-encoder/cross-encoder; Sec. 3.2) pro-

duces block relevance scores s_i and block lengths $L(b_i)$. Let the decoder reranker token cap for document-side content be B. We allocate this cap between evidence blocks and the summary cue: $B = B_K + B_S$, where B_K is the maximum budget for evidence blocks K, and B_S is the maximum budget for the summary cue S (with $B_S = 0$ when SA is disabled). Let k_s be the number of blocks used to form the summary cue.

Step 1: Adaptive Evidence Budgeting (AEB) via information-density-aware early stopping. Naive packing fills the token budget by appending blocks, but long documents often exhibit highly skewed *information density* (a few decisive blocks followed by a low-utility tail). To avoid diluting strong evidence with weak blocks, we introduce **AEB**, which *early-stops* evidence packing when the marginal relevance of newly considered blocks falls below a threshold, producing a query-adaptive evidence length that does not necessarily fill the cap B_K.

Score normalization. Raw block scores s_i can have heterogeneous scales across local scorers (e.g., BM25 vs. cross-encoder logits), making ratio-based stopping brittle. We transform scores into a bounded, monotonic scale $\tilde{s}_i$ before early stopping:

$$\tilde{s}_i = g(s_i), \tag{1}$$

where $g(\cdot)$ is either identity (NONE) or Min-Max normalization:

$$g(s_i) = \begin{cases} s_i, & \text{NONE}, \\ \dfrac{s_i - s_{\min}}{s_{\max} - s_{\min} + \epsilon}, & \text{MIN-MAX}, \end{cases} \tag{2}$$

with $s_{\min} = \min_j s_j$, $s_{\max} = \max_j s_j$, and a small constant ϵ (set to 10^{-12}).

Dynamic selection with ratio-based stopping. Let π sort blocks by descending normalized score $\tilde{s}$: $\tilde{s}_{\pi_1} \geq \cdots \geq \tilde{s}_{\pi_n}$. We scan blocks in this order and greedily add b_{π_j} into K if it fits the evidence budget ($T + L(b_{\pi_j}) \leq B_K$); otherwise, we stop (i.e., blocks are packed atomically rather than split).

To avoid selecting low-density tail blocks, we apply a ratio-based early stopping rule after selecting at least m blocks:

$$\text{stop if } |K| \geq m \ \wedge \ \tilde{s}_{\pi_j} < \rho\, \tilde{s}_{\pi_1}. \tag{3}$$

Here $\tilde{s}_{\pi_1}$ is the best normalized block score and $\tilde{s}_{\pi_j}$ is the current block score in the sorted order. The hyperparameter $\rho \in [0, 1]$ controls how aggressively we truncate the long tail: once the score drops below a fixed fraction of the best block, we stop adding further blocks even if budget remains. Setting $\rho = 0$ disables ratio-based stopping, reducing the procedure to budget-only selection.

Finally, we restore the original document order of selected blocks in K. As a safety measure, we apply a hard truncation after composing the final input (Step 3) to ensure the total document-side content does not exceed B tokens, accounting for tokenization and prefix tokens.

Step 2: Summary Augmentation (SA) as a query-agnostic cue. We build a compact summary cue from blocks. We obtain block embeddings $\{e_i\}_{i=1}^{n}$ from the bi-encoder (shared with the selector by default) and cache them offline for efficiency, then compute the centroid

$$\hat{c} \ = \ \frac{\sum_i e_i}{\|\sum_i e_i\|}, \tag{4}$$

score each block by its similarity to the centroid

$$s_i^{\text{sum}} \ = \ e_i \cdot \hat{c}, \tag{5}$$

and select the top-k_s blocks by s_i^{sum} (preserving original order) to form a short summary S.

Step 3: Compose and score with the decoder reranker. We first compose the document-side content as

$$\tilde{D} \ = \ K \parallel S, \tag{6}$$

and then enforce the hard token cap by truncation:

$$D' \ = \ \text{Trunc}_B\big(\tilde{D}\big), \tag{7}$$

where K is constructed under the evidence budget B_K and S under the summary budget B_S, with $B = B_K + B_S$.

Pointwise reranking. For the RankLLaMA-style pointwise instantiation used in our main experiments, we format the decoder-only reranker input as

$$\texttt{input} = \text{“query: \{Q\} document: \{D'\}”}. \tag{8}$$

The final relevance score is produced from the last hidden state:

$$\text{RSV}(Q, D) \ = \ \text{Linear}\big(\text{Decoder}(\texttt{input})[-1]\big). \tag{9}$$

Listwise use. The evidence construction in Eqs. 6–7 is independent of the downstream scoring interface. For a candidate set $\{D_i\}_{i=1}^{N}$, we apply the same construction independently to each document to obtain compact representations $\{D_i'\}_{i=1}^{N}$. A listwise reranker can then consume these representations in place of full-document truncations. In Section 5.2, we evaluate this setting with RankZephyr-7B while keeping its native listwise prompt and sliding-window reranking procedure unchanged.

Rationale. AEB adaptively allocates the evidence budget per query-document pair, avoiding low-utility tail blocks. SA adds a compact query-agnostic global cue that complements the selected evidence blocks. Both are lightweight: scoring operates on short blocks, and SA is a top-k selection over cached block embeddings.

4 Experimental Settings

We evaluate **EviRerank** on multiple long-document reranking benchmarks and compare it with strong baselines, mainly targeting **RQ2**-**RQ5** introduced in Section 1.

4.1 Datasets

We use four datasets emphasizing long or full-article documents:

- **TREC DL 2019** (document reranking) (Craswell et al., 2020): MS MARCO v1-based document reranking with human relevance judgments.
- **TREC DL 2022** (document reranking): MS MARCO v2-based document reranking, used as an additional held-out validation benchmark.
- **TREC DL 2023** (document reranking): built on MS MARCO v2 with web-page style documents.
- **MLDR-zh** (Chen et al., 2024): the Chinese subset of MLDR, sourced from Wikipedia and Wudao (Yuan et al., 2021).

Dataset statistics are reported in Appendix B.2.

4.2 Baselines

We compare against competitive first-stage and reranking systems. Baseline sets align with prior work while remaining consistent across datasets where applicable. Most baselines are described in Appendix B.3.

Controlled RankLLaMA baselines. On DL'19, we additionally include a matched-budget RankLLaMA-prefix control, which uses the same pointwise reranker but truncates each document to its first $p_{\max} = 600$ document-side tokens.

RankLLaMA-MaxP / AvgP To align with block-based designs, we segment each document into blocks, compute per-block scores with RankLLaMA, and then aggregate: MaxP takes the maximum block score and AvgP averages block scores. These serve as strong aggregation-only counterparts to evidence selection and composition.

RankZephyr-7B. To evaluate whether EviRerank transfers beyond pointwise RankLLaMA-style scoring, we additionally evaluate RankZephyr-7B (Pradeep et al., 2023), a listwise LLM reranker. Within each document budget, full-document truncation and EviRerank use the same candidate set, context limit, prompting protocol, listwise window/stride, and decoding settings. Detailed settings are reported in Table 8(a) of Appendix A.2.

4.3 Experimental Design and Implementation Details

Variants of EviRerank. We evaluate **EviRerank** as an LLM-augmented reranker with fine-tuning. To ensure fair comparison with RankLLaMA (LLaMA2-7B[1] backbone), we fine-tune EviRerank using the same backbone. EviRerank adopts local block scorers as modular components:

$$\text{EviRerank}_{\text{BM25}},\quad \text{EviRerank}_{\text{bi}},\quad \text{EviRerank}_{\text{cross}},$$

corresponding to BM25, bi-encoder, and cross-encoder selectors. We by default enable **Adaptive Evidence Budgeting (AEB)** (dynamic stopping) and **Summary Augmentation (SA)**.

Implementation summary. We use PyTorch 2.4 with CUDA 11.8 on a single NVIDIA A100 (40GB). We fine-tune with LoRA (Hu et al., 2021) while freezing the backbone, using pairwise hinge loss (Li et al., 2023b) and AdamW with warmup-decay in FP16. Default LoRA hyperparameters are r=32, α=64, learning rate 5×10^{-5}, batch size 2, and gradient accumulation steps 8. All models train

[1] https://huggingface.co/meta-llama/Llama-2-7b-hf

for one epoch on the same triplets. RankLLaMA uses gradient checkpointing (for longer inputs) and batch size 1; otherwise hyperparameters mirror EviRerank. All models are evaluated as rerankers over the same first-stage candidates (official runs for DL tracks; for MLDR-zh, BM25 top-k).

Training data and evaluation metrics are in Appendix B.4.

RankZephyr evaluation protocol. For the listwise evaluation, we use the `castorini/rank_zephyr_7b_v1_full` checkpoint with the official RankLLM prompt format and rerank the official top-100 candidates under a 4096-token context window. For EviRerank, each candidate document is independently converted to its compact evidence representation using the cross-encoder selector with AEB+SA before listwise reranking. The exact per-document budgets and window/stride settings are reported in Table 8(a) of Appendix A.2.

Input length and packing. Queries are truncated to 32 tokens following (Ma et al., 2024). Unless otherwise specified, we enforce a hard maximum cap of $p_{\max}$=600 document-side tokens. Under this cap, we compose the reranker context from (i) query-focused evidence blocks and (ii) a lightweight summary cue (**SA**), while never exceeding the cap. By default, we allocate up to 480 tokens to evidence blocks and up to 120 tokens to the summary cue (within the same $p_{\max}$ cap). With **AEB**, evidence blocks are allocated dynamically: selection may terminate early when additional blocks provide low marginal utility, yielding a shorter query-adaptive evidence length under the same hard cap. For controlled ablations, we remove **SA** and reallocate its token budget to evidence blocks, while keeping the same $p_{\max}$=600 cap; we also compare against a 3-block random summary cue under the same cap (Table 9). In contrast, the full-document RankLLaMA baseline consumes up to 4,096 document-side tokens. The DL'19 matched-budget RankLLaMA-prefix control instead truncates each document to the same $p_{\max} = 600$ document-side cap.

Block selectors and pretrained encoders. We use language-specific cross-encoders and a multilingual bi-encoder as selectors in EviRerank (Table 1); Unless otherwise specified, the same bi-encoder is used for both block selection and summary cue construction.

Encoder	Lang	HF checkpoint
Cross	EN	`cross-encoder/ms-marco-MiniLM-L-6-v2`
Cross	ZH	`BAAI/bge-reranker-base`
Bi	EN/ZH	`intfloat/multilingual-e5-small`

Table 1: Encoders used by EviRerank (HF checkpoints).

AEB hyperparameters. AEB applies ratio-based early stopping with threshold ρ (Eq. 3). For score normalization, we use NONE for BM25 and Min-Max normalization for neural selectors, computed over blocks *within each document*. This design reflects that BM25 scores are derived from a fixed lexical matching function and are typically more comparable across blocks for the same query, whereas neural relevance scores can vary in scale and dynamic range across queries/documents, making the ratio criterion sensitive without normalization. We select ρ on the dev set by sweeping $\{0.05, 0.10, 0.15, \ldots, 0.65\}$ and then fix it for test. We also tune the minimum number of kept blocks m (Eq. 3) on the dev set by sweeping $\{2, 3, 4, 5, 6\}$, while other AEB parameters follow the default setting (e.g., maximum kept blocks).

BM25 settings. For English, we use Anserini defaults (k_1=0.9, b=0.4). For MLDR-zh, we apply Chinese word segmentation with jieba[2]; other settings follow English. For EviRerank$_{\text{BM25}}$ and KeyB models, IDF statistics are computed with `scikit-learn` on each corpus.

5 Experimental Results

5.1 Dataset-wise Effectiveness Results

Tables 2–4 report the main pointwise results on DL'19, DL'23, and MLDR-zh. We additionally evaluate DL'22 as a held-out validation benchmark; its complete results are reported in Table 7 of Appendix A.1. Significance is assessed by a paired two-sided t-test ($p \leq 0.05$) on per-query metric scores. Superscripts † and ‡ denote statistically significant improvements over RankLLaMA and KeyB(BERT)$_{\text{BinB}}$, respectively.

TREC DL'19. The best configuration is EviRerank (AEB+SA)$_{\text{cross}}$ with **nDCG@10** 0.743 and **MAP** 0.307 (Table 2). This corresponds to clear improvements over the full-document LLM reranker RankLLaMA (0.701/0.288) and the strongest BERT-based block-selection baseline KeyB(BERT)$_{\text{BinB}}$ (0.697/0.283). Across selec-

[2] https://github.com/fxsjy/jieba

Table 2: Results on TREC DL'19 document reranking, compared with sparse-attention rerankers and prior block-selection methods. Best results are in **bold**. † and ‡ indicate statistically significant improvements under a paired two-sided t-test at $p < 0.05$ over RankLLaMA and KeyB(BERT)$_{\text{BinB}}$, respectively.

TREC 2019 DL Track Document Reranking		
Model	NDCG@10	MAP
Baseline models		
BM25	0.488	0.234
TKL	0.644	0.277
PARADE	0.655	0.280
Sparse attention models		
Sparse-Transformer	0.634	0.257
Longformer-QA	0.627	0.255
Transformer-XH	0.646	0.256
QDS-Transformer	0.667	0.278
Select blocks models		
IDCM	0.679	0.273
KeyB(PARADE5)$_{BM25}$	0.672	0.280
KeyB(PARADE5)$_{BinB}$	0.676	0.277
KeyB(PARADE5)$_{BinB2}$	0.678	0.279
KeyB(BERT)$_{BM25}$	0.683	0.281
KeyB(BERT)$_{BinB}$	0.697	0.283
LLM		
RankLLaMA	0.701	0.288
RankLLaMA-prefix (p=600)	0.692	0.281
RankLLaMA-MaxP	0.643	0.269
RankLLaMA-AvgP	0.654	0.264
Ours: EviRerank (AEB+SA)		
EviRerank$_{\text{BM25}}$	**0.744**†‡	0.302‡
EviRerank$_{\text{cross}}$	0.743†‡	**0.307**‡
EviRerank$_{\text{bi}}$	0.743†‡	0.300‡

tors (BM25/bi/cross), EviRerank consistently outperforms RankLLaMA-MaxP/AvgP, showing that simple score pooling is inferior to explicitly selecting and composing evidence before reranking. Importantly, the gain is not explained by input shortening alone: the matched-budget RankLLaMA-prefix control reaches only 0.692 nDCG@10 and 0.281 MAP under the same $p_{\max} = 600$ document-side cap. Moreover, all EviRerank (AEB+SA) variants are statistically significant ($p \leq 0.05$) over RankLLaMA (†) and KeyB(BERT)$_{\text{BinB}}$ (‡) on nDCG@10, and they are also significant over KeyB(BERT)$_{\text{BinB}}$ on MAP (Table 2).

TREC DL'23. On DL'23, the strongest result is obtained by EviRerank (AEB+SA)$_{\text{cross}}$, reaching **nDCG@10** 0.475 and **MAP** 0.157 (Table 3). Relative to RankLLaMA (0.386/0.133) and KeyB(BERT)$_{\text{BinB}}$ (0.364/0.128), this yields substantial gains on both metrics. Again, evidence construction (selection + composition) dominates aggregation-only baselines (MaxP/AvgP), reinforcing that allocating the input budget to high-utility evidence is more effective than pooling over block scores. All EviRerank (AEB+SA) variants yield statistically significant gains ($p \leq 0.05$) over both RankLLaMA (†) and KeyB(BERT)$_{\text{BinB}}$ (‡) on both metrics (Table 3).

Table 3: Results on TREC DL'23 document reranking. Best results are in **bold**.† and ‡ indicate statistically significant improvements under a paired two-sided t-test at $p < 0.05$ over RankLLaMA and KeyB(BERT)$_{\text{BinB}}$, respectively.

TREC 2023 DL Track Document Reranking		
Model	NDCG@10	MAP
Baseline models		
BM25	0.295	0.105
KeyB(BERT)$_{BM25}$	0.352	0.123
KeyB(BERT)$_{BinB}$	0.364	0.128
LLM		
RankLLaMA	0.386	0.133
RankLLaMA-MaxP	0.341	0.112
RankLLaMA-AvgP	0.349	0.117
Ours: EviRerank (AEB+SA)		
EviRerank$_{\text{BM25}}$	0.457†‡	0.154†‡
EviRerank$_{\text{cross}}$	**0.475**†‡	**0.157**†‡
EviRerank$_{\text{bi}}$	0.465†‡	0.156†‡

Table 4: Results on MLDR-zh. Best results are in **bold**. † and ‡ indicate statistically significant improvements under a paired two-sided t-test at $p < 0.05$ over RankLLaMA and KeyB(BERT)$_{\text{BinB}}$, respectively.

Model	P@1	MAP	NDCG@8
Baseline models			
BM25	0.201	0.259	0.277
BERT based models			
KeyB(BERT)$_{BM25}$	0.423	0.586	0.685
KeyB(BERT)$_{BinB}$	0.804†	0.876†	0.907†
LLM			
RankLLaMA	0.649	0.755	0.814
RankLLaMA-MaxP	0.374	0.554	0.661
RankLLaMA-AvgP	0.228	0.433	0.567
Ours: EviRerank (AEB+SA)			
EviRerank$_{\text{BM25}}$	0.909†‡	0.939†‡	0.954†‡
EviRerank$_{\text{cross}}$	0.945†‡	0.964†‡	0.973†‡
EviRerank$_{\text{bi}}$	**0.951**†‡	**0.968**†‡	**0.976**†‡

TREC DL'22. As an additional held-out MS MARCO v2 document-ranking benchmark, DL'22 exhibits the same overall trend. Without DL'22-specific tuning, EviRerank$_{\text{cross}}$ achieves 0.4488 nDCG@10 and 0.1151 MAP, compared with 0.3845/0.1019 for full-document RankLLaMA. Complete results are reported in Table 7 of Appendix A.1.

MLDR-zh. On MLDR-zh, the best configuration is EviRerank (AEB+SA)$_{\text{bi}}$, achieving the strongest scores across the reported metrics (Table 4). Compared with RankLLaMA and KeyB(BERT)$_{\text{BinB}}$,

EviRerank yields large improvements, highlighting its robustness for very long Chinese documents. Notably, the bi-encoder selector is particularly effective in this setting, likely because it provides a strong semantic prior under domain- and language-matched encoders. All EviRerank (AEB+SA) variants are statistically significant ($p \leq 0.05$) over RankLLaMA (†) and KeyB(BERT)$_{\text{BinB}}$ (‡) on all reported metrics (Table 4). Interestingly, KeyB(BERT)$_{\text{BinB}}$ also significantly outperforms RankLLaMA on MLDR-zh (†), suggesting that strong BERT selectors can remain competitive for very long Chinese documents.

Cross-dataset takeaway (RQ2). EviRerank consistently improves over the controlled RankLLaMA-style baselines across DL'19, DL'22, DL'23, and MLDR-zh (Tables 2–4 and Table 7). Selector choice remains collection-dependent: BM25/cross-encoder variants are strongest on MS MARCO-style web documents, while the bi-encoder is strongest on MLDR-zh. Overall, evidence construction remains effective across datasets with different document-length and language profiles.

5.2 Generality to Listwise LLM Reranking

To test whether EviRerank's gains are tied to the RankLLaMA-style pointwise scoring architecture, we apply the same evidence construction independently to every candidate document before RankZephyr-7B listwise reranking. Unlike Eq. 9, no document-level linear scoring head is used; the resulting compact document representations are instead passed to RankZephyr's native listwise prompt and sliding-window reranking procedure. Figure 3 illustrates this interface, and Table 5 reports the matched-budget listwise reranking results.

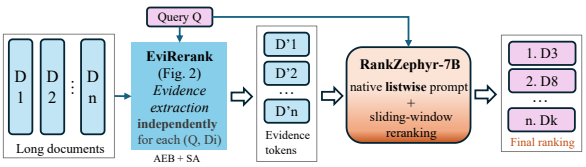


Figure 3: Applying EviRerank to RankZephyr-style listwise reranking. EviRerank independently constructs a compact representation for each candidate document, after which RankZephyr applies its native sliding-window listwise reranking protocol.

Results. EviRerank provides the clearest gains under tighter per-document budgets. At $p = 300$, it improves RankZephyr from 0.6768 to 0.7263 nDCG@10 on DL'19 and from 0.4271 to 0.4732 on DL'23, with corresponding MAP gains. The same overall pattern also holds at $p = 150$.

At $p = 600$, however, the improvement largely saturates: nDCG@10 changes only marginally and MAP is essentially tied. This suggests that evidence construction is most useful when the per-document context budget is constrained, whereas its marginal benefit decreases once substantially more evidence can already be accommodated.

Diagnostic controls. Additional matched controls in Table 8(b) of Appendix A.2 show that the improvement is not explained by arbitrary non-prefix block replacement: query-focused evidence is necessary, and AEB+SA performs best.

Table 5: RankZephyr-7B listwise validation on TREC DL'19 and DL'23 under matched candidate sets, context budgets, and listwise protocols. Within each budget p, bold denotes the better result. Superscript * marks a significant improvement over matched full-document truncation ($p < 0.05$, paired two-sided t-test).

Dataset	p	Method	nDCG@10	MAP
DL'19	150	Full-doc trunc.	0.6494	0.2691
		EviRerank	**0.6859**	**0.2951***
DL'19	300	Full-doc trunc.	0.6768	0.2769
		EviRerank	**0.7263***	**0.2910**
DL'19	600	Full-doc trunc.	0.6306	**0.2660**
		EviRerank	**0.6349**	0.2656
DL'23	150	Full-doc trunc.	0.4119	0.1451
		EviRerank	**0.4441**	**0.1504**
DL'23	300	Full-doc trunc.	0.4271	0.1376
		EviRerank	**0.4732***	**0.1516***
DL'23	600	Full-doc trunc.	0.3994	**0.1322**
		EviRerank	**0.4012**	0.1318

5.3 Selector Choice and Effectiveness-Efficiency Trade-off (RQ3)

Which selector to use. The optimal selector depends on the collection:

- **Cross-encoder** yields the highest effectiveness on MS MARCO-style web documents (DL'19/DL'23), benefiting from fine-grained token interactions, at the cost of higher online scoring.
- **Bi-encoder** provides a strong accuracy-efficiency balance when the encoder matches the language/domain (MLDR-zh). With cached block embeddings, query-time scoring is efficient.

- **BM25** remains a strong efficiency-oriented option and is robust under lexical matching, especially when dense encoders are mismatched or unavailable.

5.4 Does Summary Augmentation help? (RQ4)

Summary Augmentation (SA) adds a compact, query-agnostic global cue under the same total document-side cap. The controlled factorial experiments in Section 6 show that SA improves effectiveness on both DL'19 and DL'23, while its combination with AEB yields the strongest results. A focused comparison against random-summary and no-summary controls is reported in Appendix A.3.

5.5 Efficiency (RQ5)

5.5.1 Inference Speed

We measure end-to-end reranking latency under identical inference settings and report the wall-clock time required to rerank 100 candidate documents per query. The reported latency includes online block scoring, evidence construction, tokenization, and final LLM reranking. Fig. 1 plots nDCG@10 against latency.

Observations. Full-document RankLLaMA is substantially slower (50.65s) while reaching 0.701 nDCG@10. EviRerank (AEB+SA) reaches 0.743–0.744 nDCG@10 at only 9.02–11.81s. Importantly, the matched-budget RankLLaMA-prefix control requires 17.90s but reaches only 0.692 nDCG@10, showing that the gain cannot be explained by shorter input alone. MaxP/AvgP are similarly efficient but substantially less effective (0.643–0.654 nDCG@10).

5.5.2 Training Memory Requirements

Appendix A.5 shows the training memory requirement comparisons, showing EviRerank offers a practical accuracy-resource trade-off for long-document reranking.

6 Ablation Study

6.1 Factorial Ablation: AEB and SA

We conduct the same 2×2 factorial ablation on DL'19 and DL'23 using the cross-encoder selector, isolating (i) AEB and (ii) SA. All runs use the same reranker, candidate set, hard $p_{\max}$=600 document-side cap, score normalization, and random seed; only AEB and SA are toggled.

Table 6 shows a consistent pattern across the two benchmarks. On DL'19, AEB improves nDCG@10 while reducing low-utility context, SA provides a larger effectiveness gain, and their combination performs best. The same ordering holds on DL'23: both components improve over the No-AEB/No-SA setting and AEB+SA again achieves the strongest result. Additional token-utilization and summary-control diagnostics are reported in Appendix A.4 and Appendix A.3.

Dataset	Setting	nDCG@10	MAP
DL'19	No-AEB/No-SA	0.728	0.305
DL'19	AEB only	0.735	0.304
DL'19	SA only	0.739	0.305
DL'19	AEB+SA	**0.743**	**0.307**
DL'23	No-AEB/No-SA	0.465	0.155
DL'23	AEB only	0.470	0.156
DL'23	SA only	0.471	0.156
DL'23	AEB+SA	**0.475**	**0.157**

Table 6: 2×2 factorial ablation of AEB and SA on DL'19 and DL'23 using the cross-encoder selector.

7 Conclusion

We presented **EviRerank**, an evidence-guided framework for long-document reranking that constructs a compact document representation under a hard token cap before LLM scoring. Its **Adaptive Evidence Budgeting (AEB)** dynamically limits low-utility tail evidence, while **Summary Augmentation (SA)** adds a compact global cue under the same cap.

Across TREC DL'19, DL'22, DL'23, and MLDR-zh, EviRerank consistently improves over controlled full-document RankLLaMA-style reranking and strong block-selection baselines. Additional RankZephyr-7B experiments show that the same evidence-construction layer transfers to listwise reranking, with the largest gains under constrained per-document budgets. Together, the results show that long-document LLM reranking depends not only on which evidence to select, but also on how much evidence to retain under a limited context budget.

Limitations

First, SA is lightweight and query-agnostic; richer query-conditioned or multi-facet global cues may provide additional benefits for documents with diverse intents. Second, although we evaluate four document-ranking benchmarks and validate both RankLLaMA-style pointwise and RankZephyr-style listwise reranking, the experiments do not

cover all modern reranker architectures or retrieval domains. Finally, this work focuses on document reranking quality under a hard token budget. Evaluating whether the constructed evidence further improves downstream RAG or QA generation is complementary and left for future work.

Acknowledgments

This work was partly funded by Institut Universitaire de France, IUF, and MIAI Cluster, ANR-23-IACL-0006. This work was also supported by the National Natural Science Foundation of China (No. 62376178).

References

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *Preprint*, arXiv:2004.05150.

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*, pages 2318–2335.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *CoRR*, abs/1904.10509.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. 2019. What does bert look at? an analysis of bert's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M Voorhees. 2020. Overview of the trec 2019 deep learning track. *arXiv preprint arXiv:2003.07820*.

Zhuyun Dai and Jamie Callan. 2019. Deeper text understanding for ir with contextual neural language modeling. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 985–988.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Ming Ding, Chang Zhou, Hongxia Yang, and Jie Tang. 2020. Cogltx: Applying bert to long texts. *Advances in Neural Information Processing Systems*, 33:12792–12804.

Sebastian Hofstätter, Bhaskar Mitra, Hamed Zamani, Nick Craswell, and Allan Hanbury. 2021. Intra-document cascading: Learning to select passages for neural document ranking. In *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, pages 1349–1358.

Sebastian Hofstätter, Hamed Zamani, Bhaskar Mitra, Nick Craswell, and Allan Hanbury. 2020. Local self-attention over long text for efficient document retrieval. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2021–2024.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Jyun-Yu Jiang, Chenyan Xiong, Chia-Jung Lee, and Wei Wang. 2020. Long document ranking with query-directed sparse transformer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 4594–4605.

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.

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.

Canjia Li, Andrew Yates, Sean MacAvaney, Ben He, and Yingfei Sun. 2020. Parade: Passage representation aggregation for document reranking. *arXiv preprint arXiv:2008.09093*.

Canjia Li, Andrew Yates, Sean MacAvaney, Ben He, and Yingfei Sun. 2023a. Parade: Passage representation aggregation fordocument reranking. *ACM Transactions on Information Systems*, 42(2):1–26.

Minghan Li and Eric Gaussier. 2022. Bert-based dense intra-ranking and contextualized late interaction via multi-task learning for long document retrieval. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2347–2352.

Minghan Li, Diana Nicoleta Popa, Johan Chagnon, Yagmur Gizem Cinar, and Eric Gaussier. 2023b. The power of selecting key blocks with local pre-ranking

for long document information retrieval. *ACM Transactions on Information Systems*, 41(3):1–35.

Qi Liu, Bo Wang, Nan Wang, and Jiaxin Mao. 2024. Leveraging passage embeddings for efficient listwise reranking with large language models. *arXiv preprint arXiv:2406.14848*.

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*, pages 2421–2425.

Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, 12:2825–2830.

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

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 and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Stephen Robertson and Hugo Zaragoza. 2009. *The probabilistic relevance framework: BM25 and beyond*, volume 4. Now Publishers Inc.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2022. Colbertv2: Effective and efficient retrieval via lightweight late interaction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3715–3734.

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*, pages 14918–14937.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Jesse Vig. 2019. A multiscale visualization of attention in the transformer model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 37–42, Florence, Italy. Association for Computational Linguistics.

Peilin Yang, Hui Fang, and Jimmy Lin. 2018. Anserini: Reproducible ranking baselines using lucene. *Journal of Data and Information Quality (JDIQ)*, 10(4):1–20.

Sha Yuan, Hanyu Zhao, Zhengxiao Du, Ming Ding, Xiao Liu, Yukuo Cen, Xu Zou, Zhilin Yang, and Jie Tang. 2021. Wudaocorpora: A super large-scale chinese corpora for pre-training language models. *AI Open*, 2:65–68.

Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2021. Big bird: Transformers for longer sequences. *Preprint*, arXiv:2007.14062.

Longhui Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, Meishan Zhang, and Min Zhang. 2024. A two-stage adaptation of large language models for text ranking. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 11880–11891.

Chen Zhao, Chenyan Xiong, Corby Rosset, Xia Song, Paul Bennett, and Saurabh Tiwary. 2020. Transformer-xh: Multi-evidence reasoning with extra hop attention. In *International Conference on Learning Representations*.

A Additional Experimental Results and Diagnostics

A.1 Additional DL'22 Validation

We additionally evaluate EviRerank on TREC DL'22. To keep DL'22 as a held-out validation benchmark, we reuse the hyperparameters selected on DL'23 and perform no DL'22-specific tuning. Table 7 reports the complete results.

A.2 RankZephyr Protocol and Diagnostic Controls

All RankZephyr experiments rerank the same official top-100 candidate set under a 4096-token context using the official RankLLM prompt format. Full-document truncation and EviRerank use identical listwise settings within each document budget. Table 8(a) summarizes the budget-specific listwise settings, while Table 8(b) reports the matched diagnostic controls on DL'19 at $p = 300$.

Method	nDCG@10	MAP
BM25	0.2990	0.0800
RankLLaMA	0.3845	0.1019
RankLLaMA-MaxP	0.2248	0.0707
RankLLaMA-AvgP	0.3430	0.0984
EviRerank$_{\text{BM25}}$	0.4307	0.1119
EviRerank$_{\text{cross}}$	**0.4488**	**0.1151**
EviRerank$_{\text{bi}}$	0.4412	0.1147

Table 7: Additional validation on TREC DL'22 document reranking. No DL'22-specific hyperparameter tuning is performed.

(a) Listwise protocol

p	Window/stride	Evidence/SA budget
150	20/10	120/30
300	12/6	240/60
600	6/3	480/120

(b) Diagnostic controls on DL'19 at $p = 300$

Construction	nDCG@10	MAP
Full-document truncation	0.6768	0.2769
Random blocks	0.6392	0.2584
Fixed evidence (cross, no AEB/SA)	0.7133	0.2861
EviRerank (cross + AEB + SA)	**0.7263**	**0.2910**

Table 8: RankZephyr-7B protocol and diagnostic controls. Panel (a) reports the listwise settings for each per-document budget; panel (b) compares evidence-construction variants on DL'19 at $p = 300$ under the same candidate set and listwise protocol.

A.3 Summary Augmentation Diagnostic

We compare the default SA cue against a random-block summary and a no-summary control under the same maximum document-side cap. Table 9 reports the controlled comparison.

Variant	nDCG@10
EviRerank (SA, default)	**0.739**
Random blocks summary	0.731
EviRerank (no SA)	0.728

Table 9: Summary augmentation diagnostic on DL'19 under the same maximum document-side cap.

The default SA cue performs best, indicating that its benefit cannot be explained by simply allocating the same budget to additional query-focused or randomly selected blocks.

A.4 Efficiency and Budget Diagnostics

Token utilization. We report the average document-side token usage of the 2×2 ablation settings on DL'19. The measured length is the tokenized document-side content after evidence composition and truncation, before padding.

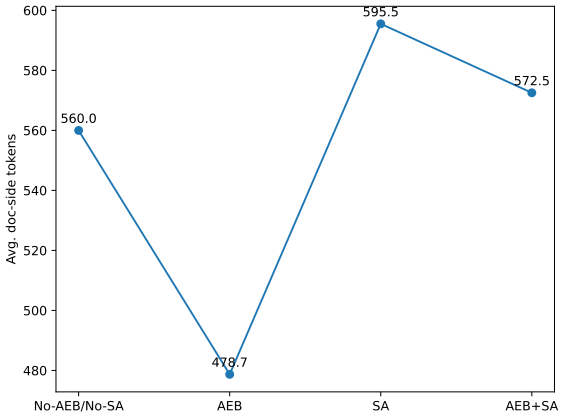


Figure 4: Average document-side token usage of the 2×2 ablations on DL'19. Values are annotated at each point.

As shown in Fig. 4, AEB reduces the average document-side length from 560.0 to 478.7 tokens (-14.6%) without SA. When SA is enabled, the reduction is smaller (595.5 to 572.5 tokens, -3.9%), because SA already consumes part of the shared token budget.

Budget sensitivity. We additionally vary the maximum document-side token budget while keeping the other settings fixed. Table 10 reports the results.

$p_{\max}$	nDCG@10	MAP
400	0.731	0.306
600	0.743	0.307
800	0.744	0.307

Table 10: Sensitivity to the document-side token cap on DL'19 using EviRerank$_{\text{cross}}$. Other settings are fixed.

As shown in Table 10, performance improves from 400 to 600 tokens and then largely saturates at 800 tokens, supporting $p_{\max} = 600$ as a practical default.

A.5 Training Memory Comparison

Fig. 5 reports peak GPU memory usage during training on an NVIDIA H20 (96 GB). Full-document RankLLaMA, which consumes up to 4096 tokens per query-document pair, exceeds the 96 GB capacity even with batch size 1; we therefore mark it as > 96 GB in the figure. This highlights the high memory footprint of full-context training under long inputs.

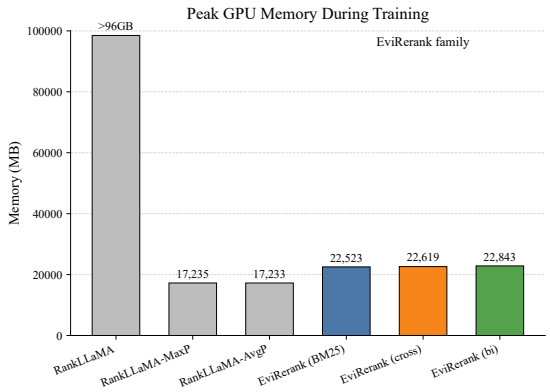


Figure 5: Peak GPU memory during training on DL19 (MB), measured on an NVIDIA H20 (96 GB). *Note*: full-document RankLLaMA (4096-token input, batch size 1) exceeds 96 GB and is reported as > 96 GB.

In contrast, EVIRERANK trains within a much smaller memory budget by constructing a compact evidence context under a strict cap. Across selectors, peak memory stays around 22-23 GB.

Pooling-based baselines (RankLLaMA-MaxP/AvgP) are the most memory-efficient (17,235/17,233 MB), but they are substantially less effective than EVIRERANK (Sec. 5.1). Overall, budget-aware evidence construction offers a practical accuracy-resource trade-off for long-document reranking.

B Implementation and Reproducibility Details

B.1 Local Block Scorers

This section provides the formal definitions and implementation details for the three local scorers used in EviRerank: BM25, cross-encoder, and bi-encoder.

BM25. Given a query q and a block blk, the Retrieval Status Value (RSV) is:

$$RSV_{\text{BM25}}(q, blk) = \sum_{w \in q \cap blk} IDF(w) \cdot s(w, blk),$$

$$s(w, blk) = \frac{tf_w^{blk}}{k_1\left(1 - b + b\frac{l_{blk}}{l_{avg}}\right) + tf_w^{blk}}.$$

where tf_w^{blk} denotes the frequency of term w in block blk, l_{blk} denotes the block length, and l_{avg} denotes the average block length.

In this paper, we use the scikit-learn (Pedregosa et al., 2011) package to calculate the IDF dictionary with IDF smoothing. $IDF(w)$ is defined by:

$$IDF(w) = \log \frac{N+1}{df_w + 1} + 1,$$

where N and df_w are added one with default scikit-learn smooth IDF setting. The added 1 after the fraction, makes sure terms with zero IDF don't get suppressed entirely. Besides, l_{blk} is the length of block, l_{avg} the average length of the blocks in d, and k_1 and b two hyperparameters of BM25.

For Chinese documents, word w is recognized using the Jieba[3] Chinese text segmentation tool to obtain meaningful terms.

Cross-encoder (interaction). For each block b_i, a pretrained encoder (e.g., BERT) takes concatenated query and block tokens:

$$b_i^{cls} = \text{PLM}\big([CLS], q_tokens, [SEP], b_i_tokens\big),$$

where b_i^{cls} is the [CLS] embedding. A linear layer maps it to a relevance score:

$$RSV(q, b_i) = W b_i^{cls},$$

with W the learnable weights. Because blocks are short, this remains efficient while retaining rich interactions.

Bi-encoder (dense retrieval). A shared encoder independently maps queries and blocks:

$$\mathbf{q}^{\text{cls}} = \text{PLM}\big([\text{CLS}], \mathbf{q}_{\text{tok}}\big),$$
$$\mathbf{b}_i^{\text{cls}} = \text{PLM}\big([\text{CLS}], \mathbf{b}_{i,\text{tok}}\big).$$

Local relevance is computed as cosine similarity:

$$RSV(q, b_i) = \cos\left(\mathbf{q}^{\text{cls}}, \mathbf{b}_i^{\text{cls}}\right).$$

Block embeddings can be precomputed offline, making this approach extremely efficient at runtime.

B.2 Dataset Statistics

Table 11 summarizes genres, corpus sizes, test query counts, and average document lengths (tokenized by LLaMA2).

B.3 Baseline Details

We compare against competitive first-stage and reranking systems. Baseline sets align with prior work while remaining consistent across datasets where applicable.

[3] https://github.com/fxsjy/jieba

Table 11: Datasets statistics (with LLaMA2-7B tokenizer).

Dataset	genre	#documents	#test query	Avg. Num. of Tokens
TREC DL'19 doc (MS MARCO)	Web Documents	3,213,835	43	1958
TREC DL'22 doc (MS MARCO v2)	Web Documents	11,959,635	76	3782
TREC DL'23 doc (MS MARCO v2)	Web Documents	11,959,635	82	3782
MLDR-zh	Wikipedia + Wudao	200,000	800	12899

DL'19 baselines (following (Li et al., 2023b)).

- *Traditional IR*: BM25 (Anserini) (Yang et al., 2018).
- *Neural IR*: TKL (Hofstätter et al., 2020), PARADE (Li et al., 2020), Sparse-Transformer (Child et al., 2019), Longformer-QA (Beltagy et al., 2020), Transformer-XH (Zhao et al., 2020), QDS-Transformer (Jiang et al., 2020).
- *Key-block selection*: IDCM (Hofstätter et al., 2021), KeyB(PARADE5) and KeyB (Li et al., 2023b).
- *LLM reranker*: RankLLaMA (Ma et al., 2024) and our segment-level variants RankLLaMA-MaxP / RankLLaMA-AvgP (defined below).

DL'22 / DL'23 / MLDR-zh baselines.

- *BM25*: Anserini for English; for MLDR-zh, we apply Chinese word segmentation.
- *KeyB family*: KeyB with BM25-based / bi-encoder-based selection; English uses BERT and MLDR-zh uses Chinese BERT-style encoders.
- *LLM reranker*: RankLLaMA and our MaxP / AvgP variants.

B.4 Training Data and Evaluation Metrics

Training data. We adopt standard triplet construction:

- **DL'19**: 200K triplets from MS MARCO v1 (positives from qrels; negatives from top-100).
- **DL'22/DL'23**: 200K triplets from MS MARCO v2 (same protocol).
- **MLDR-zh**: 10K labeled queries with one positive and one negative each.

Metrics. We follow community practice per dataset:

- **DL'19/DL'22/DL'23**: NDCG@10, MAP.
- **MLDR-zh**: The test file for each query has one positive document together with seven negative documents. We therefore report P@1, MAP, and NDCG@8.

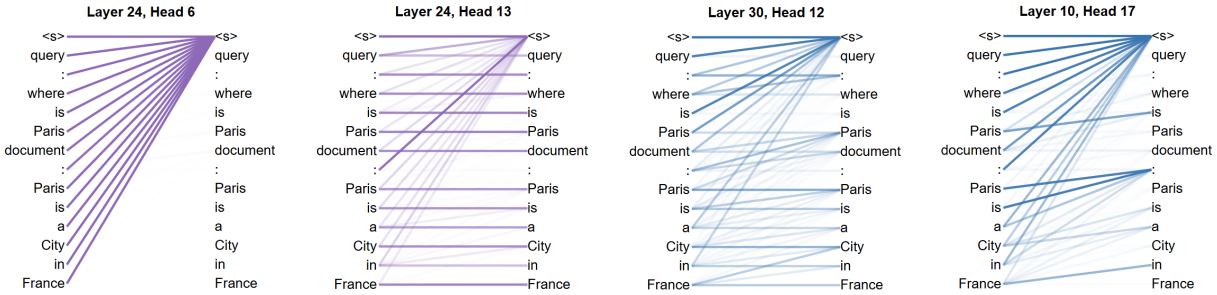
C Attention Analysis: Why Block Selection Still Matters

With the advent of PLMs like BERT, IR systems have seen substantial improvements in document ranking accuracy. Among these, re-ranking models, often referred to as cross-encoders, harness the power of fine-tuned PLMs or decoder-only LLMs for downstream IR tasks. Despite the impressive performance of decoder-only LLMs like RankLLaMA, the inner workings of these LLMs, specifically how they assess and rank the relevance of passages, remain underexplored.

C.1 Attention Heatmaps of Specific Examples

To shed light on this and answer **RQ1**, we propose first analyzing the attention heatmaps of the RankLLaMA model with several specific examples, which has been fine-tuned on the MSMARCO passage ranking dataset[4]. Clark et al. (2019) propose a similar investigation for BERT and find a significant portion of BERT's attention is focused on the delimiter token, and certain attention heads align well with linguistic features such as syntax and coreference. However, recent LLMs are unidirectional and decoder-only, especially for the IR focused LLMs, and may display a different behavior regarding how tokens attend to each other. We used the BertViz tool (Vig, 2019) to explore these attention patterns in the decoder-only LLM RankLLaMA when processing various query-passage pairs.

[4]available at: https://huggingface.co/castorini/rankLlama-v1-7b-lora-passage



(a) Delimiter heads (b) Local heads (c) Broad heads (d) Broad heads

Figure 6: Examples of attention heads that focus on delimiters, local context, or attend broadly.

We begin with a simple and arbitrary text pair: the query text is "where is Paris", and the document text is "Paris is a city in France". With the format "query: [query] document: [document]", after Llama2 tokenizer, the input tokens are: '<s>', '_query', ':', '_where', '_is', '_Paris', '_document', ':', '_Paris', '_is', '_a', '_City', '_in', '_France'. We check several attention heads which are shown in Fig. 6, Fig. 7, Fig. 8 and Fig. 9. To help showing weak attention weights, the last three figures are sharpened. Our findings are summarized in the following points:

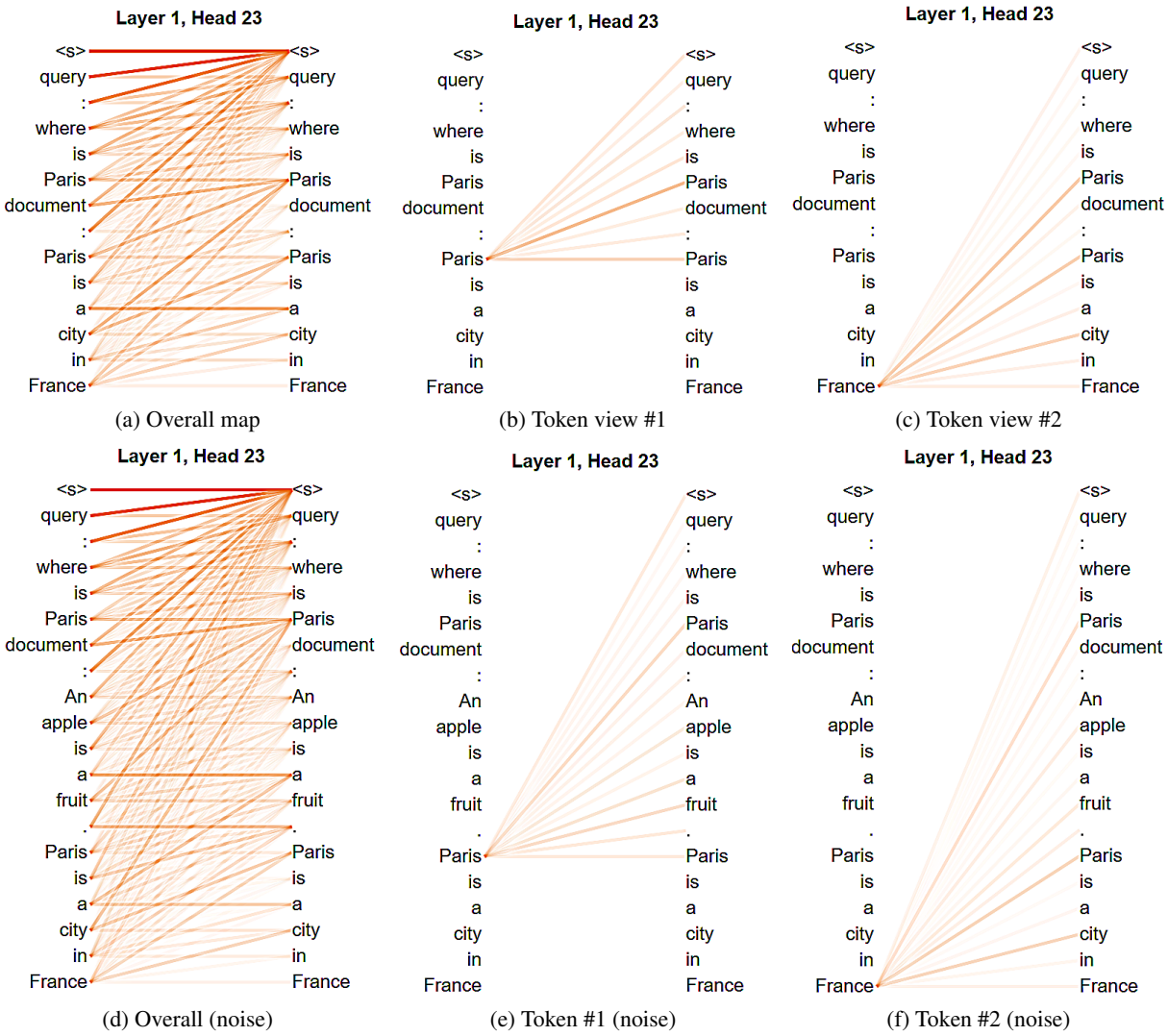
- Attention to delimiter, current and broad tokens: Similar to (Clark et al., 2019) in BERT, we observe that a substantial portion of attention is directed towards the delimiter token <s>, e.g., Fig. 6a. Clark et al. (2019) speculate that attention over the <s> delimiter tokens might be used as a sort of "no-op", and we conjecture that the attention over the beginning <s> is also "no-op" for IR scenario. Besides, we also observe a small amount of attention is directed towards the current token (Fig. 6b), and that some attention heads attend broadly over all tokens (Fig. 6c, Fig. 6d), which is consistent with (Clark et al., 2019).

- Attention focusing on relevant tokens: Beyond delimiter-focused or self-focused heads, we observe several attention heads that directly capture semantic relations between relevant tokens in the query and document. In particular, as shown in each above three subfigures in Fig. 7, Fig. 8, and Fig. 9, these relevance-focused heads consistently highlight key cross-token alignments: for head 23 in layer 1, tokens such as "Paris" and "France" in the document primarily attend to corresponding relevant tokens in both query and document segments; for head 25 in layer 8, tokens "Paris" and "France" focus strongly on the query token "Paris"; for head 24 in layer 31, similar alignment between query and document relevance tokens is also preserved. These patterns suggest that a subset of attention heads are able to capture fine-grained token-level relevance signals, forming direct associations between query intents and document content—an essential mechanism underlying accurate relevance estimation.

- Attention with irrelevant information: To investigate the effect of irrelevant tokens, we insert unrelated content such as "An apple is a fruit" into the document. This setting is illustrated in the three subfigures in Fig. 7, Fig. 8, and Fig. 9. The attention head 25 in layer 8 (Fig. 8e) effectively suppresses the irrelevant content, maintaining strong attention on the relevant query token "Paris." This head appears to filter out the noise (e.g., apple) and prioritize query-relevant content.

  In contrast, head 23 in layer 1 (Fig. 7e) continues to exhibit broad attention across all tokens. Although the attention may be weak, it reveals that the head does not clearly distinguish between relevant and irrelevant content. Similarly, head 24 in layer 31 (Fig. 9e) shows that the token "Paris" in the document attends not only to the query token "Paris" but also weakly to irrelevant tokens such as "apple," "is," and "a."

  Importantly, except for head 25 in layer 8, **the introduction of irrelevant tokens leads to a noticeable reduction in the attention weight toward the query token "Paris."** For instance, in Fig. 7f and Fig. 9f, the attention from the final document token "France" to the query token "Paris" becomes weaker after inserting irrelevant information.



(a) Overall map (b) Token view #1 (c) Token view #2

(d) Overall (noise) (e) Token #1 (noise) (f) Token #2 (noise)

Figure 7: Attention maps of Layer 1 Head 23 on a clean document (top) and with appended noise (bottom).

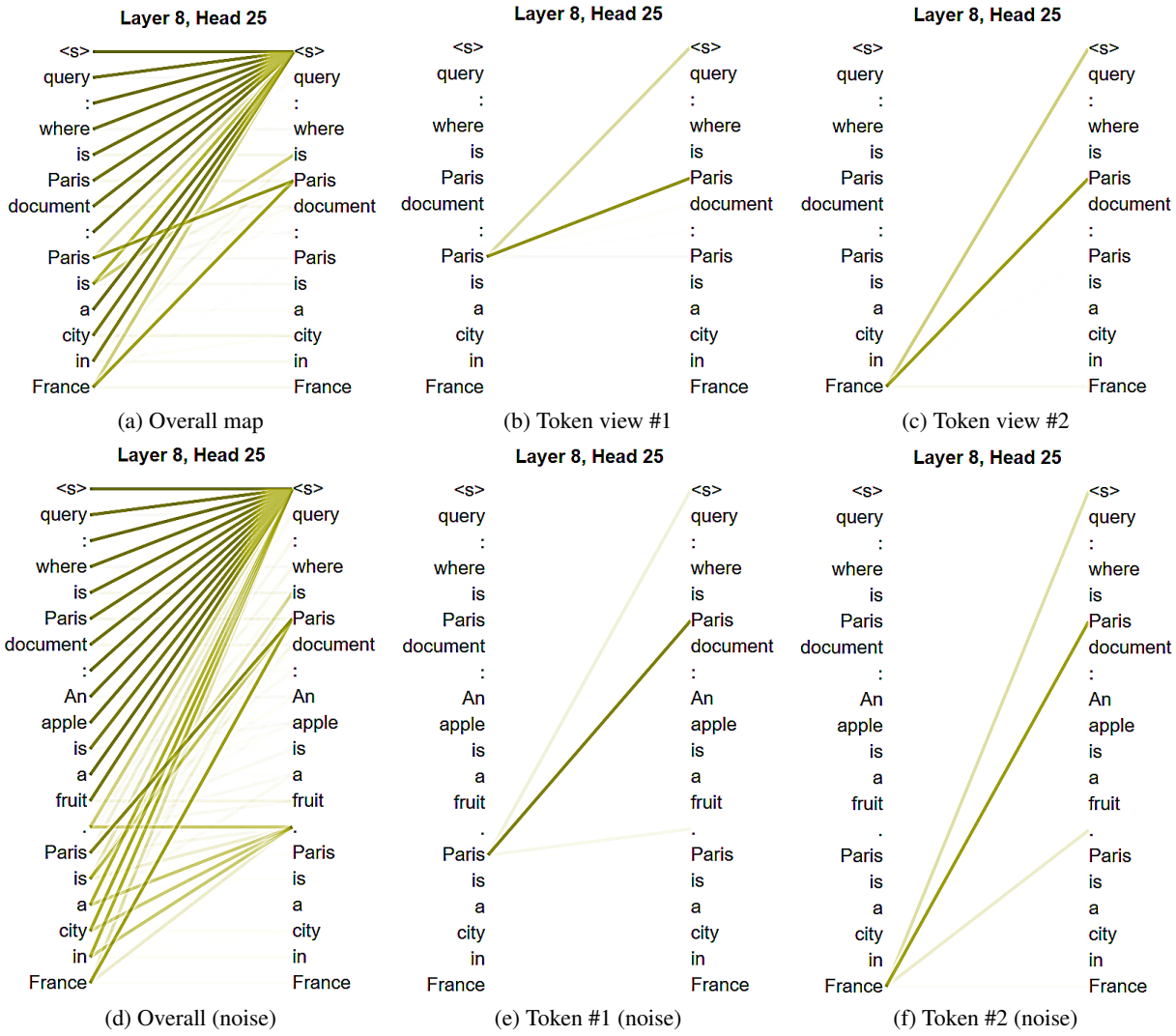
To support this observation quantitatively, Table 12 presents the measured attention scores from "France" to "Paris" in the query across the three heads. All heads exhibit decreased attention scores following the insertion of irrelevant tokens. Notably, while the decline for Layer 8 Head 25 is mild, the other two heads show substantial drops. This degradation may impair the model's ability to correctly assess the relevance between document and query, which is particularly critical in information retrieval scenarios. These irrelevant tokens will inevitably result in noisy information in the representation of the last token, which is the basic building block for computing the relevance score for RankLLaMA.

C.2 Aggregated Attention Heatmaps Across Examples

The previous section illustrates attention behaviours on individual examples. To investigate whether these patterns generalize across real-world long documents, we conduct a dataset-level aggregation analysis. Specifically, we sample 500 query–relevant document pairs from the TREC DL19 document ranking's development set (MS MARCO) (Craswell et al., 2020). The RankLLaMA model fine-tuned for document-level reranking[5] is used for evaluation. All documents are truncated to 1200 tokens.

For each example, we compute the average attention scores from document tokens to query tokens across all heads in every layer. The attention scores are then averaged across the sampled set to

[5] https://huggingface.co/castorini/rankllama-v1-7b-lora-doc



(a) Overall map (b) Token view #1 (c) Token view #2

(d) Overall (noise) (e) Token #1 (noise) (f) Token #2 (noise)

Figure 8: Attention maps of Layer 8 Head 25 on a clean document (top) and with appended noise (bottom).

produce an overall heatmap summarizing attention behaviour across layers and heads.

To assess robustness against irrelevant context, we also create two additional test settings. For each query–relevant document pair, we randomly sample a negative document, extract its first 800 tokens as noise, and insert the noise either before or after the relevant document. The aggregated attention heatmaps under these two noise injection settings are similarly computed. The resulting heatmaps for the clean and noisy cases are shown in Fig. 10.

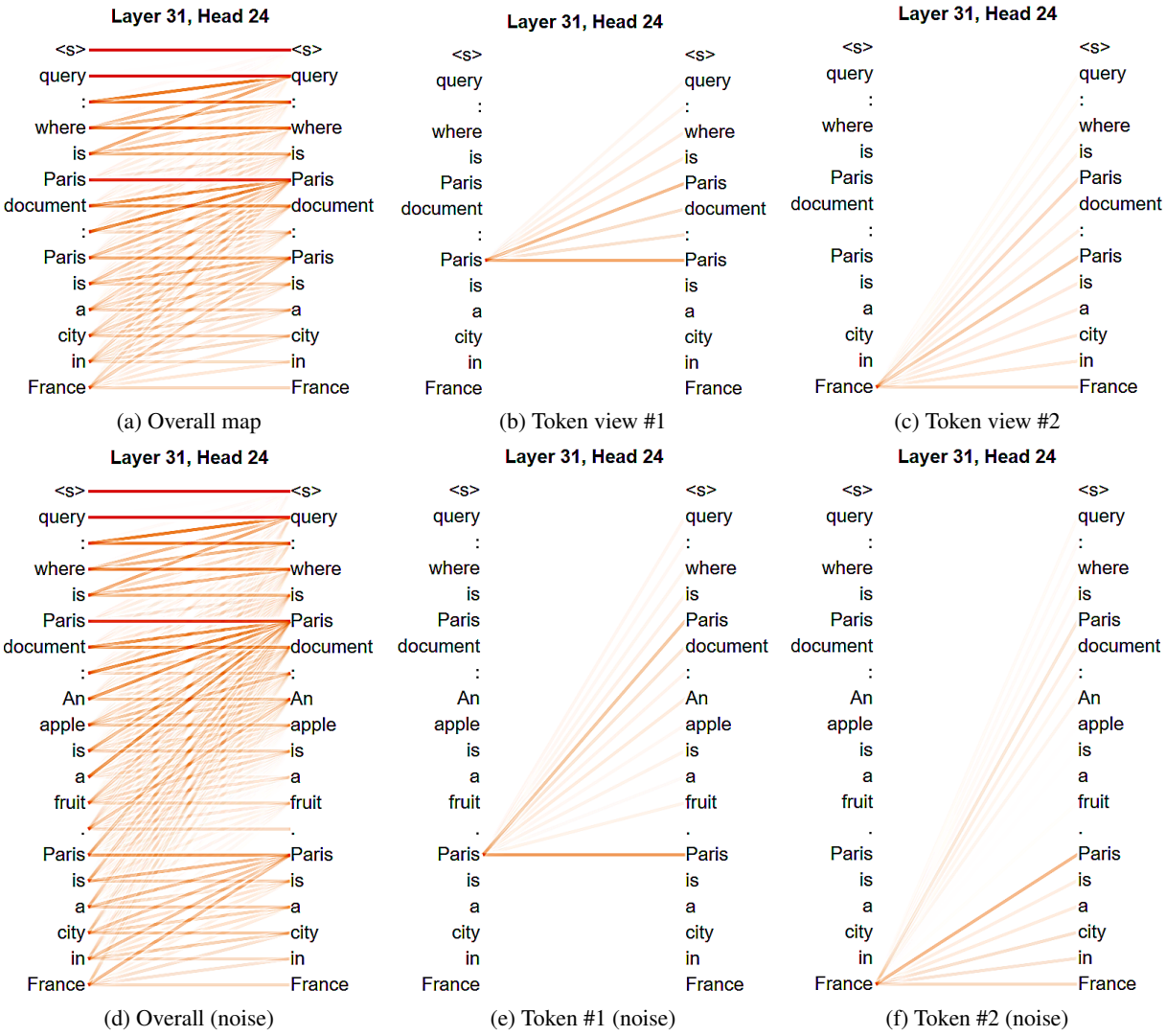
C.2.1 Findings

As shown in Fig. 10a, certain heads—primarily located in the beginning and middle layers, as well as several heads in the final layers—exhibit strong attention from document tokens to query tokens, aligning with relevance-focused behaviour observed earlier. However, when noise is inserted, both Fig. 10b and Fig. 10c display a general attenuation of attention scores, indicating that irrelevant content weakens the focus on relevant tokens. Notably, inserting noise *after* the relevant content appears to cause greater attention dispersion, likely because the model processes text autoregressively from left to right.

These aggregated patterns confirm that a fraction of attention heads identify relevance signals, and that irrelevant text can dilute these signals, motivating selective block extraction before LLM reranking.

C.3 Quantitative Analysis with Attention-Relevance Alignment Score (ARAS) and Positive Correlation Rate (PCR)

To complement the above qualitative attention observations and provide a more systematic and quantitative understanding of how LLM rerankers behave (addressing **RQ1**), we propose two evalua-



(a) Overall map (b) Token view #1 (c) Token view #2

(d) Overall (noise) (e) Token #1 (noise) (f) Token #2 (noise)

Figure 9: Attention maps of Layer 31 Head 24 on a clean document (top) and with appended noise (bottom).

tion metrics: the **Attention-Relevance Alignment Score (ARAS)** and the **Positive Correlation Rate (PCR)**. These metrics aim to measure how well the model's attention aligns with true relevance signals in long documents.

C.3.1 Metric Definitions

- Attention weight per chunk: Given a query-document pair (q, d), we first segment the document d into M non-overlapping chunks $\{C_1, C_2, \ldots, C_M\}$, where each chunk contains a fixed number of tokens. For a given attention head and layer, let $\mathbf{A} \in \mathbb{R}^{L \times L}$ denote the attention matrix for the entire input sequence of length L.

  Let $Q = \{q_1, q_2, \ldots, q_{|Q|}\}$ represent the token indices of the query portion in the input sequence. Then, for each document chunk C_i (corresponding to token indices $\{c_{i,1}, \ldots, c_{i,K}\}$), we compute its average attention weight toward the query tokens as:

$$\text{AttentionWeight}(C_i) = \frac{1}{K} \sum_{t \in C_i} \frac{1}{|Q|} \sum_{j \in Q} \mathbf{A}_{t,j}$$

  This reflects how much attention the document chunk as a whole assigns to the query tokens.

- Relevance score per chunk: For each chunk C_i, we estimate its relevance with respect to the query using an external cross-encoder model[6] which serve as an approximate "ground-truth" relevance score:

[6] https://huggingface.co/cross-encoder/ms-marco-MiniLM-L6-v2

Table 12: The attention weights between "France" to "Paris" in the query when inserting noise tokens.

	Layer 1 Head 23 (Fig 7)	Layer 8 Head 25 (Fig 8)	Layer 31 Head 24 (Fig 9)
no noise tokens	0.2097	0.6500	0.1296
with noise tokens	0.1154 (↓44.97%)	0.6492 (↓0.12%)	0.0858 (↓33.80%)

$$\text{RelevanceScore}(C_i) = \text{CrossEncoder}(q, \text{text}(C_i))$$

where $\text{text}(C_i)$ denotes the surface text of chunk C_i.

- Attention-Relevance Alignment Score (ARAS): For each query-document pair (q, d), the ARAS is defined as the Spearman rank correlation coefficient between the attention weights and relevance scores across all chunks:

$$\begin{aligned} \text{ARAS}(q,d) &= \text{Spearman}(\mathbf{a}, \mathbf{r}), \\ \mathbf{a} &= \big(\text{Attn}(C_i)\big)_{i=1}^{M}, \\ \mathbf{r} &= \big(\text{Rel}(C_i)\big)_{i=1}^{M}. \end{aligned} \tag{10}$$

This measures how well the ranking induced by attention weights aligns with the ranking of relevance scores across chunks. For reporting, we compute the overall mean ARAS across the samples:

$$\overline{\text{ARAS}} = \frac{1}{N}\sum_{j=1}^{N} \text{ARAS}(q_j, d_j)$$

This average ARAS indicates the typical alignment strength per query-document pair.

- Positive Correlation Rate (PCR): Given a collection $\mathcal{D} = \{(q_j, d_j)\}_{j=1}^{N}$ of N query-document pairs, we compute PCR as the proportion of pairs whose ARAS is positive:

$$\text{PCR} = \frac{1}{N}\sum_{j=1}^{N} \mathbb{I}\left(\text{ARAS}(q_j, d_j) > 0\right) \tag{11}$$

where $\mathbb{I}(\cdot)$ is the indicator function. PCR reflects how consistently attention correlates positively with relevance across the dataset.

ARAS focuses on the alignment strength between attention and relevance for individual query-document pairs, while PCR reflects alignment stability across the dataset. Together, these metrics offer both instance-level and corpus-level interpretability.

C.3.2 Experimental Setup

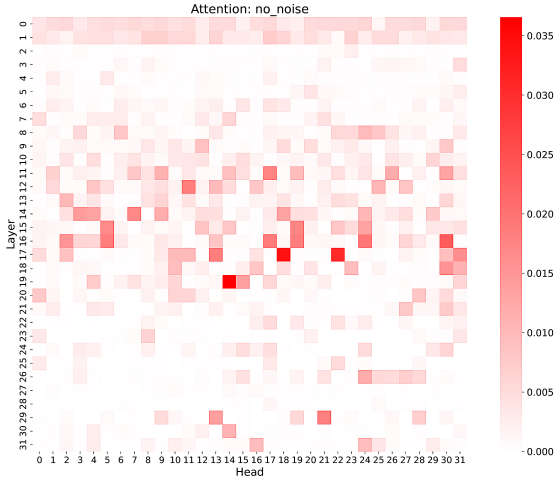
We conduct controlled experiments on 500 query-relevant document pairs sampled from the same development set (MS MARCO) as Section C.2. Each document is truncated to 1200 tokens, and segmented into 64-token chunks for analysis. The ARAS and PCR metrics are then computed across three representative attention heads previously identified: Layer 1 Head 23, Layer 8 Head 25, and Layer 31 Head 24. The figures are shown in Fig. 11.

To simulate long irrelevant contexts, we insert 800 to 1800 noise tokens either before or after the relevant document content, allowing us to test attention stability under varying noise levels.

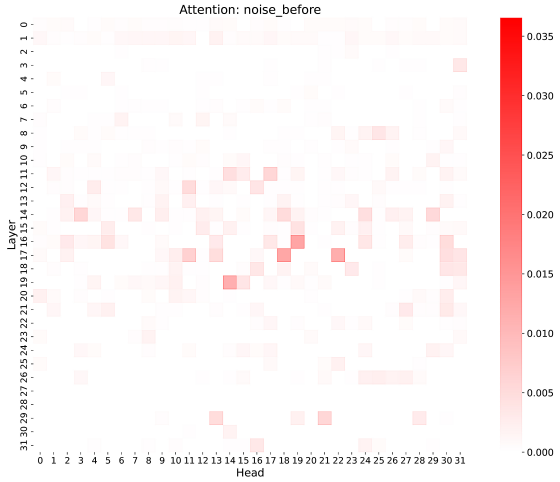
C.3.3 Results and Findings

Layer 1, Head 23. Under clean input, ARAS achieves 0.349 and PCR reaches 83.9%, suggesting moderate attention-relevance alignment at shallow layers. However, performance degrades significantly as noise is inserted, particularly when noise follows the relevant content. ARAS drops to negative values and PCR falls below 50% under heavy noise, indicating this head is highly sensitive to positional disruptions.

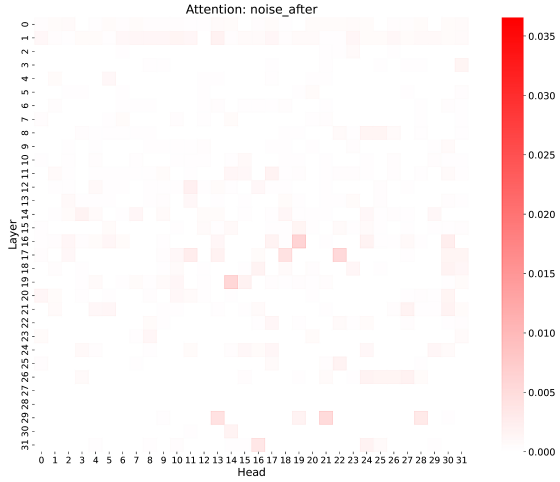
Layer 8, Head 25. This head shows the strongest relevance alignment: baseline ARAS reaches 0.602 and PCR 96.6%. Although both metrics decline with added noise, PCR remains relatively stable (above 70%), suggesting that this mid-layer head is better at resisting irrelevant information. Nevertheless, ARAS still declines sharply as more noise accumulates, indicating that although the head consistently identifies relevant regions, the precision of its attention distribution becomes less aligned with true relevance as noise increases.



(a) The overall attention heatmaps for relevant documents (without inserting noise)



(b) The overall attention heatmaps when inserting noise before relevant documents



(c) The overall attention heatmaps when inserting noise after relevant documents

Figure 10: The overall attention heatmaps across samples, comparing with inserting noise tokens.

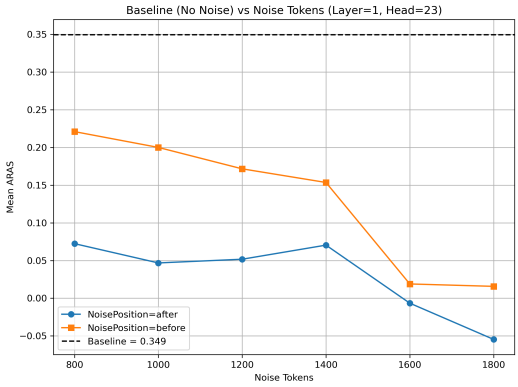
Layer 31, Head 24. This deep-layer head exhibits relatively weak alignment, with a baseline ARAS of 0.284 and PCR of 79.6%. As with earlier heads, both metrics decline in the presence of noise. Although its performance degrades more slowly than Layer 1, it still suffers substantial drops under high noise levels: ARAS falls below 0.1 and PCR drops to approximately 56%.

Overall Observations. Across all heads, inserting irrelevant tokens *after* the relevant content consistently causes more severe alignment degradation than inserting noise *before*, likely because autoregressive LLMs encounter relevant content later, reducing the usable attention capacity.

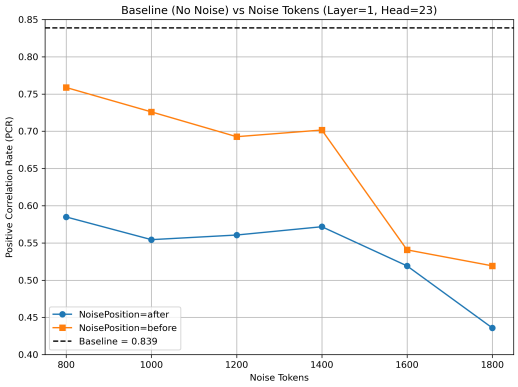
C.4 Summary: Implications for Long Document Retrieval

These findings provide a direct answer to **RQ1**, showing that while certain attention heads do exhibit relevance-focused behaviors, their ability to preserve this alignment diminishes substantially when irrelevant content accumulates. This degradation is especially pronounced when noise appears later in the sequence, likely due to the left-to-right processing nature of decoder-only LLMs such as RankLLaMA.

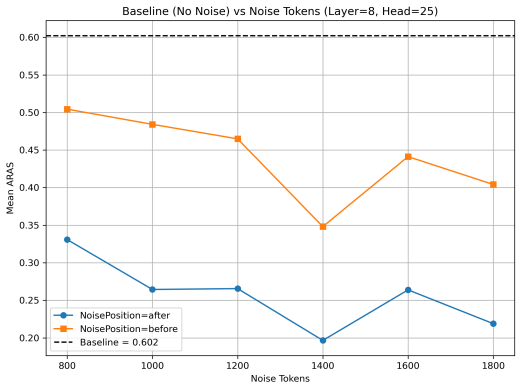
These results reinforce the need for explicit block selection before LLM reranking—not only to reduce computational overhead, but also to preserve attention focus, mitigate distraction from irrelevant text, and prevent attention dispersion. This insight validates the core motivation behind EviRerank: block selection remains necessary even in the era of large language models. By selecting and ranking the most relevant content blocks, we enable LLMs to concentrate attention on relevant information, thereby improving both retrieval effectiveness and computational efficiency.



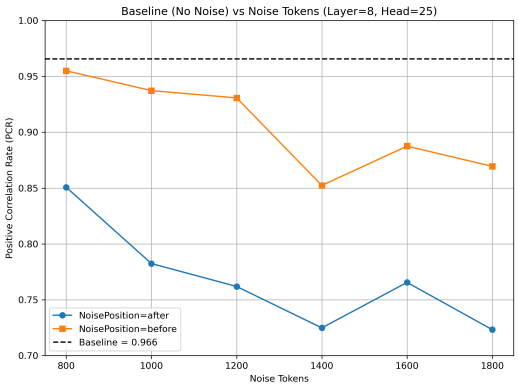
(a) ARAS of Layer 1 Head 23



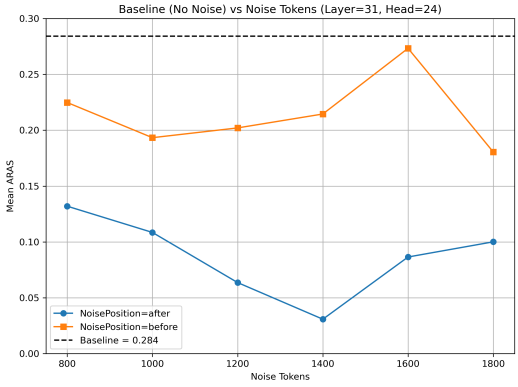
(b) PCR of Layer 1 Head 23



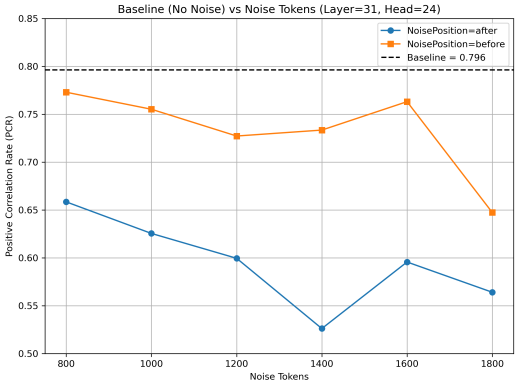
(c) ARAS of Layer 8 Head 25



(d) PCR of Layer 8 Head 25



(e) ARAS of Layer 31 Head 24



(f) PCR of Layer 31 Head 24

Figure 11: Mean ARAS scores and positive correlation rates for different layers across various noise lengths and different noise insertion positions. Baseline is the score without inserting noise tokens.