Efficient Listwise Reranking with Compressed Document Representations

Anonymous ACL submission

Abstract

Reranking, the second stage of modern retrieval pipelines, is computationally expensive when performed by Large Language Models (LLMs). Existing efficiency-oriented approaches either rely on smaller models or restrict input length, both at the cost of effectiveness. Inspired by recent advances in soft document compression for retrieval-augmented generation (RAG), we introduce **RRK**, an efficient and effective listwise reranker that operates directly on multi-token compressed document representations rather than raw text. Trained via simple distillation from a strong teacher, RRK combines rich compressed representations with listwise reranking to deliver a highly efficient yet competitive system: despite its 8B-parameter backbone, it runs **3×–18× faster** than smaller rerankers (0.6–4B parameters) while matching or outperforming them in effectiveness. The advantage widens further on long-document benchmarks, where input compression yields the largest speedups.

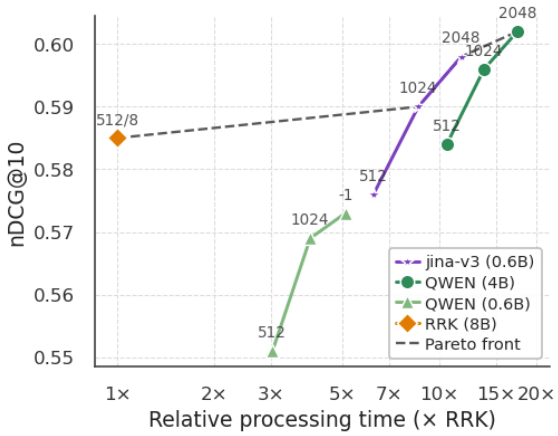

Figure 1: Effectiveness vs. efficiency on BEIR. Combining compressed document representations with listwise scoring, our 8B RRK reranker Pareto-dominates the baselines — lowest processing time at competitive nDCG@10. The x-axis is processing time relative to RRK (log scale); point labels give the document max-length in tokens.

1 Introduction

Information Retrieval (IR) is commonly understood as a two-stage process: a *first stage* that rapidly retrieves a set of candidate documents for a given query, followed by a more expensive *reranking* phase that refines the ordering by assessing relevance more carefully. The rise of retrieval-augmented generation (RAG) (Lewis et al., 2020) and, more recently, of LLM-based agents that retrieve and invoke tools, APIs, or external knowledge at scale (Qin et al., 2024) has made high-quality reranking more important than ever: when retrieved items are fed directly to an LLM as context, the precision of the top-ranked results has a direct impact on the quality of the generated answer or downstream decision.

The dominant approach to reranking has long been cross-encoders but Large Language Models (LLMs) have recently emerged as particularly strong rerankers, especially when used in a *listwise* setting where the model directly produces an ordering over a set of candidates (e.g. $d_2 > d_3 > d_1$) rather than scoring each document independently (Sun et al., 2023; Ma et al., 2023a; Pradeep et al., 2023), often pushing state-of-the-art effectiveness on standard IR benchmarks. This gain in effectiveness, however, comes at a substantial cost in efficiency: listwise LLM rerankers must process the concatenation of the query and all candidate documents in a single long context, making them considerably slower than traditional cross-encoder rerankers (Zhuang et al., 2023; Déjean et al., 2024).

Improving the efficiency of LLM-based rerankers without sacrificing their effectiveness is therefore an open challenge (Zhu et al., 2024), and several lines of work have begun to address this gap. A first direction reduces generation cost by producing the full candidate ranking in a single

forward pass using first-token logits (Gangi Reddy et al., 2024; Zhuang et al., 2023), cutting latency by roughly 50%.

A second line of work instead trains smaller but data-rich models (Wang et al., 2025). Yet another direction replaces textual document representations with IR-based embeddings: PE-Rank (Liu et al., 2025a) compresses each document into a single token to shorten the prompt, while E2Rank (Liu et al., 2026) uses a unified embedding model for both retrieval and reranking. However, single-token compression limits representational capacity, and query embeddings that depend on many documents constrain efficiency gains.

A complementary line of work has developed *soft* prompt compression methods to accelerate LLMs in long-context, dialogue, and retrieval-augmented generation (RAG) settings (Ge et al., 2023; Rau et al., 2024; Louis et al., 2025; Pilchen et al., 2025). These methods learn to map full document texts into compact multi-token representations, replacing lengthy inputs with only a handful of tokens in the LLM prompt while preserving downstream task performance. This raises a natural question: can such soft-compressed representations replace textual documents in reranking as well?

Contribution: We introduce **RRK**, an efficient and effective listwise reranker built on soft-compressed document representations. Unlike prior efficiency-oriented approaches that rely on IR-based single-token embeddings or smaller models, RRK leverages the soft compression literature to produce rich, multi-token document representations, and formulates listwise reranking directly over these compressed inputs. This drastically reduces the sequence length seen by the reranker, alleviating the core efficiency bottleneck of LLM-based reranking without sacrificing representational quality. Despite its 8B-parameter backbone, RRK matches or outperforms state-of-the-art rerankers in effectiveness while running **3×–18×** faster, as illustrated in Figure 1. Efficiency gains are even more pronounced on long-document benchmarks, where input compression yields the greatest speedups.

2 Related Work

2.1 Efficient Rerankers

LLMs have demonstrated strong potential as zero-shot rerankers: RankGPT (Sun et al., 2023), built on GPT-4, achieved state-of-the-art listwise reranking performance without any task-specific training. However, open-source LLMs of moderate size often yield uninformative outputs in this setting (Qin et al., 2023), motivating distillation-based approaches such as RankZephyr (Pradeep et al., 2023), which fine-tunes a Zephyr-7B model via knowledge transfer to match GPT-4-level performance. Zhuang et al. (2023) systematically compare pointwise, pairwise, and listwise reranking strategies and propose a setwise prompting method that improves zero-shot listwise effectiveness.

A key efficiency bottleneck of listwise LLM rerankers is the generation step: the original approach of Sun et al. (2023) explicitly generates the identifiers of reranked documents, which is costly. Gangi Reddy et al. (2024) already reduce this cost by deriving the ranked order from the logits of the first generated token, cutting latency by roughly 50%. More recent work eliminates generation altogether by computing reranking scores directly from query and document representations (Liu et al., 2026; Wang et al., 2025).

A parallel direction reduces input length by replacing full textual documents with compact embeddings. E2Rank (Liu et al., 2026) trains document embeddings jointly for first-stage retrieval and listwise reranking; at inference, documents are replaced by embeddings, though the query representation still depends on the raw text of multiple documents. PE-Rank (Liu et al., 2025a), most closely related to our work, replaces textual documents with first-stage retriever embeddings following Cheng et al. (2024), achieving a $4\times$–$6\times$ speedup depending on document length, but still relies on a sequential generation step. Liu et al. (2025b) take a complementary approach, applying a Matryoshka architecture to allow configurable reranker depth and width, achieving a $2\times$ speedup with sequence lengths of 1024. Wang et al. (2025) show that training a compact model (0.6B parameters) on large-scale data can further improve efficiency, and serves as the teacher in our distillation setup.

2.2 Soft Compression

Soft compression methods learn to encode documents into compact token representations that can be injected directly into an LLM prompt, replacing full retrieved texts with only a handful of tokens.

Chevalier et al. (2023) introduce Autocompressor, a recursive context compression method trained on a language modelling objective. It ap-

pends special compression tokens to the context and extracts their hidden states, enabling support for longer contexts and making it applicable to document compression in RAG-QA settings.

The In-Context Auto-Encoder (ICAE) (Ge et al., 2023) streamlines this idea by freezing the decoder, removing the recursive mechanism, and pretraining via a straightforward document auto-encoding objective.

xRAG (Cheng et al., 2024) takes a different angle: rather than learning new representations, it reuses existing retrieval embeddings and introduces a lightweight adapter that maps them into the input space of a frozen decoder LLM, reducing both storage and computational overhead.

PISCO (Louis et al., 2025) achieves the most aggressive compression via knowledge distillation: both the compressor and the decoder LLM are trained to reproduce the outputs of a teacher given raw text inputs, attaining a 16× compression rate with only a 0–3% accuracy drop across RAG-based question answering benchmarks. ARC-encoder (Pilchen et al., 2025) further demonstrates that with appropriate pretraining and only light fine-tuning, a compressor can be learned without altering the target model's architecture, reaching state-of-the-art compression performance.

RRK builds directly on PISCO, repurposing its compressed representations for reranking rather than generation, and is the first work to apply soft compression in this setting.

3 Reranking Compressed Representation

RRK[1] builds on the offline soft-compression framework PISCO (Louis et al., 2025) and consists of two jointly trained components: (i) a LoRA-finetuned compressor that maps each document to a fixed-size sequence of compressed token embeddings, and (ii) a LoRA-finetuned decoder reranker that assigns relevance scores to candidate documents given a query.

Formally, let $\mathcal{D} = \{d_i\}_{i=1}^{N}$ be a document collection and q a query. The compressor $f_{\theta_c} : d_i \mapsto \mathbf{c}_i$ maps each document to a sequence of l compressed embeddings $\mathbf{c}_i = (c_i^1, \ldots, c_i^l)$. Given a candidate set $D_k = \{d_1, \ldots, d_k\}$ from a first-stage retriever, the listwise reranker g_{θ_r} jointly processes the query together with all compressed candidates and produces a relevance score s_i for each document, $(s_1, \ldots, s_k) = g_{\theta_r}(q, \mathbf{c}_1, \ldots, \mathbf{c}_k)$.

[1]RRK: compressed version of **R**e**R**an**K**er

Document Compression. For each document d_i, a fixed set of l memory tokens $(m_1, \ldots, m_l)$ is appended, forming $(d_i; m_1, \ldots, m_l)$, processed by the compressor. The final hidden states of the memory tokens yield $\mathbf{c}_i = (c_i^s)_{s=1,\ldots,l}$. We fix $l = 8$, giving a 16× compression factor relative to the 128-token maximum used during PISCO training. Compression is performed **offline** after training, so document embeddings are precomputed once and reused at inference. We show in Section 6.1 that the compressor generalises naturally to longer documents (up to 2048 tokens).

Listwise Reranker. Following the input format of our Jina teacher (Wang et al., 2025), the decoder receives the query tokens **in textual form**, sandwiching the compressed document representations:

$$X = (q; \mathbf{c}_1; [\text{SEP}]; \mathbf{c}_2; [\text{SEP}]; \ldots; \mathbf{c}_k; [\text{SEP}]; q)$$

Repeating the query at both ends compensates for the lack of bidirectional attention. The query representation is taken from the final token $\mathbf{q} = H_{|X|}$, and each document d_i is represented by the hidden state $\mathbf{h}_i$ at the separator token following its compressed embeddings. The relevance score is $s_i = \cos(\mathbf{q}, \mathbf{h}_i)$.

Training. We optimise a RankNet loss (Burges et al., 2005; Gangi Reddy et al., 2024; Liu et al., 2026) over preference pairs $\mathcal{P}$:

$$\mathcal{L}(q, D_k) = \sum_{(i,j)\in\mathcal{P}} \log\left(1 + \exp\left(-\frac{s_i - s_j}{\tau}\right)\right)$$

with temperature $\tau = 1/8$. Critically, the compressor and reranker are trained *jointly*: gradients from the ranking loss flow back through the dependence of each s_i on every $\mathbf{c}_j = f_{\theta_c}(d_j)$, so the compressor learns representations specifically suited to reranking.

Complexity. A standard LLM reranker operates on sequences of length $|q| + k|d|$ with complexity $\mathcal{O}((|q| + k|d|)^2)$. RRK processes sequences of length $2|q|+k(l+1)$, yielding $\mathcal{O}((2|q|+k(l+1))^2)$. Since $l \ll |d|$ ($l = 8$ vs. $|d| \approx 200$ on most BeIR collections), this substantially reduces the quadratic attention cost. quadratic attention cost.

4 Experimental Framework

Distillation and Teacher. Training high-quality rerankers typically requires large annotated

datasets and carefully selected hard negatives (Cao, 2024). We instead adopt a distillation-based approach: a teacher reranker generates training signal, and several student models are trained on a small, annotation-free dataset — a base LLM, a ModernBERT baseline, and RRK. This enables a direct, controlled comparison between compressed and textual representations.

We retain SPLADE-V3 (Lassance et al., 2024) as first-stage retriever and Jina Reranker V3 as teacher, a listwise model built on Qwen 0.6B (Wang et al., 2025) that matches Qwen3-4B in effectiveness. Scaling to 8B yields only marginal gains (Zhang et al., 2025).

Training Set. Our primary training collection is MS MARCO (passage) (Bajaj et al., 2018). For each of the 0.5 million training queries, we retrieve the top 50 documents with SPLADE-V3, score them with the teacher reranker, and randomly sample 16 of these documents per query for training. No relevance judgements are required. To further boost performance, we supplement this with a second collection provided by Liu et al. (2026), derived from BGE-M3 (Zhang et al., 2025): it contains 150,000 queries, each paired with 16 documents scored by Qwen-32B used as a zero-shot teacher. We refer to **RRK-MS** as the model trained only on the MS MARCO collection, and to **RRK** as the model trained on both collections.

Backbone. We use Qwen-2.5 8B-instruct as our backbone, since the Qwen family provides multiple rerankers that serve as our teacher and our baselines, making the comparison more controlled. The compressor itself is obtained by training a PISCO model on this backbone, precisely following the methodology of Louis et al. (2025).

Pointwise Variant (RRK PW). We also train a pointwise variant of RRK for comparison purposes. At inference time, the compressed document embeddings $\mathbf{c}$ are loaded and passed to the Qwen-2.5 8B decoder, which has been finetuned for reranking. The decoder receives the query in its textual form together with the compressed representation of the document, and produces a scalar score: specifically, a linear layer maps the final-layer hidden state of the last token to a scalar. The model is trained with a mean squared error (MSE) loss to reproduce the scores assigned by the teacher reranker.

Training Details. All RRK models are trained for 2 epochs, as additional epochs did not yield significant improvements. We use 16 documents per query, a batch size of 2, a learning rate of 1×10^{-4}, and gradient accumulation over 16 steps; the full finetuning takes 48 hours on a single A100 GPU. The pointwise variant uses 4 documents per query and a batch size of 8, with comparable total wall-clock time. For the latency experiments, all computations were performed on a single A100 GPU and we used a batch size of 128, which corresponds to over 90% GPU memory utilisation; for the baseline models with an input length of 512, we increased this to 256. At inference, RRK requires access to the precomputed compressed representations, which we store in a Hugging Face dataset and fetch using the `select()` function; this loading step accounts for less than 10% of the total reranking time.

5 Evaluation

For evaluation, we use standard IR benchmarks: TREC Deep Learning 2019/2020 (Craswell et al., 2020, 2021) and BeIR (Thakur et al., 2021). For long-document ablations, we use the MS MARCO Document variants of TREC-DL 2019/2020. Following (Déjean et al., 2024; Liu et al., 2026), we exclude the BeIR ArguAna collection, which targets counter-argument retrieval. We rerank the top-50 candidates from SPLADE-v3, which yields higher effectiveness than using the BM25 top-100 while also improving efficiency by reranking fewer documents. We report nDCG@10 ($\times$100 for legibility) on all datasets, and measure efficiency using the *latency ratio*, defined as the relative processing time with respect to RRK, which is always the fastest. Listwise rerankers (RRK, Jina-V3) are marked with † in the tables.

As baselines, we focus on latency comparison rather than effectiveness: we include the publicly available Qwen-3 0.6B and 4B pointwise rerankers (Zhang et al., 2025), which are trained on substantially larger datasets (7M labeled and 12M synthetic examples) and may therefore retain an effectiveness advantage. We also compare to the teacher itself, the listwise Jina-v3 reranker, since non-compressed listwise rerankers are typically slower than their pointwise counterparts and this comparison directly quantifies the speedup from compression.

To isolate the impact of compression versus textual inputs, we also train two textual-input models under the same setting: Qwen2.5 8B (the

backbone used for RRK) and ModernBERT-large (Warner et al., 2024), a bidirectional encoder that is competitive in both effectiveness and efficiency. The Qwen2.5 8B textual model serves as an upper bound on effectiveness despite its impractical latency, while ModernBERT provides a strong encoder-only baseline for latency. Both are trained with a pointwise loss.

6 Results

Model	Len.	nDCG@10	Ratio	s/q ↓
RRK Rerankers (QWEN2.5-8B)				
RRK †	512	58.4	1	0.06
RRK	512	57.5	x3	0.21
Public Rerankers				
Jina-v3 † (QWEN3-0.6B)	512	57.6	x6	0.44
	1024	59.0	x8	0.53
	-1	59.8	x11	0.72
Qwen3-0.6B	512	55.1	x3	0.18
	1024	56.9	x4	0.24
	2048	57.3	x5	0.31
Qwen3-4B	512	58.4	x10	0.64
	1024	59.6	x14	0.84
	2048	**60.2**	x17	1.0
Fine-tuned models without compression				
ModernBert-Large	512	57.2	x2	0.13
Qwen2.5-8B	512	59.7	x20	1.26

Table 1: Evaluation and latency of our models for the BeIR collection (12 datasets). nDCG@10 and latency ratio relative to RRK. While public models perform the best when considering a long input length (2048), their latency compare badly to our RRK models using compressed document representation. †: listwise reranker. See also Figure 1.

The main results are presented in Table 1 while Figure 1 illustrates the efficiency-effectiveness trade-offs. We report the average processing time per query across the full BeIR dataset (31,828 queries): detailed BeIR results are given in Table 2.

Our key result is that the listwise RRK model is the fastest system—over $2\times$ faster than ModernBERT and up to $17\times$ faster than 4B parameter models—while maintaining strong effectiveness; RRK outperforms all baselines using 512 tokens as input, and outperforms the 0.6B rerankers in both effectiveness and efficiency in nearly all input length configurations. Figure 1 clearly illustrates RRK's speed advantage.

The fine-tuned models behave as expected: the Qwen2.5 8B textual model matches its teacher's effectiveness, while ModernBERT-large is substantially faster ($4\times$–$6\times$) with acceptable effectiveness.

The pointwise RRK variant achieves lower effectiveness than the listwise RRK and is less competitive overall: ModernBERT performs similarly but is twice as fast, and Qwen-3 0.6B is comparable. Nonetheless, despite its 8B size, its latency is remarkable relative to the 0.6B models.

For publicly released rerankers, effectiveness is highly sensitive to input length (-2 points), as is latency (about $2\times$ slower when increasing from 512 to 2048 tokens). This sensitivity is surprising on BeIR, where many sub-collections contain relatively short documents. We do not observe this phenomenon with our trained rerankers, which may be a consequence of using MS MARCO, a collection of short documents. For this reason, we present RRK results with an input length of 512.

Overall, combining compressed representations with a listwise objective yields a highly efficient reranker without sacrificing effectiveness.

6.1 Reranking Long Documents

We now focus on datasets where compression significantly enhances efficiency: specifically, datasets containing "long" documents (those exceeding 512 tokens, a typical length for encoder-only rerankers). We employ the MS-MARCO document collection along with the TREC 2019/2020 evaluation set for this purpose. The average document length is approximately 1000 tokens (Ma et al., 2023b). We reranker the top 50 documents provided by Splade-v3 as before. In this experiment, we test several input lengths for all models: our baselines as well as RRK: 512, 1024 and 2048. The results, presented in Table 3 and Figure 2, are surprising: the open rerankers underperform. Our RRK models, along with our trained ModernBert, demonstrate impressive effectiveness. Interestingly, RRK, although trained to compress short documents (128 tokens in the PISCO setting), effectively handles long documents and responds favourably to increased document lengths within these collections, a phenomenon not observed with the BeIR collection. RRK is capable of achieving a remarkable compression factor up to 256 (the best result for DL20 is obtained with an input length of 2048), being ten times faster than the pointwise Qwen 0.6B reranker in this setting.

Figure 2 further shows that long documents substantially increase the latency of non-compressed listwise rerankers: Jina-v3, our 0.6B-parameter

	SPLADE-V3 (retriever)	Jina v3† (teacher)	Qwen3 4B	ModernBERT (text)	Qwen-2.5 8B (text)	RRK	RRK†
TREC							
DL 19	72.3	75.3	76.5	76.3	**77.9**	77.5	75.8
DL 20	75.4	66.8	75.3	76.7	**79.0**	77.6	77.1
BeIR							
TREC-COVID	74.8	87.8	88.1	89.0	87.7	86.5	**89.3**
NFCorpus	35.7	36.7	38.6	38.1	**38.7**	**38.7**	37.2
NQ	58.6	72.5	**77.5**	66.0	72.3	66.3	70.2
HotpotQA	69.2	**80.3**	79.1	75.4	78.4	73.7	76.3
FIQA	37.4	46.1	46.9	47.6	**49.4**	47.5	45.3
Touché 2020-v2	29.3	32.8	32.5	**35.2**	32.7	31.1	33.6
Quora	81.4	**89.9**	84.9	86.0	89.2	86.8	87.6
DBPedia	45.0	48.7	48.3	**52.0**	48.7	49.9	51.2
SCIDOCS	15.8	22.2	**23.3**	19.5	21.3	19.6	21.1
FEVER	79.6	**91.6**	90.5	88.4	89.4	84.8	85.3
Climate-FEVER	23.3	33.7	**39.5**	25.3	28.3	27.0	28.1
SciFact	71.0	75.6	77.4	75.4	**77.6**	76.4	75.3
AVG	51.8	59.8	**60.2**	57.9	59.4	57.5	58.4

Table 2: Detailled evaluation for the TREC DL 19/20 and BeIR collections ($nDCG@10 * 100$). The top 50 of the SPLADE-v3 retrieved documents is used as input. For the public rerankers we report their best performances with an input length of 2048. Trained models' performances are reported with an input length of 512 (no improvement with longer input for those datasets). †: listwise rerankers.

teacher, becomes as slow as the pointwise Qwen 4B reranker. This confirms that listwise reranking, when used without compression, does not by itself provide sufficient efficiency benefits.

Our small ModernBert model performs exceptionally well in this context, offering competitive performance. We anticipated better results from the public rerankers, particularly Jina-V3, which was meticulously trained to handle long documents. We suspect that the underwhelming performance of these public rerankers may be due to the diverse training material utilized, whereas our dataset is domain-specific (MS MARCO). Note that these models also underperform for the passage TREC DL 19/20 collections (Table 2).

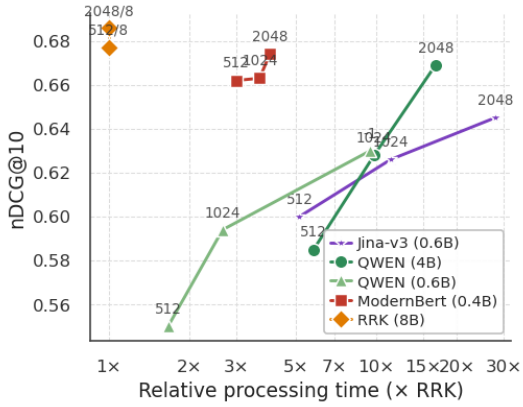

Figure 2: Effectiveness vs. Efficiency on MS-MARCO Document TREC 2020.

6.2 Ablations

We perform two types of ablations: the first examines the impact of the training dataset and teacher, while the second investigates the process of generating the PISCO compressor. While working to enhance the effectiveness of our RRK, one evident strategy was to incorporate additional training data by utilizing the training set provided by (Liu et al., 2026). This dataset comprises 150,000 queries and uses a different teacher model (Qwen3-32B). On its own, it yields worse results compared to our MS-MARCO dataset, even worst than a MS-MARCO subsample of the same size. We hypothesize that E2Rank's performance is attributed to its first-stage training with a much larger dataset (1.5M queries). However, as shown in Table 4, the combination of both datasets leads to significant improvement, rendering RRK a highly competitive reranker. Interestingly, this combination hinders the convergence of the pointwise RRK. We suspect that the diversity in score distributions from different teachers is the complicating factor, whereas the listwise approach,

Doc Length	RRK-MS†		RRK†		Jina-v3†		Qwen3 4B		Qwen3 0.6B		MBerT	
	nDCG	Lat	nDCG	Lat	nDCG	Lat	nDCG	Lat	nDCG	Lat	nDCG	Lat
MS MARCO DOCUMENT DL19												
512	68.5	1	70.6	1	62.6	x11	63.0	x13	60.3	x4	69.7	x6
1024	68.6	1	**72.1**	1	66.4	x24	66.9	x21	64.5	x5	70.0	x7
2048	68.6	1	72.0	1	68.3	x58	70.0	x37	65.5	x10	69.7	x8
MS MARCO DOCUMENT DL20												
512	66.1	1	67.7	1	60.0	x12	58.5	x13	55.0	x4	66.3	x6
1024	66.5	1	67.0	1	62.6	x24	62.8	x20	59.4	x6	66.4	x7
2048	67.0	1	**68.6**	1	64.5	x59	66.9	x35	63.7	x10	67.4	x8

Table 3: Reranking long documents: RRK performs very well for the MS Marco Document DL 19/20 collections, and its efficiency is even more remarkable (See also Figure 2). nDCG@10, latency ratio relative to RRK.

leveraging ranking information, can effectively utilize multiple teachers.

Training set	Nb. queries	nDCG@10
MS-MARCO	0.50M	57.7
MS-MARCO	0.15M	57.1
E2RANK	0.15M	55.6
MS MARCO + E2RANK	0.65M	58.4

Table 4: RRK evaluation on BeIR (12 datasets) based on MS Marco and E2RANK (Liu et al., 2026) datasets. The 0.15M MS-Marco subsamples produces better results than the E2Rank dataset while using a smaller teacher (0.6B/32B). Combining both improves by 0.7pt the nDCG@10 score.

Our second ablation study focuses on how the PISCO compressor is generated. We obtain the best results by fine-tuning a pretrained PISCO model (compressor and decoder). Table 5 shows that the original **frozen** PISCO compressor yields poor results (55.5), whereas training the compressor from scratch jointly with the reranker (using a compressor and a reranker LORA adapter) leads to competitive performance (57.7). This demonstrates that RRK requires high-quality compressed representations to remain competitive.

Compressor configurations	nDCG@10
Frozen PISCO compressor	55.5
Compressor from scratch	57.7
fine-tuned PISCO compressor	58.4

Table 5: Effectiveness of different PISCO compressor configurations.

6.3 Comparison to PE-Rank and E2RANK

The methods most closely related to RRK are PE-Rank (Liu et al., 2025a) and E2RANK (Liu et al., 2026), both of which also aim to improve reranking efficiency through compressed document representations. Table 6 compares RRK-MS and RRK with PE-Rank and E2RANK under their evaluation protocol: reranking the top 100 documents retrieved by BM25 (Robertson et al., 1996). We report results on the BeIR subsets used by both methods: TREC-COVID, SciFact, Web-Touché, NFCorpus, and DBPedia.

Model	nDCG@10	Lat. Ratio
RRK†-MS	55.4	1.0 (0.06)
RRK†	56.5	1.0 (0.06)
E2RANK† (MS)		
0.6B	53.9	x2.1
4B	56.2	x7.0
8B	56.8	x10.4
E2RANK† (BGE)		
0.6B	55.0	x2.1
4B	57.0	x7.0
8B	57.2	x10.4
PE-RANK† (7B,MS)	51.3	x7

Table 6: RRK compared to E2RANK (trained resp. with MS-Marco and BGE datasets) and PE-RANK . Latency estimated from (Liu et al., 2025a) and (Liu et al., 2025a)

Model	Input Length	Scoring Method
RRK	$2\|q\| + k(l+1)$	$\cos(q, d_i)$
PE-RANK	$\|q\| + k$	k decoding steps
E2RANK	$\|q\| + 20 \times \|d\| + k$	$\cos(q, d_i)$

Table 7: Model Input Length and Scoring Methods for RRK, PE-RANK and E2RANK. k: number of reranked documents, l: number of PISCO memory tokens

The approach most similar to RRK is PE-Rank. Although the two methods are conceptually related,

they differ in both their foundations and their underlying assumptions. Following Cheng et al. (2024), PE-Rank assumes that first-stage dense retrieval embeddings (Jina-Embeddings; Mohr et al., 2024) already provide effective document representations for reranking. In other words, it treats compression inherited from an IR model as a suitable proxy. In contrast, RRK builds on the literature on soft compression for LLMs, which suggests that (a) representing documents with multiple tokens is beneficial, and (b) compression is most effective when learned within the LLM itself. RRK therefore assumes that compression learned through a question-answering objective yields more fine-grained representations than first-stage retrieval embeddings. Our experimental results support this hypothesis.

Regarding efficiency, one might expect PE-Rank to be faster, since it takes only k tokens as input to rerank k documents, whereas RRK uses 8 tokens per document. However, PE-Rank still relies on a final, slow decoding step to generate the ranked sequence of document identifiers, which prevents it from being faster than RRK (Table 7).

E2RANK provides a different comparison point. It learns document embeddings shared by the first-stage retriever and the reranker. While this document representation is highly efficient, as it consists of a single embedding per document, its query representation is computationally expensive: to compute it, the top 20 documents (in text form) are concatenated with the query. As a result, the overall efficiency of the model is limited by the cost of processing roughly 20 documents. E2RANK's overall good effectiveness compared to PE-Rank and RRK is likely due to the use of the E5 dataset (Wang et al., 2024) a large collection (1.5M queries) during its first-stage training.

Built on a richer compressed representation using 8 compression tokens, RRK consistently outperforms PE-Rank in both effectiveness and efficiency. It also surpasses E2RANK in speed while achieving comparable effectiveness on the shared datasets.

Overall, these comparisons highlight a key distinction between the approaches. Although all three methods rely on compressed document representations for efficient reranking, PE-Rank and E2RANK inherit compression from fixed IR embeddings, whereas RRK derives a soft, task-adaptive compression directly from the internal representations of a language model. This richer compressed representation preserves effectiveness without the degradation often observed with IR-based compression (Cheng et al., 2024; Liu et al., 2025a).

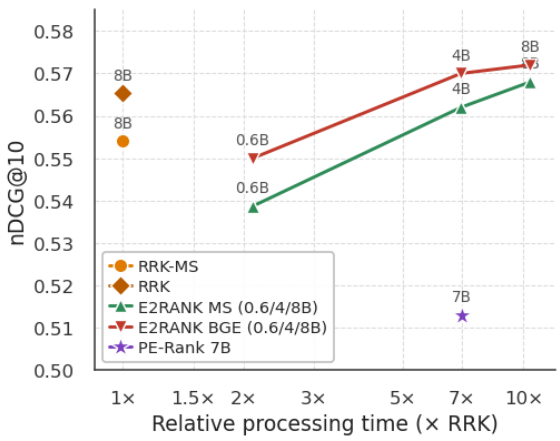

Figure 3: RRK against PE-Rank and E2RANK.

7 Conclusion

We introduced **RRK**, a listwise reranker that operates on soft-compressed document representations rather than raw text. By repurposing the PISCO compression framework— originally designed for retrieval-augmented generation—RRK shrinks the input sequence seen by the reranker while preserving the semantic richness needed for accurate ranking.

Our experiments yield two key takeaways. First, compressed representations enable an 8B-parameter model to run **3×–18× faster** than rerankers an order of magnitude smaller (0.6–4B parameters), while matching or outperforming them in effectiveness. The advantage is particularly pronounced on long documents, where input length dominates the reranking cost. Second, the *quality* of the compressed representation is critical: LLM-based soft compression yields fine-grained, expressive embeddings that retain ranking-relevant signal, whereas IR-based compression—constrained by retrieval objectives—loses important semantic detail. This explains the consistent gap we observe between RRK and prior efficiency-oriented approaches such as PE-Rank.

These findings position LLM-based soft compression as a promising direction for efficient reranking, and more broadly suggest that compression learned for one downstream task (here, RAG) can transfer effectively to others when paired with appropriate training objectives.

Limitations

First, the efficiency of RRK is mostly due to its tiny input length. This advantage holds as long as the query itself is short. Using datasets like the BRIGHT dataset (Su et al., 2024), where queries length is comparable to BeIR document length, breaks this advantage and makes the RRK model slow.

Secondly, it would be beneficial to employ smaller PISCO-based models instead of billion-sized ones as reranker. Unfortunately, our initial attempts to use smaller models, such as 1-4B parameter models, have not yet been successful. Using smaller models would lead to even better efficiency, and may reduce the index footprint (using smaller hidden dimensions).

Thirdly, the primary drawback of this method is the increased data storage requirement: each document requires $c \times h$ storage, where c is the number of memory tokens and h is the hidden dimension of the model (3584 in our case). For the MS-MARCO collection (8.8 million documents), a PISCO model with 8 memory tokens results in a storage size of 230 GB when using `float16` encoding. It is about the size of the first version of Colbert indexes on MS-MARCO (286GB cf Table 4 in Khattab and Zaharia (2020)). We believe that additional optimizations such as quantization, are likely to reduce this footprint.

References

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. 2018. Ms marco: A human generated machine reading comprehension dataset. Preprint, arXiv:1611.09268.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In Proceedings of the 22nd international conference on Machine learning, pages 89–96.

Hongliu Cao. 2024. Recent advances in text embedding: A comprehensive review of top-performing methods on the mteb benchmark. Preprint, arXiv:2406.01607.

Xin Cheng, Xun Wang, Xingxing Zhang, Tao Ge, Si-Qing Chen, Furu Wei, Huishuai Zhang, and Dongyan Zhao. 2024. xrag: Extreme context compression for retrieval-augmented generation with one token. arXiv preprint arXiv:2405.13792.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. Adapting language models to compress contexts. arXiv preprint arXiv:2305.14788.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2021. Overview of the trec 2020 deep learning track. Preprint, arXiv:2102.07662.

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

Hervé Déjean, Stéphane Clinchant, and Thibault Formal. 2024. A thorough comparison of cross-encoders and llms for reranking splade. Preprint, arXiv:2403.10407.

Revanth Gangi Reddy, JaeHyeok Doo, Yifei Xu, Md Arafat Sultan, Deevya Swain, Avirup Sil, and Heng Ji. 2024. FIRST: Faster improved listwise reranking with single token decoding. In Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 8642–8652, Miami, Florida, USA. Association for Computational Linguistics.

Tao Ge, Jing Hu, Xun Wang, Si-Qing Chen, and Furu Wei. 2023. In-context autoencoder for context compression in a large language model. arXiv preprint arXiv:2307.06945.

O. Khattab and Matei A. Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval.

Carlos Lassance, Hervé Déjean, Thibault Formal, and Stéphane Clinchant. 2024. Splade-v3: New baselines for splade. Preprint, arXiv:2403.06789.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In Advances in Neural Information Processing Systems, volume 33, pages 9459–9474.

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

Qi Liu, Yanzhao Zhang, Mingxin Li, Dingkun Long, Pengjun Xie, and Jiaxin Mao. 2026. E^2Rank: Your text embedding can also be an effective and efficient listwise reranker.

Zheng Liu, Chaofan Li, Shitao Xiao, Chaozhuo Li, Defu Lian, and Yingxia Shao. 2025b. Matryoshka reranker: A flexible re-ranking architecture with configurable depth and width. Preprint, arXiv:2501.16302.

Maxime Louis, Hervé Déjean, and Stéphane Clinchant. 2025. Pisco: Pretty simple compression for retrieval-augmented generation. Preprint, arXiv:2501.16075.

Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. 2023a. Fine-Tuning LLaMA for Multi-Stage Text Retrieval. arXiv preprint. ArXiv:2310.08319 [cs].

Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. 2023b. Fine-tuning llama for multi-stage text retrieval. Preprint, arXiv:2310.08319.

Isabelle Mohr, Markus Krimmel, Saba Sturua, Mohammad Kalim Akram, Andreas Koukounas, Michael Günther, Georgios Mastrapas, Vinit Ravishankar, Joan Fontanals Martínez, Feng Wang, and 1 others. 2024. Multi-task contrastive learning for 8192-token bilingual text embeddings. arXiv preprint arXiv:2402.17016.

Hippolyte Pilchen, Edouard Grave, and Patrick Pérez. 2025. Arc-encoder: learning compressed text representations for large language models. Preprint, arXiv:2510.20535.

Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023. RankZephyr: Effective and Robust Zero-Shot Listwise Reranking is a Breeze! arXiv preprint. ArXiv:2312.02724 [cs].

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, and 1 others. 2024. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In The Twelfth International Conference on Learning Representations (ICLR).

Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, Xuanhui Wang, and Michael Bendersky. 2023. Large Language Models are Effective Text Rankers with Pairwise Ranking Prompting. arXiv preprint. ArXiv:2306.17563 [cs].

David Rau, Shuai Wang, Hervé Déjean, and Stéphane Clinchant. 2024. Context embeddings for efficient answer generation in rag. Preprint, arXiv:2407.09252.

Stephen E Robertson, Steve Walker, MM Beaulieu, Mike Gatford, and Alison Payne. 1996. Okapi at trec-4. Nist Special Publication Sp, pages 73–96.

Hongjin Su, Howard Yen, Mengzhou Xia, Weijia Shi, Niklas Muennighoff, Han-yu Wang, Haisu Liu, Quan Shi, Zachary S Siegel, Michael Tang, Ruoxi Sun, Jinsung Yoon, Sercan O Arik, Danqi Chen, and Tao Yu. 2024. Bright: A realistic and challenging benchmark for reasoning-intensive retrieval.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Pengjie Ren, Dawei Yin, and Zhaochun Ren. 2023. Is ChatGPT Good at Search? Investigating Large Language Models as Re-Ranking Agent. arXiv preprint. ArXiv:2304.09542 [cs].

Nandan Thakur, Nils Reimers, Andreas Ruckl'e, Abhishek Srivastava, and Iryna Gurevych. 2021. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models. ArXiv, abs/2104.08663.

Feng Wang, Yuqing Li, and Han Xiao. 2025. jina-reranker-v3: Last but not late interaction for listwise document reranking. Preprint, arXiv:2509.25085.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024. Improving text embeddings with large language models. In Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 11897–11916.

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, Nathan Cooper, Griffin Adams, Jeremy Howard, and Iacopo Poli. 2024. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. Preprint, arXiv:2412.13663.

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

Yutao Zhu, Huaying Yuan, Shuting Wang, Jiongnan Liu, Wenhan Liu, Chenlong Deng, Haonan Chen, Zheng Liu, Zhicheng Dou, and Ji-Rong Wen. 2024. Large language models for information retrieval: A survey. Preprint, arXiv:2308.07107.

Shengyao Zhuang, Honglei Zhuang, Bevan Koopman, and Guido Zuccon. 2023. A Setwise Approach for Effective and Highly Efficient Zero-shot Ranking with Large Language Models. arXiv preprint. ArXiv:2310.09497 [cs].

A RRK Architecture

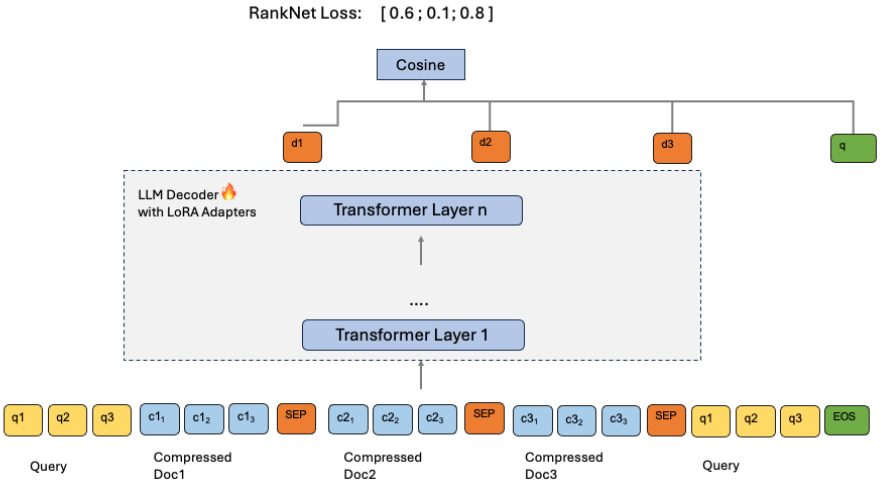

Figure 4: RRK Architecture Schema

B Full comparison between RRK, PE-Rank and E2RANK models

Model	TREC-Covid	NFCorpus	Touché	DBPedia	SciFact	Avg
E2 RANK (BGE)						
0.6B	79.2	38.6	41.9	42.0	73.4	55.0
4B	83.3	39.2	**43.2**	43.0	77.2	**57.2**
8B	84.1	39.1	42.2	43.4	77.5	**57.2**
E2 RANK (MS)						
0.6B	80.0	37.6	36.6	41.9	73.2	53.9
4B	84.9	39.3	35.4	43.6	**77.7**	56.2
8B	85.4	**39.6**	36.6	44.3	78.2	56.8
PE-RANK (MS)	77.5	36.4	33.1	40.1	69.4	51.3
RRK (MS)	81.5	37.9	39.1	44.7	74.1	55.6
RRK (MS+BGE)	**87.6**	38.9	35.6	**45.8**	74.8	56.5

Table 8: RRK compared to E2RANK (trained resp. with MS-Marco and BGE datasets) and PE-RANK.