COMPRANK: EFFICIENT LONG-LIST LLM RERANKING VIA TOKEN-LEVEL COMPRESSION AND DECODING-FREE SCORING

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model (LLM) rerankers have become an important component of modern retrieval and retrieval-augmented generation pipelines, yet their computational cost makes reranking long candidate lists expensive. We introduce **CompRank**, a token-efficient reranking framework that reduces redundant computation through contextualized KV compression and decoding-free attention scoring. CompRank first encodes each document with its full causal context and then exposes only a small subset of the resulting key–value (KV) states to the scoring stage, allowing the retained states to preserve information from the omitted context. We instantiate CompRank in two complementary variants: **CompRank-QC**, which conditions document representations on the query for stronger relevance modeling, and **CompRank-Cache**, which uses query-independent document representations to enable cross-query KV reuse. Experiments on seven BEIR datasets show that CompRank-QC achieves an average NDCG@10 of 46.7 with Mistral-7B, outperforming the evaluated efficient reranking baselines, with consistent effectiveness gains also observed on Qwen2.5-7B. Notably, with Mistral-7B, Step-10 compression retains only 10.2% of document KV states while reducing average NDCG@10 by only 0.5 points compared with full-token scoring. It also yields approximately $1.4\times$ end-to-end speedup at 500 candidates even when documents are re-encoded for each query, while document-side KV reuse further reduces online latency by amortizing document encoding across queries. On TREC-COVID, CompRank further generalizes from 30-document training lists to candidate sets of up to 500 documents. These results demonstrate that contextualized KV compression and reusable document representations provide an effective path toward scalable long-list LLM reranking.

1 INTRODUCTION

Reranking is an important component of modern information retrieval (IR) and retrieval-augmented generation (RAG) pipelines Lewis et al. (2020); Lu et al. (2025). By applying fine-grained relevance modeling to candidates retrieved by first-stage models, rerankers, especially those based on large language models (LLMs), can improve retrieval precision and provide stronger evidence for downstream reasoning Lu et al. (2026b); Huang et al. (2026). However, these gains come with substantial computational cost. In long-list scenarios, such as reranking the top-100 retrieved documents, LLM-based rerankers process long contexts and incur expensive attention computation, making latency and memory major bottlenecks. Recent work has explored several approaches to improve reranking efficiency. Direct-Rank Lu et al. (2026a) shows that effective ranking does not require lengthy chain-of-thought reasoning. ICR Chen et al. (2025) and QRHead Zhang et al. (2025) reduce decoding overhead by deriving relevance scores directly from internal attention patterns, while BlockRank Gupta et al. (2025) and LongRanker Zhou et al. (2026) use structured attention to reduce unnecessary interactions in long contexts. Despite these advances, substantial redundancy remains in the document states exposed to relevance scoring.

In this work, we revisit reranking from the perspective of *contextualized document compression*. Once a document has been encoded, accurate relevance estimation need not require access to every token state. In a causal language model, each token representation incorporates information from

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

its preceding context, so a small subset of contextualized states can represent information beyond the selected positions themselves. This motivates a simple principle: preserve full context during document encoding, then reduce the number of states involved in downstream scoring. Step-wise selection supports this principle by retaining states throughout the document and maintaining broad coverage. A related design choice is whether document representations should incorporate the query. Placing the query before the documents produces query-conditioned states with stronger relevance signals, whereas encoding documents independently makes their representations reusable across queries. These configurations provide complementary operating points: query-conditioned encoding prioritizes ranking effectiveness, while query-independent encoding enables efficient serving through document-side KV reuse.

To this end, we propose **CompRank**, a token-efficient reranking framework for long candidate lists. CompRank combines block-decoupled document encoding, contextualized KV compression, and directly supervised attention scoring. Document blocks share positional coordinates and do not attend to other candidates, removing cross-document dependencies during encoding. Step-wise compression restricts the document states exposed to the trailing scoring block, while a candidate-level cross-entropy objective directly optimizes the attention-derived scores used at inference. Ranking is performed by sorting these scores without generating document identifiers. We instantiate CompRank in two variants. **CompRank-QC**, our primary model, conditions document representations on the query for stronger ranking effectiveness. **CompRank-Cache** instead uses query-independent document representations, enabling independent precomputation and cross-query KV reuse.

Experiments on seven BEIR datasets demonstrate the effectiveness of CompRank across two LLM backbones. CompRank-QC achieves average NDCG@10 scores of 46.7 with Mistral-7B and 43.0 with Qwen2.5-7B, obtaining the best average performance among the evaluated methods for each backbone. With Mistral-7B, it outperforms BlockRank on all seven datasets. Compression experiments further show that Step-10 retains only 10.2% of document states while preserving most full-token effectiveness in both variants: CompRank-QC achieves 46.7 versus 47.2 under full-token scoring, while CompRank-Cache achieves 39.2 versus 39.7. Across the evaluated selection strategies, step-wise compression provides a strong effectiveness–compression trade-off.

Further scaling experiments on TREC-COVID show that CompRank generalizes from 30-document training lists to candidate sets of up to 500 documents. At 500 candidates, Step-10 yields approximately $1.4\times$ end-to-end speedup over full-token scoring even when documents are re-encoded for each query. CompRank-Cache further reduces online latency through document-side KV reuse. Together, these results show that contextualized KV compression can substantially reduce scoring cost with only minor effectiveness degradation, while reusable document representations provide an additional efficiency gain for repeated-query workloads.

Our contributions are summarized as follows:

- We propose **CompRank**, a token-efficient framework for long-list LLM reranking that compresses contextualized document KV states and directly optimizes decoding-free attention scores.
- We introduce **CompRank-QC** and **CompRank-Cache**, two complementary variants that expose the trade-off between query-conditioned ranking effectiveness and reusable document computation.
- We demonstrate strong reranking performance across seven BEIR datasets and two backbones, and show that step-wise compression preserves most full-token effectiveness while retaining only approximately 10% of document states.

2 CompRank

We introduce **CompRank**, a token-efficient framework for long-list LLM reranking. Its central design is to encode documents with their full context, then compress the resulting KV states for decoding-free attention scoring. This allows retained states to incorporate information beyond the selected token positions. Our primary variant, **CompRank-QC**, conditions document encoding on the query to improve relevance modeling. We first describe its encoding, compression, and scoring

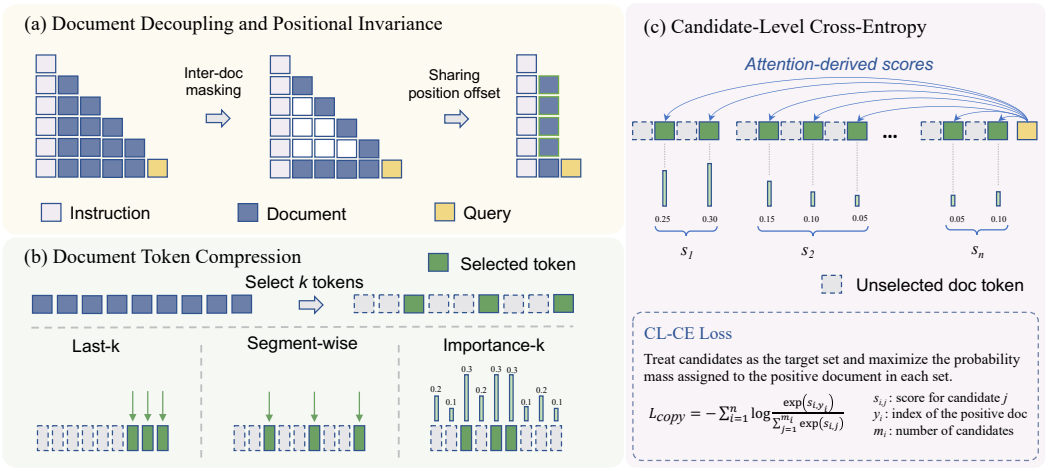

Figure 1: Overview of CompRank. (a) Block-decoupled encoding prevents cross-document attention and assigns shared positional coordinates to document blocks. CompRank-QC incorporates the query into the shared instruction prefix to construct query-conditioned document representations. (b) Contextualized token compression retains a subset of document KV states using last-k, segment-wise, or importance-based selection. (c) Attention mass over the retained states produces candidate scores, which are directly optimized with a candidate-level cross-entropy objective.

components, then introduce **CompRank-Cache**, a query-independent extension for cross-query KV reuse.

2.1 Query-Conditioned Block Encoding

Given a query q and candidate documents $\mathcal{D} = \{d_1, \ldots, d_N\}$, CompRank assigns each document a relevance score s_i. Let $I(q)$ denote the instruction augmented with the query, Q the trailing query block, D_i the token block of document d_i, and B the decision block. CompRank-QC uses

$$X_{\mathrm{QC}} = [I(q); D_1; \ldots; D_N; Q; B].$$

Following BlockRank Gupta et al. (2025), each document attends to the shared query-conditioned instruction and its own causal context, but not to other documents:

$$(K_i, V_i) = f_\theta(D_i \mid I(q)).$$

Thus, document representations are query-conditioned while remaining independent across candidates. The trailing query and decision blocks attend to the retained document states for relevance scoring.

To avoid position shifts caused by candidate ordering, all document blocks share local positional coordinates:

$$\mathrm{pos}(x_{i,t}) = C_D + t, \qquad \mathrm{pos}(b_t) = C_B + t.$$

The offsets are independent of document index and candidate-list length.

Together, block masking and shared positions make document representations independent across candidates for a fixed query, while final scores remain candidate-set dependent through the decision block and attention normalization. Details are provided in Appendix A.

2.2 Contextualized KV Compression

Block encoding removes inter-document attention, but the decision block still interacts with all document tokens. CompRank reduces this interaction cost by exposing only a subset of contextualized document KV states to scoring.

For a document with L_i valid tokens, let $\mathcal{T}_i = \{1, \ldots, L_i\}$ and $\mathcal{T}'_i \subseteq \mathcal{T}_i$ denote its original and retained position sets. At layer ℓ, the selected states are

$$\widetilde{K}_i^{(\ell)} = K_i^{(\ell)}[\mathcal{T}'_i], \qquad \widetilde{V}_i^{(\ell)} = V_i^{(\ell)}[\mathcal{T}'_i].$$

Compression applies to document states accessed by the decision block: document encoding continues to use the full causal context and query prefix. A retained state can thus incorporate information from preceding tokens that are not themselves selected, motivating compression after contextual encoding.

Step-wise selection. Our default strategy samples states at regular intervals, providing coverage throughout the document without an additional importance-estimation step. With stride r, start-anchored selection retains

$$\mathcal{T}_i' = \{1 + jr \mid j \in \mathbb{Z}_{\geq 0},\ 1 + jr \leq L_i\}.$$

The end-anchored alternative retains $\{L_i - jr \mid j \in \mathbb{Z}_{\geq 0},\ L_i - jr \geq 1\}$. Both retain approximately $1/r$ of the document states; we use $r = 10$ by default. This design combines contextualized representations with distributed sampling, motivating our analysis of contextualization and document coverage.

Alternative selection strategies. We compare step-wise selection with two alternatives. **Last-k selection** retains the final $\min(k, L_i)$ tokens, whose states have access to the preceding document context. **Intrinsic importance selection** uses the final document token to score positions at layer ℓ_{sel}:

$$\alpha_{i,t} = \frac{1}{H_{\text{sel}}} \sum_{h=1}^{H_{\text{sel}}} \frac{Q_{i,L_i}^{(\ell_{\text{sel}},h)} \left(K_{i,t}^{(\ell_{\text{sel}},h)}\right)^\top}{\sqrt{d_h}},$$

where H_{sel} is the number of selection heads and d_h is the head dimension. We retain the top-k positions, optionally supplemented with anchors such as the final token. These importance scores are query-conditioned in CompRank-QC.

The overall retention ratio, including any anchors, is

$$\rho = \frac{\sum_i |\mathcal{T}_i'|}{\sum_i L_i}.$$

For n_s scoring-block tokens and attention width d, compression reduces document interaction at each affected layer from $O(n_s \sum_i L_i\, d)$ to $O(n_s \sum_i |\mathcal{T}_i'|\, d)$. Full document encoding remains necessary, so end-to-end savings depend on the share of computation spent on these interactions.

2.3 Attention-Based Scoring and Supervision

Building on attention-based reranking Chen et al. (2025); Zhang et al. (2025), CompRank scores each candidate by aggregating attention mass over its retained states. Let $U \subseteq B$ denote the decision-token positions, ℓ^* the scoring layer, and H the number of scoring heads. For decision token u and retained document position $t \in \mathcal{T}_i'$, the attention logit is

$$a_{u,i,t}^{(h)} = \frac{Q_u^{(\ell^*,h)} \left(K_{i,t}^{(\ell^*,h)}\right)^\top}{\sqrt{d_h}} + m_{u,i,t},$$

where $m_{u,i,t}$ is the attention mask. We define the scoring distribution by normalizing over retained document positions:

$$p_{u,i,t}^{(h)} = \frac{\exp(a_{u,i,t}^{(h)})}{\sum_{j=1}^{N} \sum_{t' \in \mathcal{T}_j'} \exp(a_{u,j,t'}^{(h)})}.$$

Instruction, query, and decision tokens are excluded from this scoring normalization.

The document score is

$$s_i = \text{Agg}_{u \in U} \left[\frac{1}{H} \sum_{h=1}^{H} \sum_{t \in \mathcal{T}_i'} p_{u,i,t}^{(h)} \right],$$

where Agg aggregates decision positions. Our main configuration uses a single decision token. The scoring layer and readout configuration are fixed across evaluation datasets for each backbone; details appear in Appendix A.

Candidate-level supervision. We directly supervise these scores with a candidate-level cross-entropy objective:

$$P_\theta(i \mid q, \mathcal{D}) = \frac{\exp(s_i/\tau)}{\sum_{j=1}^{N} \exp(s_j/\tau)}, \qquad \mathcal{L}_{\text{rank}} = -\log P_\theta(y \mid q, \mathcal{D}),$$

where $\tau > 0$ is a temperature and y is the annotated positive candidate. This standard objective selects among input candidates, as in pointer- and copy-based models Gu et al. (2016), and aligns supervision with the scores used at inference.

At inference, we sort $\{s_i\}_{i=1}^{N}$ directly, without autoregressively generating document identifiers. The candidate-defined output space accommodates different list lengths without a fixed identifier vocabulary; we evaluate generalization to longer lists empirically.

2.4 CompRank-Cache: Query-Independent Encoding

CompRank-QC incorporates query information during document encoding, requiring documents to be re-encoded for each new query. For workloads with repeated document access, **CompRank-Cache** moves the instruction and query after the document blocks:

$$X_{\text{Cache}} = [D_1; \ldots; D_N; I; Q; B].$$

Each document uses only its own causal context, yielding query-independent states:

$$(K_i, V_i) = f_\theta(D_i).$$

As in CompRank-QC, documents share local positions, and suffix positions are independent of candidate order and list length.

Offline encoding and online scoring. Documents can be encoded and compressed offline, with retained KV states stored alongside document identities, selected positions, and positional metadata. For each query, the candidate states are assembled at the required layers, and only the trailing instruction–query–decision sequence is computed. The suffix attends to retained document states, and ranking follows the same scoring rule as CompRank-QC. Reuse requires unchanged model parameters, document content, tokenization, attention masks, and positional conventions. Under these conditions, cached execution matches the corresponding uncached CompRank-Cache computation up to numerical differences.

The two variants share contextualized KV compression and attention-based supervision, while differing in when query information enters the computation. CompRank-QC prioritizes query-conditioned relevance modeling; CompRank-Cache supports amortizing document encoding across repeated queries.

3 Experiments

This section presents the experimental setup, the main results of CompRank, and the comparison of different token compression strategies.

3.1 Experiment Setup

We evaluate CompRank on seven datasets from the BEIR benchmark Thakur et al. (2021), covering diverse retrieval tasks and domains: FiQA, SciFact, TREC-COVID, DBPedia, Climate-FEVER, NFCorpus, and SciDocs. For comparison, we consider representative retrieval and reranking baselines, including the sparse retriever BM25, generation-based listwise rerankers such as RankGPT Sun et al. (2023), and attention-based in-context reranking methods including ICR Chen et al. (2025) and CoRe heads Tran et al. (2026). For evaluation, BM25 is used to retrieve the top-100 candidate documents for each query, and all reranking methods reorder the same candidate set. We report NDCG@10 multiplied by 100 throughout the text, tables, and figures; averages are unweighted means over the seven datasets.

We evaluate CompRank with Mistral-7B and Qwen2.5-7B backbones and use full-parameter fine-tuning. Training and readout details are given in Appendix A. We adopt MS MARCO-derived training

data following the BlockRank Gupta et al. (2025) construction pipeline, with query–document candidate sets and annotated positive instances. During training, each query is paired with 30 candidate passages retrieved by a first-stage retriever. Unless otherwise specified, we use step-wise compression with Step-10 as the default compression strategy. The learning rate is 3.0×10^{-6}. Training and effectiveness evaluation use 8 NVIDIA H20 GPUs; the scaling latency experiment in Section 4.2 uses a single A100 80GB GPU with batch size 1. The controlled selection study in Section 4.1 uses five sampling seeds for randomized selectors while holding the trained checkpoint fixed.

3.2 Main Results

Comparison with reranking baselines. Table 1 compares the two CompRank variants with retrieval and reranking baselines across seven BEIR datasets. CompRank-QC achieves the highest average performance within each backbone group, reaching 46.7 with Mistral-7B and 43.0 with Qwen2.5-7B. With Mistral-7B, CompRank-QC improves over BlockRank from 45.5 to 46.7 on average and obtains higher scores on all seven datasets. With Qwen2.5-7B, it also improves over the strongest evaluated baseline, Direct-List, from 41.9 to 43.0. These results indicate that the effectiveness of query-conditioned document encoding is consistent across different backbone models rather than being specific to a single architecture. CompRank-Cache obtains an average of 39.2 with both backbones. Although its effectiveness is lower than that of CompRank-QC, it represents a different operating point: document representations are query-independent and can therefore be reused across queries. This trade-off is especially relevant for serving scenarios in which the same documents are repeatedly accessed.

Model	Climate-FEVER	DBPedia	FiQA	NFCorpus	SciDocs	SciFact	TREC-COVID	Avg.
BM25	16.5	31.8	23.6	33.8	14.9	67.9	59.5	35.4
Qwen2.5-7B Backbone								
RankGPT	<u>19.6</u>	<u>37.1</u>	29.5	34.1	16.6	69.3	70.5	39.5
ICR	<u>19.6</u>	35.3	27.0	32.7	16.4	<u>71.1</u>	66.1	38.3
CoRe heads	19.3	32.4	30.6	32.5	14.5	70.0	67.7	38.1
Direct-List	17.0	**41.9**	<u>34.3</u>	**35.7**	<u>17.4</u>	69.7	<u>77.4</u>	<u>41.9</u>
CompRank-Cache	17.7	36.0	30.4	32.0	14.5	62.7	**81.1**	39.2
CompRank-QC	**23.2**	36.3	**37.4**	<u>35.1</u>	**18.4**	**74.2**	76.5	**43.0**
Mistral-7B Backbone								
RankGPT	16.7	32.9	24.1	31.0	14.7	65.5	60.6	35.1
ICR	20.6	31.4	31.0	33.2	16.2	72.4	63.9	38.4
BlockRank	<u>24.6</u>	<u>46.5</u>	<u>41.0</u>	<u>35.9</u>	<u>17.8</u>	<u>74.2</u>	78.5	<u>45.5</u>
CompRank-Cache	16.6	34.8	31.7	31.7	14.3	64.2	**81.2**	39.2
CompRank-QC	**25.5**	**46.6**	**41.7**	**36.2**	**18.5**	**77.0**	<u>81.1</u>	**46.7**

Table 1: Results on seven BEIR datasets. Both CompRank variants use Step-10. Best and second-best results within each backbone group are shown in bold and underlined, respectively.

QC compression. The complete Mistral QC sweep appears in Appendix B, Table 4. Step-10 achieves 46.7 with 10.2% of document states retained, compared with a full-token reference of 47.2. Step-5 retains 20.0% and obtains 46.1, while Step-20 retains 5.2% and obtains 45.4. Among the compressed QC configurations, Step-10 has the highest average. Last-32 retains more states (14.1%) but obtains 45.0; pure importance-based Top-k selection obtains 42.8–43.8. The corresponding full-token score is a reference configuration, not a theoretical upper bound.

Cache compression. Table 5 reports the Mistral Cache sweep. Step-10 obtains 39.2 with 10.2% retention, 0.5 points below the full-token reference of 39.7. Step-5 obtains 39.1 at 20.0% retention, and Step-20 obtains 38.4 at 5.2%. Last-32 reaches 38.0 at 14.1%, while pure Top-k configurations obtain 35.8–36.8. Step-wise selection therefore provides a strong trade-off in both variants. These sweeps compare the reported strategy configurations; the shared-checkpoint, matched-budget study in Section 4.1 separately examines the role of selection coverage and contextualization.

4 ANALYSIS

4.1 UNDERSTANDING TOKEN SELECTION

We examine document coverage and full-context encoding with a controlled Mistral-7B CompRank-QC comparison. Unlike the default Step-10 configuration, this study uses one full-token-trained checkpoint and changes the inference-time selection rule while keeping scoring and structural-token treatment fixed.

Matched-budget selection. For document length L_i, each selector retains $b_i = \max(1, \lceil \rho L_i \rceil)$ states, with $\rho \in \{0.05, 0.10, 0.20\}$. Last selects the final b_i positions; Global Importance selects the highest-scoring positions; Random samples without replacement. Stratified Random partitions the document into b_i contiguous segments and samples one position per segment. Stratified Importance selects the highest-scoring position in each segment. Step-wise distributes the budget approximately uniformly. Importance selectors use the same scoring function and selection layer. Randomized selectors are evaluated over five sampling seeds with the checkpoint fixed. At 10% retention, Step-wise and Stratified Random obtain 46.7 and 46.0, compared with 44.8 for unrestricted Random selection. Stratified Importance improves over Global Importance from 44.3 to 46.1. These matched-budget comparisons support distributing retained states across document regions, rather than concentrating the budget solely at globally high-scoring positions. They do not establish where relevance evidence is located within individual documents. The 20% and 5% Step-wise results here are 46.4 and 45.3. They are distinct from the 46.1 and 45.4 entries in the stride-based QC sweep: the controlled experiment uses a full-token-trained checkpoint and exact per-document budgets, whereas the sweep reports Step-5 and Step-20 configurations with aggregate retention ratios of 20.0% and 5.2%. These are separate experimental settings, not repeated measurements of the same configuration.

Selection Strategy	**5%**	**10%**	**20%**
Encode → Select			
Last	42.7	43.9	45.0
Global Importance	43.2	44.3	45.1
Random	44.0	44.8	45.7
Stratified Random	45.1	46.0	46.3
Stratified Importance	45.2	46.1	46.4
Step-wise	45.3	46.7	46.4
Select → Encode			
Step-wise	–	43.8	–
Full-token reference		46.8	

Table 2: Matched-budget selection results (macro-average NDCG@10 $\times 100$) for a full-token-trained CompRank-QC.

Contextualized states versus raw tokens. Encode–Select encodes the full document before retaining selected KV states. Select–Encode removes unselected input tokens before encoding while preserving the selected positions' original position IDs and the query prefix. Using identical selected positions at 10% retention, Encode–Select achieves 46.7, compared with 43.8 for Select–Encode. This 2.9-point difference supports the contribution of full causal context to the retained representations; it does not by itself identify which omitted evidence those states encode. Together, these controls support Step-wise selection as a simple way to combine distributed coverage with contextualized states.

4.2 SCALING AND EXTRAPOLATION

We analyze CompRank-QC on TREC-COVID by comparing its full-token and Step-10 configurations with the generation-based Direct-List baseline. Models are trained on 30-document lists and evaluated on 30–500 candidates without further fine-tuning. This experiment uses a single A100 80GB GPU with batch size 1 and *does not reuse cached document states*: every request re-encodes its documents. Figure 2 reports both NDCG@10 and end-to-end latency per query, showing a clear difference in extrapolation behavior. Direct-List suffers a substantial performance drop as the candidate set grows, decreasing from 75.8 NDCG@10 at 30 documents to 59.7 at 500 documents. In contrast, both CompRank variants remain more stable. The full-token variant maintains competitive performance, while CompRank with Segment-10 reaches 79.5 NDCG@10 at 500 documents. This suggests that attention-based scoring is more robust than decode-ID generation in this long-list setting.

The degradation of Direct-List is likely related to its reliance on autoregressive decoding of document identifiers. Since the model is trained to generate identifiers within a limited range, increasing

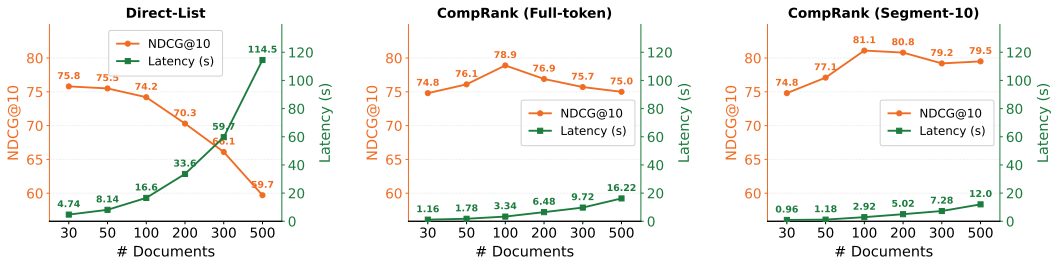

Figure 2: Scaling performance and end-to-end latency on TREC-COVID as the number of candidate documents increases. All models are trained on 30-document lists and evaluated up to 500 documents.

the number of candidates introduces distribution shift in the output space and may lead to invalid predictions, such as missing identifiers, duplicated identifiers, or out-of-range indices. A more detailed decoding error analysis is provided in Appendix D, where we observe that the model can maintain nearly error-free decoding when trained and evaluated on 30-document candidate sets, but the decoding error rate increases substantially when generalizing to larger candidate spaces (50–500 documents). This suggests that identifier-based autoregressive ranking struggles to generalize to unseen decoding sequences and larger output spaces. CompRank also demonstrates substantially lower end-to-end latency. Direct-List becomes increasingly expensive as the candidate set grows, reaching 114.5s per query at 500 documents. In comparison, CompRank (Segment-10) requires only 12.0s, corresponding to a $9.5\times$ speedup. Across different candidate sizes, CompRank achieves $4.9\times$–$9.5\times$ speedup over Direct-List. Token-level compression further improves efficiency: compared with CompRank (Full-token), Segment-10 reduces latency from 16.22s to 12.0s at 500 documents, yielding approximately $1.4\times$ additional speedup while maintaining stronger ranking performance.

4.3 Efficiency gains from compression and KV reuse.

We examine two complementary sources of efficiency: reducing document interactions through token compression and avoiding repeated document encoding through KV reuse. Table 3 reports end-to-end latency across candidate-list sizes.

Token compression. CompRank-QC consistently benefits from Step-10 compression across candidate-list sizes. At 500 candidates, latency decreases from 16.22s to 12.00s, an 26% reduction and approximately $1.35\times$ speedup. Compression reduces the document states accessed during scoring while preserving full query-conditioned document encoding. The remaining encoding cost explains why end-to-end savings are smaller than the reduction in retained states.

	CompRank-QC		CompRank-Cache
# Docs	Full	Step-10	Step-10 + Reuse
30	1.16	0.96	**0.160**
50	1.78	1.18	**0.185**
100	3.34	2.92	**0.437**
200	6.48	5.02	**0.648**
300	9.72	7.28	**0.829**
500	16.22	12.00	**1.231**

Table 3: Measured end-to-end latency (s/query) on TREC-COVID. Cache reuse includes state assembly and other online overhead; offline cache construction is excluded.

KV reuse. CompRank-Cache uses query-independent KV states, allowing retained candidate states to be assembled for online scoring. The online computation processes the instruction–query–decision suffix and includes state assembly and other online overhead; offline cache construction is excluded. As reported in Table 3, the measured online latency with Step-10 KV reuse is 1.231s per query for 500 candidates. This result illustrates the potential of reusing document representations across queries. Together, the results show that token compression reduces query-conditioned scoring latency, while CompRank-Cache supports online scoring from reusable document-side states.

5 Related Work

LLMs for Reranking Recent advances have established large language models (LLMs) as a dominant paradigm for reranking (Yin et al., 2026; Lu et al., 2026c; Fan et al., 2026). Existing approaches can be broadly categorized into pointwise (Lu et al., 2026a), pairwise (Qin et al., 2024), setwise (Zhuang et al., 2024), and listwise methods (Pradeep et al., 2023a). Pointwise methods evaluate each query–document pair independently by generating relevance scores, with representative models including MonoBERT (Nogueira et al., 2019), MonoT5 (Nogueira et al., 2020), and RankLLaMA (Ma et al., 2024). While effective, this paradigm incurs substantial redundant computation on large candidate sets. Pairwise and setwise methods model relative preferences among documents (Zhuang et al., 2024; Qin et al., 2024), but introduce additional overhead due to repeated comparisons. Listwise and in-context ranking approaches jointly process multiple candidates within a single context, enabling richer inter-document interactions. Early work demonstrated strong zero-shot performance using GPT-based rerankers (Sun et al., 2023), followed by adaptations to open-source LLMs such as Vicuna, Zephyr, and Mistral (Pradeep et al., 2023a;b), as well as structured frameworks like RankLLM (Sharifymoghaddam et al., 2025). To scale to long candidate lists, these methods often employ sliding windows or hierarchical strategies (Zhou et al., 2026), which reduce context length but introduce redundant computation across windows.

Efficient Rerankers Improving the efficiency of long-context reranking has attracted increasing attention. One line of work reduces decoding cost. Attention-based rerankers infer relevance directly from internal attention patterns without explicit generation (Chen et al., 2025; Zhang et al., 2025; Wang et al., 2026; Tran et al., 2026), while FIRST reduces listwise reranking to single-token decoding (Reddy et al., 2024). These methods avoid or shorten autoregressive ranking outputs, thereby reducing inference overhead. Another line of work reduces input or attention computation. BlockRank (Gupta et al., 2025) shows that inter-document attention is largely redundant and proposes block-wise attention for scalable in-context ranking. LongRanker (Zhou et al., 2026) studies efficient one-pass reranking under long-context settings. General long-context models also explore sparse attention patterns such as sliding-window or global-local attention (Beltagy et al., 2020; Zaheer et al., 2020), and recent KV-level compression methods reduce the number of tokens participating in attention (DeepSeek-AI, 2025; 2026). A complementary direction compresses or reuses document representations. HyperRAG (An et al., 2025), CompressedDoc (Déjean & Clinchant, 2025), and BlockDoc (Li et al., 2026) precompute or compress document representations for reuse. More closely related, Compress-then-Rank applies ranking-aware passage compression before listwise reranking (Zhi et al., 2026), and PE-Rank uses passage embeddings as compact surrogates for efficient listwise reranking (Liu et al., 2025). In contrast, CompRank preserves token-level document representations, compresses the subset of document tokens exposed to attention-based scoring, and performs decoding-free reranking by directly sorting attention-derived document scores.

6 Conclusion

We presented **CompRank**, a token-efficient reranking framework for long-list LLM-based reranking. CompRank reduces redundant computation by decoupling document representations, applying segment-wise token compression, and replacing autoregressive identifier decoding with attention-based scoring trained via a CopyNet-style objective. Experiments on seven BEIR datasets show that segment-wise compression preserves most of the full-token reranking performance while retaining only 10.2% of document tokens. The comparison across compression strategies further suggests that maintaining global document coverage is more effective than relying solely on final tokens or document-internal saliency estimates. Additional scaling experiments show that CompRank remains stable when evaluated on candidate lists up to 500 documents after training on 30-document lists, while substantially reducing end-to-end latency compared with generation-based listwise reranking. Overall, our results suggest that token-level compression, document representation decoupling, and training–inference alignment provide a practical direction for scalable decoding-free LLM reranking. Future work includes implementing fully optimized document-side KV caching and exploring adaptive query-conditioned token selection.

ETHICS STATEMENT

CompRank studies efficient LLM-based document reranking. Our experiments use publicly available retrieval benchmarks and training data, including BEIR and MS MARCO, and do not introduce or collect additional sensitive or proprietary data. As with other LLM-based retrieval systems, ranking errors or biases inherited from the underlying models and datasets may affect downstream retrieval results. We encourage appropriate evaluation and responsible deployment in real-world applications.

REPRODUCIBILITY STATEMENT

We will release the code and scripts for training, inference, compression, and evaluation of CompRank. The paper and appendix provide the model configurations, prompts, training hyperparameters, scoring and compression settings, hardware, and evaluation protocols required to reproduce the reported results. We also specify the experimental settings for both CompRank-QC and CompRank-Cache, including long-list scaling and KV-reuse evaluation.

REFERENCES

Yuwei An, Yihua Cheng, Seo Jin Park, and Junchen Jiang. Hyperrag: Enhancing quality-efficiency tradeoffs in retrieval-augmented generation with reranker kv-cache reuse, 2025. URL https://arxiv.org/abs/2504.02921.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer, 2020. URL https://arxiv.org/abs/2004.05150.

Shijie Chen, Bernal Jimenez Gutierrez, and Yu Su. Attention in large language models yields efficient zero-shot re-rankers. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=yzloNYH3QN.

DeepSeek-AI. Deepseek-v3.2-exp: Boosting long-context efficiency with deepseek sparse attention, 2025.

DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026.

Hervé Déjean and Stéphane Clinchant. Reranking with compressed document representation, 2025. URL https://arxiv.org/abs/2505.15394.

Yingqi Fan, Xuan Lu, Anhao Zhao, Junlong Tong, Ping Nie, Kai Zou, Yunpu Ma, Wei Zhang, and Xiaoyu Shen. minireranker: Efficient multimodal reranking through visual cache reuse and interaction sparsity, 2026. URL https://arxiv.org/abs/2606.10759.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. Incorporating copying mechanism in sequence-to-sequence learning. In Katrin Erk and Noah A. Smith (eds.), *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1631–1640, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1154. URL https://aclanthology.org/P16-1154/.

Nilesh Gupta, Chong You, Srinadh Bhojanapalli, Sanjiv Kumar, Inderjit S Dhillon, and Felix X. Yu. Scalable in-context ranking with generative models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=zj45hoQhjD.

Haohang Huang, Xuan Lu, Mingyi Su, Xuan Zhang, Ziyan Jiang, Ping Nie, Kai Zou, Tomas Pfister, Wenhu Chen, Wei Zhang, et al. Mmeb-v3: Measuring the performance gaps of omni-modality embedding models. *arXiv preprint arXiv:2604.23321*, 2026.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33: 9459–9474, 2020.

Minghan Li, Eric Gaussier, and Guodong Zhou. Efficient long-document reranking via block-level embeddings and top-k interaction refinement, 2026. URL https://arxiv.org/abs/2501.17039.

Qi Liu, Bo Wang, Nan Wang, and Jiaxin Mao. Leveraging passage embeddings for efficient listwise reranking with large language models. In *Proceedings of the ACM on Web Conference 2025*, pp. 4274–4283, 2025.

Xuan Lu, Sifan Liu, Bochao Yin, Yongqi Li, Xinghao Chen, Hui Su, Yaohui Jin, Wenjun Zeng, and Xiaoyu Shen. MultiConIR: Towards multi-condition information retrieval. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 13471–13494, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. URL https://aclanthology.org/2025.findings-emnlp.726/.

Xuan Lu, Haohang Huang, Rui Meng, Yaohui Jin, Wenjun Zeng, and Xiaoyu Shen. Rethinking reasoning in document ranking: Why chain-of-thought falls short. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=txmqENuRcc.

Xuan Lu, Haohang Huang, Rui Meng, Yaohui Jin, Wenjun Zeng, and Xiaoyu Shen. Tools are under-documented: Simple document expansion boosts tool retrieval. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=g9D9MgG7iW.

Xuan Lu, Kangle Li, Haohang Huang, Rui Meng, Wenjun Zeng, and Xiaoyu Shen. Beyond global similarity: Multi-conditional retrieval for fine-grained cross-modal understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9699–9709, 2026c.

Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. Fine-tuning llama for multi-stage text retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 2421–2425, 2024.

Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. Multi-stage document ranking with bert, 2019. URL https://arxiv.org/abs/1910.14424.

Rodrigo Frassetto Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. Document ranking with a pretrained sequence-to-sequence model. In *EMNLP (Findings)*, 2020.

Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. Rankvicuna: Zero-shot listwise document reranking with open-source large language models, 2023a. URL https://arxiv.org/abs/2309.15088.

Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. Rankzephyr: Effective and robust zero-shot listwise reranking is a breeze!, 2023b. URL https://arxiv.org/abs/2312.02724.

Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Le Yan, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, Xuanhui Wang, and Michael Bendersky. Large language models are effective text rankers with pairwise ranking prompting. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 1504–1518, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-naacl.97. URL https://aclanthology.org/2024.findings-naacl.97/.

Revanth Gangi Reddy, JaeHyeok Doo, Yifei Xu, Md Arafat Sultan, Deevya Swain, Avirup Sil, and Heng Ji. First: Faster improved listwise reranking with single token decoding. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 8642–8652, 2024.

Sahel Sharifymoghaddam, Ronak Pradeep, Andre Slavescu, Ryan Nguyen, Andrew Xu, Zijian Chen, Yilin Zhang, Yidi Chen, Jasper Xian, and Jimmy Lin. Rankllm: A python package for reranking with llms. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 3681–3690, 2025.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 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*, pp. 14918–14937, 2023.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 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)*, 2021. URL https://openreview.net/forum?id=wCu6T5xFjeJ.

Linh Tran, Yulong Li, Radu Florian, and Wei Sun. Contrastive retrieval heads improve attention-based re-ranking, 2026. URL https://arxiv.org/abs/2510.02219.

Juyuan Wang, Chenxing Wang, Yuchen Fang, Huiyun Hu, Junwu Du, Aolin Li, Haijun Wu, Jin Xu, Ligang Liu, and Dongliang Liao. Headrank: Decoding-free passage reranking via preference-aligned attention heads, 2026. URL https://arxiv.org/abs/2604.17237.

Bochao Yin, Xuan Lu, Zhengyu Qi, and Xiaoyu Shen. When should queries be decomposed? a stage-aware study of query decomposition for multi-condition retrieval, 2026. URL https://arxiv.org/abs/2606.08577.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297, 2020.

Wuwei Zhang, Fangcong Yin, Howard Yen, Danqi Chen, and Xi Ye. Query-focused retrieval heads improve long-context reasoning and re-ranking. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 23791–23805, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1214. URL https://aclanthology.org/2025.emnlp-main.1214/.

Zhewei Zhi, Yingyi Zhang, Yizhen Jing, Xianneng Li, Jianing Liu, Huajie Liu, and Yongliang Ding. Compress-then-rank: Faster and better listwise reranking with large language models via ranking-aware passage compression. 2026.

Changjiang Zhou, Ruqing Zhang, Jiafeng Guo, Maarten de Rijke, Fan Yixing, and Xueqi Cheng. Longranker: Efficient one-pass document reranking with long-context large language models. In *Proceedings of the ACM Web Conference 2026*, pp. 2004–2013, 2026.

Shengyao Zhuang, Honglei Zhuang, Bevan Koopman, and Guido Zuccon. A setwise approach for effective and highly efficient zero-shot ranking with large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 38–47, 2024.

A Implementation Details

A.1 CompRank Training Details

We provide the implementation details for the Mistral-7B configuration used in our main CompRank experiments. Each training instance contains 30 candidate documents. CompRank derives document scores from attention logits at a fixed Transformer layer. For Mistral-7B, we use zero-based layer index 20, corresponding to the 21st Transformer block, as the scoring layer. The decision-token set U contains a single token: the first supervised token in the completion, i.e., the "[" token following `Final Answer:`. For each candidate document, CompRank aggregates the softmax-normalized attention mass over its selected document states and then averages the resulting scores across attention heads. Since U contains a single decision token in our main experiments, the aggregation operator Agg directly uses the score of this token. Our implementation also supports multiple decision tokens with mean or log-sum-exp aggregation. Qwen2.5-7B follows the same training and scoring configuration as Mistral-7B, including zero-based layer index 20 as the scoring layer, the "[" token as the decision token, temperature $\tau = 0.05$, and the same optimization settings.

We train CompRank with a temperature-scaled candidate-level cross-entropy objective over the attention-derived document scores, using $\tau = 0.05$. Optimization uses 8-bit AdamW with a learning rate of 3.0×10^{-6}, weight decay of 0, cosine learning-rate scheduling, and a warmup ratio of 0.03. Training is performed for one epoch in bf16 precision with gradient checkpointing enabled. The training set contains 50,000 examples derived from the MS MARCO training split following the BlockRank data construction pipeline. We use a train–validation split ratio of 0.99, resulting in approximately 49,500 training examples and 500 validation examples. The per-device batch size is 1 with 4 gradient accumulation steps, giving an effective batch size of $4 \times G$, where G is the number of GPUs. Each candidate document block is truncated to at most 256 tokens.

A.2 CompRank Input Prompts

CompRank-QC and CompRank-Cache share the same prompt template. CompRank-QC additionally includes the query in the instruction before the candidate documents. Both variants retain the query block after the documents. The decision token after `Final Answer:` is used for attention-based document scoring.

A.3 Direct-List Prompt and Decoding Setup

Fig. 4 shows the prompt template for the Direct-List baseline. The model is instructed to generate a ranked list of document identifiers in descending order of relevance.

B Complete Compression Results

Tables 4 and 5 report the complete Mistral-7B QC and Cache sweeps. The main baseline comparison is Table 1; no duplicate main table is included here. Scores use NDCG@10 multiplied by 100.

C Ablation on Layer Selection and Decision Token Selection

C.1 Ablation on Transformer Layer Selection

CompRank relies on the internal attention weights of the language model to score documents. Since different Transformer layers capture different levels of abstraction, the choice of layer significantly impacts the ranking performance. We evaluate the Attention Accuracy across all 32 layers of the Mistral-7B model, fixing the decision token to `'['`.

As illustrated in Figure 5, the early layers (layers 0 to 13) exhibit very low Attention Accuracy (approximately 10–20%), indicating that these layers have not yet aggregated sufficient semantic information to distinguish document relevance. Performance rises sharply around layer 14 and remains generally strong throughout the middle-to-late layers (layers 15 to 31), with natural oscillation. We observe the highest peak at zero-based layer index 20, corresponding to the 21st Transformer

CompRank-QC Input Prompt

Prompt:

```
You will be given a query and a list of documents.
Each document will be formatted as CONTENT: | END.
Rank passages based on their relevance to the query.
Query: {query}
Documents:
CONTENT: {document_0} | END
CONTENT: {document_1} | END
...
CONTENT: {document_N-1} | END
====== Now let's start! ======
Find the most relevant document position.
Query: {query}
The following document(s) can help answer the query.
Final Answer:
[
```

CompRank-Cache Input Prompt

Prompt:

```
You will be given a query and a list of documents.
Each document will be formatted as CONTENT: | END.
Rank passages based on their relevance to the query.
Documents:
CONTENT: {document_0} | END
CONTENT: {document_1} | END
...
CONTENT: {document_N-1} | END
====== Now let's start! ======
Find the most relevant document position.
Query: {query}
The following document(s) can help answer the query.
Final Answer:
[
```

Figure 3: Input prompt templates for CompRank-QC and CompRank-Cache. CompRank-QC adds the query to the instruction before the candidate documents while retaining the query block after the documents. CompRank-Cache uses only the latter query block. In both variants, the decision token after `Final Answer:` is used for attention-based document scoring.

Prompt Template for Direct-List

System prompt:

```
You are RankLLM, an intelligent assistant that can
rank passages based on their relevance to the query.
```

User prompt:

```
I will provide you with {n} passages, each indicated
by a numerical identifier []. Rank the passages
based on their relevance to the search query:
{query}.

[1] {passage_1} [2] {passage_2} ... [n] {passage_n}

Search Query: {query}.

Rank the {n} passages above based on their relevance
to the search query. All the passages should be
included and listed using identifiers, in descending
order of relevance. The output format should be [] >
[], e.g., [2] > [1]. Only respond with the ranking
results, do not say any word or explain.
```

Figure 4: Prompt template used for the Direct-List baseline.

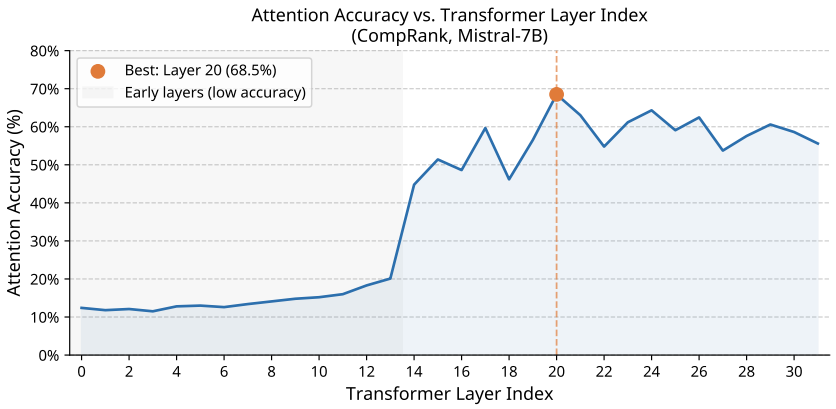

Figure 5: Attention Accuracy across all 32 Transformer layers of Mistral-7B. The metric measures the proportion of queries where the highest attention score is assigned to the positive document. The shaded region marks early layers with near-random performance.

Strategy	Token Ratio	Climate-FEVER	DBPedia	FiQA	NFCorpus	SciDocs	SciFact	TREC-COVID	Avg.
CompRank-QC (Mistral-7B)									
Full Tokens	100%	25.8	47.6	43.1	37.2	17.9	78.0	80.5	47.2
Last-k Compression									
Last-1	0.4%	25.6	46.2	39.4	34.6	17.0	62.9	77.3	43.3
Last-2	0.9%	26.1	46.6	40.8	34.4	17.2	64.3	77.3	43.8
Last-4	1.8%	26.3	47.9	38.6	34.9	17.6	66.2	78.2	44.2
Last-8	3.5%	25.1	47.2	41.5	35.8	17.1	66.6	78.4	44.5
Last-16	7.0%	24.9	47.0	41.5	35.9	17.1	68.2	78.1	44.7
Last-32	14.1%	24.5	47.1	42.5	35.5	17.6	70.3	77.4	45.0
Attention-based Top-k Compression									
Top-4	1.8%	25.7	44.3	38.0	33.6	17.0	65.5	75.4	42.8
Top-8	3.5%	25.5	45.0	38.8	34.2	17.1	66.8	75.9	43.3
Top-16	7.0%	25.1	45.4	39.6	34.6	17.2	68.4	76.2	43.8
Last-4 + Top-4	3.5%	28.0	44.3	37.3	33.8	17.7	64.5	75.5	43.0
Last-8 + Top-8	7.0%	24.5	46.4	38.6	33.4	17.6	66.1	75.9	43.2
Step-wise Compression									
Step-5	20.0%	25.7	46.0	41.2	36.6	18.4	76.3	78.2	46.1
Step-10	10.2%	25.5	46.6	41.7	36.2	18.5	77.0	81.1	46.7
Step-20	5.2%	25.0	46.3	42.0	35.8	18.2	72.7	77.7	45.4
Step-30	3.5%	27.0	45.8	39.3	34.9	17.3	70.6	77.2	44.6

Table 4: Mistral-7B CompRank-QC compression results on seven BEIR datasets. Scores are NDCG@10 $\times 100$; Avg. is the macro-average. Token ratio denotes retained contextualized document states. Step-10 is the configuration reported in Table 1.

Strategy	Token Ratio	Climate-FEVER	DBPedia	FiQA	NFCorpus	SciDocs	SciFact	TREC-COVID	Avg.
No Compression									
Full Tokens	100%	16.9	35.8	33.1	32.7	13.7	67.2	78.4	39.7
Last-k Token Compression									
Last-1	0.4%	16.7	34.4	29.4	30.1	12.8	52.1	78.8	36.3
Last-2	0.9%	17.2	34.8	30.8	29.9	13.0	53.5	78.8	36.9
Last-4	1.8%	17.4	36.1	28.6	30.4	13.4	55.4	79.7	37.3
Last-8	3.5%	16.2	35.4	31.5	31.3	12.9	55.8	79.9	37.6
Last-16	7.0%	16.0	35.2	31.5	31.4	12.9	57.4	79.6	37.7
Last-32	14.1%	15.6	35.3	32.5	31.0	13.4	59.5	78.9	38.0
Attention-based Top-k Compression									
Top-4	1.8%	16.8	32.5	28.0	29.1	12.8	54.7	76.9	35.8
Top-8	3.5%	16.6	33.2	28.8	29.7	12.9	56.0	77.4	36.4
Top-16	7.0%	16.2	33.6	29.6	30.1	13.0	57.6	77.7	36.8
Last-4 + Top-4	3.5%	19.1	32.5	27.3	29.3	13.5	53.7	77.0	36.1
Last-8 + Top-8	7.0%	15.6	34.6	28.6	28.9	13.4	55.3	77.4	36.3
Step-wise Compression									
Step-5	20.0%	16.8	34.2	31.2	32.1	14.2	65.5	79.7	39.1
Step-10	10.2%	16.6	34.8	31.7	31.7	**14.3**	64.2	**81.2**	**39.2**
Step-20	5.2%	16.1	34.5	32.0	31.3	14.0	61.9	79.2	38.4
Step-30	3.5%	18.1	34.0	29.3	30.4	13.1	59.8	78.7	37.6

Table 5: Mistral-7B CompRank-Cache compression results. NDCG@10 $\times 100$ and document-state retention ratios are reported. Avg. is the macro-average over datasets.

block, which achieves 68.5% Attention Accuracy. Consequently, we select zero-based layer index 20, corresponding to the 21st Transformer block, as the default scoring layer for all Mistral-7B experiments.

C.2 Ablation on Decision Token Selection

The second critical design choice is determining which token(s) in the prompt or generation sequence should be used to aggregate the attention scores over candidate documents. We compare the following five configurations, all evaluated at layer 20:

- **Query Token Aggregation:** Attention scores are aggregated over the individual query token positions only (excluding non-query tokens).
- **Query Block Aggregation:** Attention scores are averaged over all tokens in the entire query block.
- **`'Final','Answer'`:** Individual tokens from the answer-prefix template are used as the decision token.
- **`'['`:** The first generated token of the output sequence, which opens the listwise ranking output.

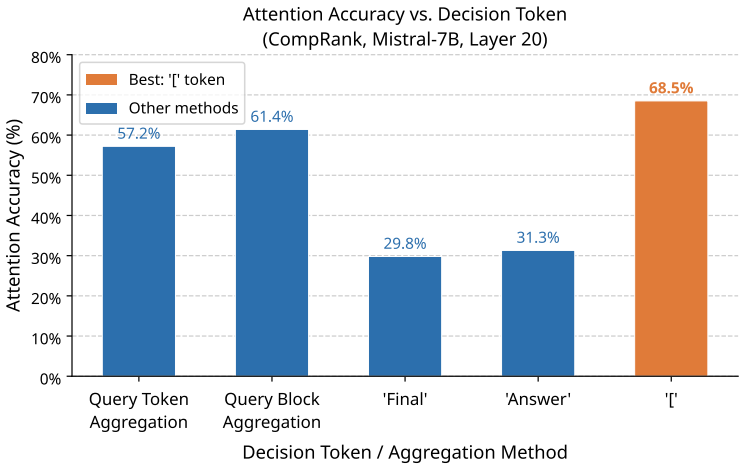

Figure 6: Attention Accuracy under different decision token and aggregation strategies (CompRank, Mistral-7B, Layer 20). Query Token Aggregation and Query Block Aggregation aggregate attention over multiple positions, while the remaining three use a single token as the decision point.

Figure 6 shows the Attention Accuracy for each configuration. Query Token Aggregation and Query Block Aggregation achieve 57.2% and 61.4% respectively, demonstrating that aggregating over query-side positions provides a reasonable signal. However, the answer-prefix tokens `'Final'` and `'Answer'` perform substantially worse (29.8% and 31.3%), suggesting that these positions do not concentrate relevance-discriminative attention. The highest Attention Accuracy of 68.5% is achieved by using the first generated token `'['` as the sole decision token. This token marks the exact moment the model begins producing its ranking output, and its attention distribution appears to be the most focused and informative for document relevance scoring. We therefore adopt `'['` as the default decision token in all CompRank experiments.

D Analysis of Decoding Errors in Direct-List

In Section 4.2, we observe that the generation-based Direct-List baseline degrades as the candidate list becomes longer. To better understand this behavior, we analyze its decoded identifier sequences on TREC-COVID when the number of candidate documents increases from 30 to 500. Since the Direct-List model is trained with 30-document candidate lists, we first examine whether it can correctly produce valid identifier sequences under the same list size, and then evaluate how the decoding behavior changes under longer candidate lists. We consider four types of decoding errors:

- **Duplicate IDs:** occur when the same document identifier is generated multiple times within a single ranked list.

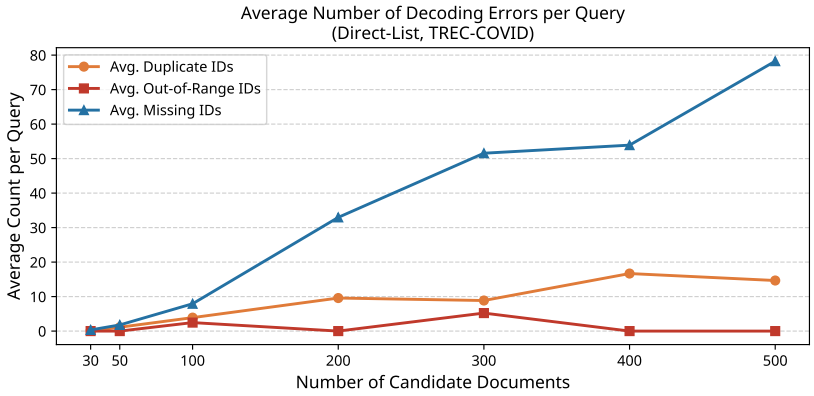

Figure 7: Average number of decoding errors per query for Direct-List on TREC-COVID as the number of candidate documents increases. The model produces valid identifier sequences under the 30-document setting seen during training, but decoding errors increase when evaluated on longer candidate lists.

- **Out-of-range IDs:** occur when the model generates an identifier outside the valid candidate range.
- **Missing IDs:** refer to valid candidate identifiers that are not generated in the output sequence.
- **Truncated Output:** occurs when the model terminates generation before producing the expected number of identifiers.

These errors directly affect listwise ranking quality because the generated sequence no longer corresponds to a valid permutation of the candidate set.

Table 6: Decoding error statistics of Direct-List on TREC-COVID (50 queries). Percentages indicate the proportion of queries exhibiting each error type.

List Size	Any Error	Duplicate IDs	Out-of-Range IDs	Missing IDs	Truncated Output
30	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)
50	46 (92.0%)	17 (34.0%)	0 (0.0%)	43 (86.0%)	32 (64.0%)
100	47 (94.0%)	21 (42.0%)	3 (6.0%)	46 (92.0%)	33 (66.0%)
200	50 (100.0%)	29 (58.0%)	0 (0.0%)	49 (98.0%)	36 (72.0%)
300	50 (100.0%)	31 (62.0%)	2 (4.0%)	48 (96.0%)	43 (86.0%)
400	50 (100.0%)	39 (78.0%)	0 (0.0%)	50 (100.0%)	36 (72.0%)
500	50 (100.0%)	33 (66.0%)	0 (0.0%)	50 (100.0%)	41 (82.0%)

Table 6 reports the proportion of queries affected by each error type, while Figure 7 shows the average number of errors per query. When evaluated with 30 candidates, which matches the training list size, Direct-List produces valid outputs without duplicate, out-of-range, missing, or truncated identifiers. However, once the candidate list grows beyond the training setting, decoding errors appear rapidly. At 50 candidates, 92.0% of queries already contain at least one error; from 200 candidates onward, all queries exhibit decoding errors. Missing identifiers are the dominant failure mode. As shown in Table 6, the proportion of queries with missing IDs increases from 86.0% at 50 candidates to 100.0% at 400 and 500 candidates. Figure 7 further shows that the average number of missing IDs per query grows substantially, reaching 78.2 under the 500-document setting. Duplicate IDs also become frequent, affecting 66.0% of queries at 500 candidates, indicating that the model struggles to maintain a valid one-to-one mapping between generation steps and candidate identifiers. Out-of-range IDs occur less frequently, but their presence shows that the generated sequence is not always constrained to the valid candidate space. Overall, these results suggest that decode-ID listwise reranking has limited extrapolation ability when the inference-time candidate list is substantially longer than the training-time list. Although the model can generate valid identifier sequences for the 30-document setting it was trained on, extending this pattern to 50–500 documents introduces a distribution shift in

Case Study: Duplicate and Missing IDs in Decode-ID Reranking

Dataset: TREC-COVID
Query: *How does the coronavirus differ from seasonal flu?*
Candidate List Size: 50
Expected Output Length: 50
Actual Output Length: 56

Duplicate IDs: `[3, 4, 12, 13, 17, 19, 26, 27, 35]`
Missing IDs: `[29, 33, 39]`

Raw Output Snippet:

```
[1] > [21] > [22] > [20] > [5] > [13] > [26] > [27]
> [35] > [4] > [17] > [19] > [3] > [12] > [24] >
[18] > [6] > [16] > [25] > [42] > [43] > [7] > [9]
> [14] > [8] > [31] > [32] > [28] > [34] > [44] >
[48] > [10] > [11] > [15] > [23] > [30] > [36] >
[37] > [38] > [40] > [41] > [45] > [46] > [47] >
[49] > [50] > [2] > [13] > [26] > [27] > [35] > [4]
> [17] > [19] > [3] > [12] <|im_end|>
```

Observation: The model initially generates many valid identifiers, but later repeats a block of previously generated IDs. This creates both duplicate IDs and missing candidate IDs, so the output no longer forms a valid permutation of the 50 candidate documents.

Figure 8: Qualitative example of decode-ID failure in Direct-List. The model repeats previously generated identifiers and omits several valid candidates when evaluated beyond its 30-document training setting.

Case Study: Severe Coverage Failure under Long Candidate Lists

Dataset: TREC-COVID
Query: *What are the mortality rates overall and in specific populations?*
Candidate List Size: 300
Expected Output Length: 300
Actual Output Length: 295

Duplicate IDs: `[192, 193, 194, ..., 201]`
Missing IDs: `[16, 17, 18, 19, 21, 22, 23, 24, 26, 27, ...]`

Observation: The model omits a large number of valid identifiers and repeatedly generates a subset of IDs. This indicates that identifier generation becomes increasingly unstable when the output sequence is much longer than those seen during training.

Figure 9: Qualitative example of severe coverage failure for Direct-List under a 300-document candidate list.

the output sequence space. As a result, the model increasingly fails to produce a valid permutation of candidate identifiers, providing additional motivation for CompRank's decoding-free attention-based scoring paradigm.

E LIMITATIONS

CompRank has several limitations. First, although we evaluate the method with both Mistral-7B and Qwen2.5-7B, broader validation across additional model families would help further characterize its generality. Second, our efficiency evaluation covers selected settings for document-side KV reuse; further work could study its behavior across a wider range of workloads and system configurations. Third, static segment-wise compression may be less effective when relevance evidence is sparse or highly localized, motivating the study of adaptive query-conditioned token selection.

F THE USE OF LARGE LANGUAGE MODELS (LLMS)

During the preparation of this manuscript, large language models were used solely for language polishing, including improving grammar, fluency, readability, and stylistic consistency. All research ideas, methodological design, experimental procedures, analyses, and conclusions were developed and validated by the authors. The use of LLMs did not extend to the generation of scientific claims, experimental results, or substantive research content.