Compress-then-Rank: Faster and Better Listwise Reranking with Large Language Models via Ranking-Aware Passage Compression

Zhewei Zhi[1,2,*], **Yingyi Zhang**[1,2,3,*], **Yizhen Jing**[4], **Xianneng Li**[1,2,†], **Jianing Liu**[4], **Huajie Liu**[4], **Yongliang Ding**[4]

[1]School of Economics and Management, Dalian University of Technology,
[2]Institute of Systems Engineering, Dalian University of Technology,
[3]Department of Data Science, City University of Hong Kong,
[4]Meituan.
xianneng@dlut.edu.cn

Abstract

Listwise reranking with Large Language Models (LLMs) has emerged as the state-of-the-art approach, consistently establishing new performance benchmarks in passage reranking. However, their practical application faces two critical hurdles: the prohibitive computational overhead and high latency of processing long token sequences, and the performance degradation caused by phenomena like "lost in the middle" in long contexts. To address these challenges, we introduce Compress-then-Rank (C2R), an efficient framework that performs listwise reranking not on original passages, but on their compact multi-vector surrogates. These surrogates can be pre-computed and cached for all passages in the corpus. The effectiveness of C2R hinges on three key innovations. First, the compressor model is pre-trained on a combination of text restoration and continuation objectives, enabling high-fidelity compressed vector sequences that mitigate the semantic loss common in single-vector methods. Second, a novel input scheme prepends embeddings of each ordinal index (e.g., [1]:) to its corresponding compressed vector sequence, which both delineates passage boundaries and guides the reranker LLM to generate a ranked list. Finally, the compressor and reranker are jointly optimized, making the compression ranking-aware for the ranking objective. Extensive experiments on major reranking benchmarks demonstrate that C2R provides substantial speedups while achieving competitive and even superior ranking performance compared to full-text reranking methods.

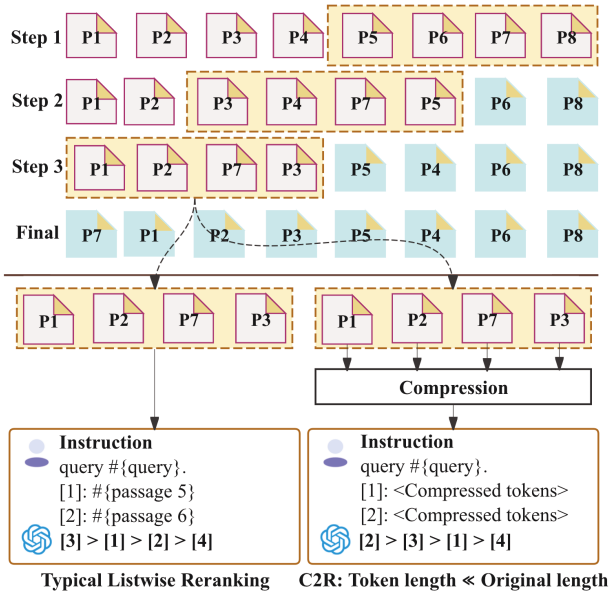

Figure 1: Comparison of our proposed C2R framework and typical listwise reranking, illustrated with eight candidates. Typical listwise reranking processes full-text passages (left), whereas our C2R framework leverages compact, compressed vector sequences for listwise reranking (right).

1 Introduction

Passage ranking is a fundamental task in information retrieval and natural language processing, aiming to precisely order a list of candidate passages based on their relevance to a user's query. It serves as a cornerstone in numerous applications, such as web search (Krestel and Fankhauser 2012), question answering (Mao et al. 2021), dialogue systems (Won et al. 2023), and the broader Retrieval-Augmented Generation paradigm (Gao et al. 2023). The dominant approach follows a two-stage "retrieve-then-rerank" pipeline. An initial, fast retriever recalls a broad set of candidates, which are then meticulously reranked by a more powerful but computationally intensive reranker to boost precision (Matveeva et al. 2006; Asadi and Lin 2013; Cambazoglu et al. 2010). In the reranking stage, while cross-encoder models were once the standard, the emergence of Large Language Models like GPT-4 (Achiam et al. 2023) has opened a new frontier. Researchers have explored various prompting strategies to leverage their superior text understanding and reasoning capabilities for passage reranking, primarily categorized as pointwise (Liang et al. 2022; Sachan et al. 2022), pairwise (Qin et al. 2023; MacAvaney and Soldaini 2023), and listwise (Sun et al. 2023; Pradeep, Sharifymoghaddam, and Lin 2023a) methods. Among these, listwise approaches such as RankGPT (Sun et al. 2023) have established new state-of-the-art performance. By generating

[*]Equal contribution.
[†]Corresponding author.

a globally optimal permutation for a list of passages simultaneously, they capture relative differences between passages that other methods overlook.

Despite their exceptional performance, listwise LLM rerankers are constrained by two critical challenges. First, despite the increasing context window sizes of modern LLMs, concatenating numerous passages into a single input often leads to the "lost in the middle" phenomenon (Liu et al. 2023). This limitation impairs their capacity to fully utilize and extract relevant information from the entire extended context, especially from its central parts, thereby degrading reranking precision. Conversely, employing more concise textual representations can alleviate this issue (An et al. 2024; Rau et al. 2025). Second, incorporating the full text of all passages into the prompt significantly escalates inference costs and computational overhead, resulting in high latency that is frequently prohibitive for practical, time-sensitive ranking scenarios (Chen et al. 2025).

To address these critical challenges, we propose Compress-then-Rank (C2R), a novel two-stage framework illustrated in Figure 1. At its core, this framework introduces a compression mechanism designed to simultaneously address the twin challenges of computational inefficiency and degraded ranking accuracy due to excessive input sequence lengths. Instead of processing full-text passages, C2R employs a compressor to transform each passage into a short, information-rich sequence of special vectors. This strategy yields a dual benefit. For efficiency, these compact surrogate vectors can be pre-computed and cached, which drastically reduces the reranker's input length and thereby mitigates prohibitive latency and computational costs. For effectiveness, representing passages as a sequence of vectors—rather than collapsing them into a single vector—mitigates the severe semantic loss that is common in single-vector methods. This design largely preserves the semantic information and helps alleviate the "lost in the middle" phenomenon associated with long contexts, thus safeguarding ranking performance. Finally, to effectively structure the input for the listwise ranking task, a novel hybrid input scheme is employed. Specifically, it prepends the embedding of each textual index marker (e.g., `[1]:`) to its corresponding compressed vector sequence. This approach serves to clearly delineate passage boundaries and provide explicit structural cues to the reranker LLM. The compressor and reranker are jointly optimized, this synergistic design enables C2R to perform high-quality listwise ranking directly on the compressed vector sequences, achieving the dual goals of drastically reducing the computational load while preserving the semantic fidelity required for accurate relevance judgment.

We evaluate our proposed framework, C2R, on popular reranking benchmarks, including the TREC DL and BEIR. Experimental results show that C2R achieves a compelling balance between efficiency and effectiveness. Notably, when reranking the top-100 candidates retrieved by BM25 on the DL19 dataset, C2R's input token count is reduced by approximately 92% compared to the equivalent full-text reranking approach, while simultaneously improving the nDCG@10 score by 4.9%. This demonstrates that our method drastically reduces computational costs while enhancing ranking quality. Our main contributions are summarized as follows:

- We propose C2R, an efficient and effective listwise reranking framework designed to reduce computational overhead while improving ranking accuracy.
- We introduce two novel C2R components: (1) a multi-vector compression method to represent passages, avoiding single-vector semantic bottlenecks and capturing nuanced information from long passages; and (2) a hybrid input scheme integrating textual indices with compressed vector sequences, which delineates passage boundaries and guides the reranker LLM.
- The compressor and reranker are jointly fine-tuned, creating a ranking-aware, end-to-end system tailored for ranking objective. Our extensive experiments demonstrate C2R's superior performance in efficiency and ranking accuracy.

2 Preliminaries

Before diving into the details of the proposed method, we formally define the listwise reranking task in the era of large language models.

Given a query q and a list of N candidate passages $P = \{p_1, p_2, \ldots, p_N\}$ from a first-stage retriever, the goal of listwise reranking is to find a permutation π^* that maximizes a ranking metric, such as nDCG@k.

An LLM-based listwise reranker, denoted as M_{rank}, processes passages by handling them in sublists. For a given sublist of w passages, $\{p_{i_1}, \ldots, p_{i_w}\}$, the reranker constructs a prompt P_{full} that concatenates some instructional text I, the query q, and the full text of each passage:

$$P_{\text{full}} = \text{PromptTemplate}(I, q, \{p_{i_1}, \ldots, p_{i_w}\}). \quad (1)$$

The LLM then generates a locally optimal permutation $\hat{\pi}_{\text{window}}$ for this sublist:

$$\hat{\pi}_{\text{window}} = M_{\text{rank}}(P_{\text{full}}). \quad (2)$$

To handle candidate lists where the total number of passages N exceeds the context window size w, methods like RankGPT (Sun et al. 2023) employ a sliding window mechanism. This technique iteratively applies the reranking process to overlapping windows of passages, allowing the most relevant ones to "bubble up" towards the top of the final list. While this approach is highly effective, its primary challenge is the length of the input prompt, $|P_{\text{full}}|$. This length is dominated by the sum of the passage texts, $\sum_{j=1}^{w} |\text{text}(p_{i_j})|$, which leads to significant computational overhead.

3 Methodology

3.1 Overview of the C2R Framework

The overall architecture of our proposed framework, C2R, is depicted in Figure 2. Designed to mitigate the efficiency bottleneck of listwise reranking with LLMs while maintaining high ranking effectiveness, C2R's core idea is to replace lengthy passage texts with compact, high-fidelity vector sequences before feeding them to the reranker LLM. This two-stage architecture primarily comprises a passage compressor, denoted as M_{comp}, and a listwise reranker LLM, M_{rank}.

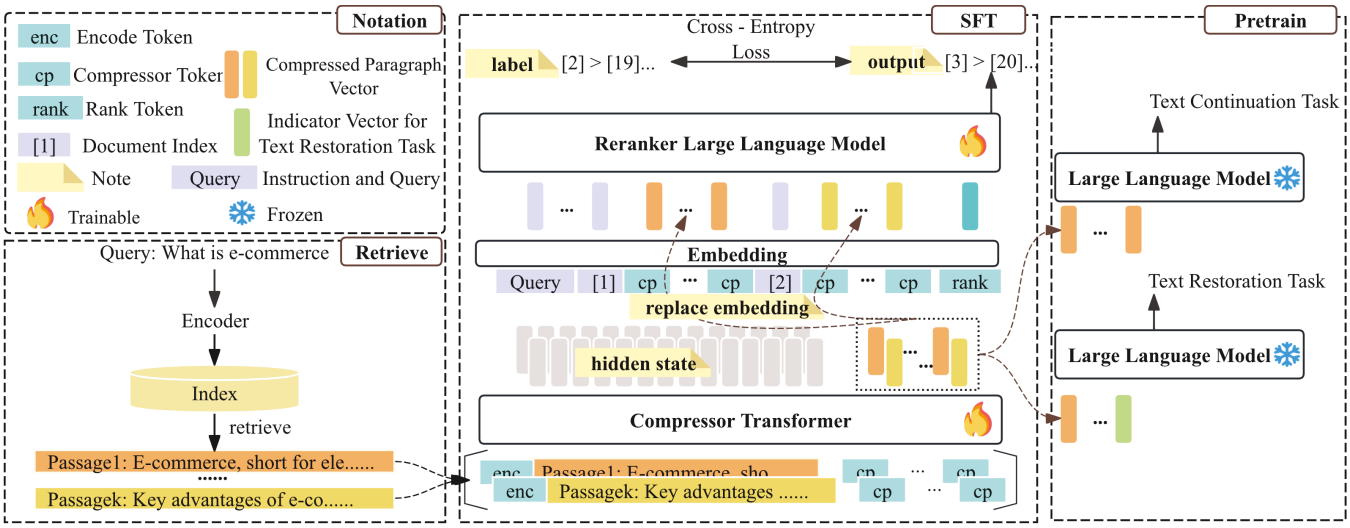

Figure 2: The architecture of the C2R framework. The model is trained in two phases: a pre-training phase (right) where the compressor learns to generate passage representations, and a supervised fine-tuning phase (center) where the compressor and reranker LLM are jointly optimized for listwise reranking using our proposed hybrid input scheme.

Following the standard sliding window paradigm (Sun et al. 2023), the C2R framework processes a sublist of candidate passages to produce a ranked permutation. This is achieved by transforming the reranking process to one that leverages compact compressed vector sequences, as detailed in the following sections.

Passage Compressor M_{comp}. Our compressor is a decoder-only Transformer, parameterized by θ_{comp}. It implements a function $\mathcal{C}$ that maps the text of a passage p_i to a sequence of l_c compressed vectors:

$$\mathbf{C_i} = \mathcal{C}(p_i; \theta_{\text{comp}}), \tag{3}$$

where $\mathbf{C_i} \in R^{l_c \times h}$, with h being the hidden dimension of the model. This compressed vector sequence is designed to preserve richer semantic information compared to single-vector methods.

Listwise Reranker M_{rank}. A powerful LLM with parameters θ_{rank} acts as the reranker. It operates on a structured input composed of the query, the instruction, and the compressed vector sequences $\{\mathbf{C_i}\}$. A hybrid input scheme integrates textual index markers (e.g., `[1]:`) by prepending them to their corresponding compressed vector sequences, guiding the LLM to generate a ranked list.

The effectiveness of C2R hinges on a carefully sequenced training strategy for these components. First, the compressor M_{comp} is independently pre-trained to optimize its vector representations, enabling a frozen LLM to perform both text restoration and continuation tasks based solely on these vectors. Subsequently, the pre-trained compressor and the reranker M_{rank} are jointly optimized end-to-end via supervised fine-tuning (SFT). This crucial second stage aligns the entire system with the final listwise ranking objective, adapting the compressor to produce ranking-aware compressed vector sequences and teaching the reranker to interpret them effectively.

3.2 Compressor Pre-training

To ensure the compressed vector sequences $\mathbf{C_i}$ are semantically faithful surrogates of the original passages, we pre-train the compressor M_{comp} using a dual-objective approach. We use a transformer as the encoder, while employing the reranker LLM itself as the decoder model.We keep the decoder's parameters θ_{rank} frozen during this pre-training phase, to perform both text restoration and continuation, a strategy inspired by recent context compression work (Ge et al. 2023; Rau et al. 2025).

The pre-training process begins by constructing an input sequence for M_{comp} for each passage p_i from a large corpus $\mathcal{P}_{\text{pretrain}}$. This is done by concatenating a special `[enc]` token, the passage text, and l_c special `[comp]` tokens, as depicted in Figure 2. M_{comp} then processes this input, and the final hidden states corresponding to the `[comp]` token positions are extracted to form the compressed vector sequence $\mathbf{C_i} = \mathcal{C}(p_i; \theta_{\text{comp}})$. Finally, $\mathbf{C_i}$ is passed to M_{rank}, and the compressor's parameters θ_{comp} are optimized through text restoration and text continuation tasks to refine the compressed vector sequences:

Text Restoration The objective is to reconstruct the original passage tokens $p_i = (t_1, t_2, \ldots, t_{|p_i|})$ from the compressed vector sequence $\mathbf{C_i}$. The restoration loss, $\mathcal{L}_{\text{restore}}$, is therefore defined as the negative log-likelihood of the ground-truth token sequence, conditioned on $\mathbf{C_i}$:

$$\mathcal{L}_{\text{restore}}(\theta_{\text{comp}}) = -\sum_{j=1}^{|p_i|} \log P(t_j | \mathbf{C_i}, t_{<j}; M_{\text{rank}}), \tag{4}$$

where t_j denotes the j^{th} token in the original passage.

Text Continuation To capture the broader context vital for effective reranking, where text restoration alone is insufficient, we introduce text continuation. This complementary task requires the model to generate a subsequent text chunk d'_i from C_i, compelling C_i to encode the passage's contextual trajectory. The continuation loss is formulated as:

$$\mathcal{L}_{\text{cont}}(\theta_{\text{comp}}) = -\sum_{j=1}^{|d'_i|} \log P(t'_j | \mathbf{C_i}, t'_{<j}; M_{\text{rank}}), \quad (5)$$

where t'_j denotes the j^{th} token in the continuation segment, and $t'_{<j}$ represents the preceding tokens $\{t'_1, \ldots, t'_{j-1}\}$.

Our pre-training employs a multi-task learning approach, stochastically alternating between text restoration and text continuation objectives. For each sample, only the selected task's loss is backpropagated to optimize θ_{comp}. This dual-objective training forces θ_{comp} to learn robust, compact representations in the frozen LLM's latent space, capturing both detailed content and contextual trajectory of the original passage.

3.3 Joint Supervised Fine-tuning for Reranking

Following pre-training, we jointly fine-tune the compressor M_{comp} and the reranker M_{rank} on a labeled dataset $\mathcal{D}_{\text{SFT}}$. Each instance in this dataset is a triplet consisting of a query q, a list of w candidate passages, and the corresponding target permutation string Y^* (e.g., `"[3] > [20] > ..."`), which serves as the supervision signal.

Hybrid Input Construction. For a query q and a window of w passages $\{p_{i_1}, \ldots, p_{i_w}\}$, we first construct a hybrid representation $\mathbf{S_j}$ for each passage p_{i_j} by concatenating its index embedding with its compressed vector sequence:

$$\mathbf{S}_j = E(\texttt{[i}_\texttt{j]:}) \| \mathcal{C}(p_{i_j}; \theta_{\text{comp}}), \quad (6)$$

where $\|$ denotes sequence concatenation. The final input embedding sequence for the reranker, X_{input}, is then formed by concatenating the instruction embedding $E(I)$, the query embedding $E(q)$, the sequence of all hybrid passage representations, and a final rank token embedding:

$$\mathbf{X}_{\text{input}} = E(I) \| E(q) \| \left(\bigoplus_{j=1}^{w} \mathbf{S}_j \right) \| E(\texttt{[rank]}), \quad (7)$$

where $E(I)$ denotes the embeddings of the instructional text tokens, $E(q)$ denotes the embeddings of the query tokens, and $E(\texttt{[rank]})$ is the embedding of a special token we introduce to prompt the model to generate the ranked output. As illustrated in Figure 2, the compressed vector sequences $\mathcal{C}(p_{i_j}; \theta_{\text{comp}})$ directly replace the embeddings of the `[CP]` placeholders.

Listwise Ranking Objective. The parameters θ_{comp} and θ_{rank} are jointly optimized by minimizing the cross-entropy loss between the reranker's predicted token sequence $\hat{Y}$ and the ground-truth permutation Y^*:

$$\mathcal{L}_{\text{SFT}}(\theta_{\text{comp}}, \theta_{\text{rank}}) = -\sum_{k=1}^{|Y^*|} \log P(Y^*_k | \mathbf{X}_{\text{input}}, Y^*_{<k}; \theta_{\text{rank}}). \quad (8)$$

To effectively leverage datasets of varying quality and scale, our SFT process unfolds in two sequential stages using the same loss function from Equation 8, but on different data.

Stage 1: Coarse-grained Alignment. First, we use a large-scale, lower-quality dataset to establish a foundational understanding of the task. The primary goal is to teach the compressor M_{comp} to generate ranking-aware representations and the reranker M_{rank} to interpret these vectors and generate the correct output format (e.g., `"[3] > [1] > [2]"`).

Stage 2: Fine-grained Refinement. Subsequently, we fine-tune the system on a smaller, high-quality dataset. With the basic input-output mapping already learned, this stage focuses exclusively on refining ranking precision and capturing nuanced relevance judgments.

This two-stage strategy fosters a powerful co-adaptation between the two models: the compressor learns to produce representations tailored for the ranking task, while the reranker simultaneously learns to effectively interpret these specialized input vector sequences.

4 Experiment

4.1 Experimental Setup

Datasets and Metrics. Our training process involved pre-training the compressor on 10 million Wikipedia passages (Petroni et al. 2020). Subsequently, Supervised Fine-Tuning was conducted in two stages, utilizing a total of approximately 242,397 instances derived from MS MARCO (Bajaj et al. 2016): first, a Coarse-grained alignment stage utilized 232,419 samples from PE-Rank (Liu et al. 2025b), where ranking results were generated by a MiniLM reranker; and second, a Fine-grained refinement stage employed 9,978 high-quality samples from RankZephyr (Pradeep, Sharifymoghaddam, and Lin 2023b), which featured RankGPT-4 generated ranking results and were selected for having exactly 20 candidates. We evaluated our models on in-domain TREC DL 2019 & 2020 (Craswell et al. 2020, 2021) and on 8 out-of-domain BEIR datasets (Thakur et al. 2021; Sun et al. 2023; Liu et al. 2025b), using nDCG@10 as the primary metric.

Baselines. We compare C2R against a suite of strong reranking models, categorized as follows:

- **Cross-encoders:** monoBERT (Nogueira et al. 2019) and monoT5 (Nogueira, Jiang, and Lin 2020).
- **Zero-shot Rerankers:** RankGPT (Sun et al. 2023) and TourRank (Chen et al. 2025).
- **Full-text Fine-tuned LLM Rerankers:** ListT5 (Yoon et al. 2024), RankVicuna (Pradeep, Sharifymoghaddam, and Lin 2023a), RankZephyr (Pradeep, Sharifymoghaddam, and Lin 2023b), and RankMistral (Liu et al. 2025b).
- **Compressed-token Fine-tuned LLM Rerankers:** PE-Rank (Liu et al. 2025b), which efficiently reranks by encoding each passage into a single embedding.

Model	Ret.	Covid	NFC	Touché	DBPedia	SciFact	Signal	News	Robust	Avg.
BM25	-	0.5947	0.3375	**0.4422**	0.3180	0.6789	0.3305	0.3952	0.4070	0.4380
Jina-Embeddings	-	0.6894	0.3143	0.2868	0.3332	0.6553	0.2576	0.3980	0.3823	0.4146
monoBERT	BM25	0.7001	0.3688	0.3175*	0.4187	0.7136	0.3144	0.4462	0.4935	0.4716
monoT5	BM25	0.8071	**0.3897***	0.3241*	0.4445	0.7657*	0.3255	0.4849	0.5671	0.5136
$\text{RankGPT}_{3.5}$	BM25	0.7667	0.3562	0.3618	0.4447	0.7043	0.3212	0.4885	0.5062	0.4937
RankGPT_{4}	BM25	**0.8551**	0.3847*	0.3857	0.4712	0.7495	0.3440	0.5289	0.5755	0.5368*
TourRank_{10}	BM25	0.8259*	0.3799*	0.2998	0.4464	0.7217	0.3083	0.5146	0.5787	0.5094
RankMistral	BM25	0.7800	0.3310	0.2746	0.3771	0.6622	0.3004	0.3710	0.3954	0.4365
ListT5-base	BM25	0.7830	0.3560	0.3340	0.4370	0.7410	0.3350	0.4850	0.5210	0.5090
ListT5-3B	BM25	0.8470	0.3770*	0.3360	0.4620	**0.7700**	0.3380	0.5320	0.5780	0.5300
PE-Rank	BM25	0.7772	0.3639	0.3306	0.4005	0.6938	0.3374	0.4970	0.4740	0.4843
C2R	BM25	0.8217	0.3839	0.3152	**0.5285**	0.7276	**0.4262**	**0.5623**	**0.5968**	**0.5452**

Table 1: Results (nDCG@10) of reranking top-100 passages on the BEIR benchmark. Ret. denotes the retrieval model used in the first stage. An asterisk (*) indicates no statistically significant difference between C2R and the baseline ($p \geq 0.05$ level) using a two-sided t-test. The overall best model is shown in bold, while the best model within each block is underlined.

Model	Ret.	TREC DL19	TREC DL20
BM25	-	0.5058	0.4796
Jina-Embedding	-	0.6594	0.6389
Supervised models trained with human annotation			
monoBERT	BM25	0.7050	0.6728
monoT5	BM25	0.7183	0.6889
Unsupervised LLM-based listwise models			
$\text{RankGPT}_{3.5}$	BM25	0.6580	0.6291
RankGPT_{4}	BM25	0.7559	0.7056
TourRank_{10}	BM25	0.7163	0.6956
LLM-based listwise models trained with distillation			
RankVicuna	BM25	0.6682	0.6549
RankZephyr	BM25	0.7420	0.7086
RankMistral	BM25	0.7173	0.6807
ListT5-base	BM25	0.7180	0.6810
ListT5-3B	BM25	0.7180	0.6910
PE-Rank	BM25	0.7048	0.6354
C2R	BM25	**0.7905**	**0.7791**

Table 2: Results (nDCG@10) of reranking top-100 passages on the TREC DL dataset. Ret. denotes the retrieval model used in the first stage. The best overall model is shown in bold, while the best in each block is underlined.

Implementation Details. Following previous work (Liu et al. 2025b; Chen, Pradeep, and Lin 2025, 2024), our C2R framework employs Mistral-7B-Instruct-v0.2 (Jiang et al. 2023) for both the reranker and the compressor, with these two components sharing the same underlying model weights to maximize parameter efficiency. The compressed vector sequence length is set to $l_c = 8$, and these passage vectors are stored in Elasticsearch. For sequences exceeding 500 tokens, we truncate them to the first 500 tokens prior to compression. For parameter-efficient adaptation, we utilize LoRA (Hu et al. 2022) in all training stages. The model is pre-trained for one epoch with a learning rate of 1×10^{-4} and subsequently fine-tuned for one epoch with a learning rate of 5×10^{-5}. The hyperparameters were determined based on empirical observations due to resource constraints. We adopt FlashAttention (Dao et al. 2022) and DeepSpeed ZeRO-2 (Rasley et al. 2020) to accelerate training and reduce memory usage. For evaluation, we rerank the top-100 passages retrieved for each query using the standard sliding window technique (window size 20, step 10). Further details on LoRA configuration, training infrastructure, and acceleration techniques can be found in Appendix A. See Appendix B for input prompt construction.

4.2 Effectiveness Analysis

The experimental results, presented in Table 1 and Table 2, show that C2R achieves strong performance across standard benchmarks. Notably, it outperforms established full-text listwise rerankers and single-token compression methods, demonstrating that the compress-then-rank paradigm can enhance, rather than compromise, ranking effectiveness. Remarkably, even with smaller model sizes, C2R surpasses the performance of larger models such as GPT-4.

In-Domain Performance on TREC DL. On the TREC DL tracks, when using BM25 (Robertson, Zaragoza et al. 2009), C2R's nDCG@10 scores of 0.7905 on DL19 and 0.7791 on DL20 are higher than other baselines. This finding is noteworthy, as it indicates that C2R, operating on compact vector surrogates, can achieve higher performance than models that process the full text.

Out-of-Domain Performance on BEIR. C2R demonstrates strong generalization on the BEIR benchmark. Under the BM25 setting, it achieves the highest average nDCG@10 among all models and ranks first on four of the eight datasets, indicating robust performance on out-of-domain tasks.

Model	nDCG@10	#Proc.	Avg. L_p	#Gen.
TREC DL19				
RankMistral$_p$	0.7196	9635.1	96.4	910.2
RankMistral$_s$	0.7050	6021.2	60.2	881.6
RankMistral$_t$	0.4543	653.3	6.5	865.1
PE-Rank	0.7048	**100.0**	**1.0**	**180.0**
C2R	**0.7905**	800.0	8.0	888.7
Covid				
RankMistral$_p$	0.7780	40038.5	400.38	986.5
RankMistral$_s$	0.7385	9702.1	97.0	929.6
RankMistral$_t$	0.7540	2636.4	26.4	916.9
PE-Rank	0.7772	**100.0**	**1.0**	**180.0**
C2R	**0.8217**	800.0	8.0	956.9

Table 3: Efficiency analysis for reranking the top-100 candidates retrieved by BM25. #Proc. is the total number of tokens in concatenated passages, Avg. L_p is the average token length per passage, and #Gen. is the number of generated tokens. Subscripts of RankMistral denote input forms: passage (p), summary (s), or title (t).

The Role of Ranking-Aware Compression. C2R's superior effectiveness stems from its design philosophy: jointly optimizing the compressor and reranker to create ranking-aware compressed vector sequences. This trains the compressor to extract key relevance signals while filtering noise, producing a condensed, signal-rich input that enables more precise reranking. The result is improved performance across benchmarks, showing that targeted, relevance-focused compression benefits ranking over processing full unfiltered text.

4.3 Efficiency Analysis

Our efficiency analysis, detailed in Table 3, demonstrates that C2R resolves the classic trade-off between effectiveness and computational cost. The full-text baseline, RankMistral$_p$, sets a strong performance bar but at a significant computational cost (e.g., 9635.1 tokens on TREC DL19). While other compression strategies, including using summaries or the single-vector approach of PE-Rank, reduce this token count, they generally fail to surpass the full-text model's performance, highlighting the typical dilemma.

C2R changes the trade-off by reducing computational overhead and improving accuracy. Compared to the full-text baseline, it cuts processed tokens by over 91% on TREC DL19 and nearly 98% on the long Covid dataset. Importantly, this efficiency gain comes with a significant performance boost on both benchmarks (e.g., from 0.7196 to 0.7905 on DL19), showing that C2R is not just faster but also a more effective reranking method.

4.4 Ablation Study

To fully understand the contribution of each key component in our C2R framework, we conduct a series of ablation studies. We analyze the impact of our proposed training stages and the effect of different compression lengths.

Setting	DL20	News	Robust	SciFact
Full Model	**0.7791**	**0.5623**	**0.5968**	**0.7276**
w/o Pre-train	0.7385	0.5161	0.5646	0.6529
w/o SFT Stage 1	0.5516	0.4342	0.4418	0.6802
w/o SFT Stage 2	0.5973	0.4467	0.4529	0.6578
w/o SFT	Fails	Fails	Fails	Fails

Table 4: Ablation study of C2R's training stages, measured by nDCG@10. The "Full Model" denotes the complete C2R framework. The "w/o SFT" variant fails to generate valid outputs.

Impact of Training Stages. To validate our training methodology, we conducted an ablation study with results presented in Table 4. The analysis confirms that every stage is critical. The entire Supervised Fine-Tuning process is indispensable, as the model (w/o SFT) completely fails without learning the task's required output format. Furthermore, removing the compressor pre-training stage (w/o Pre-train) leads to a significant performance drop across all datasets, highlighting its vital role in establishing a robust semantic foundation before task-specific adaptation.

The results also underscore the synergistic value of our two-stage SFT strategy. Omitting the initial coarse-grained alignment (w/o SFT Stage 1) causes a catastrophic decline (e.g., to 0.5516 on DL20), proving its necessity for bootstrapping the model's understanding of the task structure. Subsequently, removing the fine-grained refinement stage (w/o SFT Stage 2) also severely degrades performance, confirming its crucial role in honing ranking precision. This demonstrates that our carefully designed curriculum—from pre-training to coarse and then fine-grained tuning—is essential for achieving C2R's competitive results.

Impact of Compression Length. We investigate how compressed sequence length affects ranking performance, with results shown in Table 5. Counter-intuitively, we observe that shorter compressed lengths consistently and substantially outperform longer ones across all tested datasets. For instance, on DL19, reducing the length from 32 to 8 boosts the nDCG@10 score from 0.7382 to 0.7905. This suggests that more aggressive, ranking-aware compression more effectively distills salient relevance signals, implying that in ranking tasks, a small number of vectors can carry the necessary information. This highlights significant redundancy in full-length passages when assessing relevance. Exploring various compressed lengths and their trade-offs is a promising direction for future work.

Furthermore, we present the performance of C2R with different retrieval models, including BGE (Xiao et al. 2024), Jina (Günther et al. 2023), and BM25, in Appendix C to demonstrate its robustness across diverse retrievers.

4.5 Robustness to Initial Passage Ordering

The performance of listwise rerankers can be affected by the initial order of passages, a phenomenon known as positional bias (Stoehr et al. 2023; Tang et al. 2023; Yoon et al.

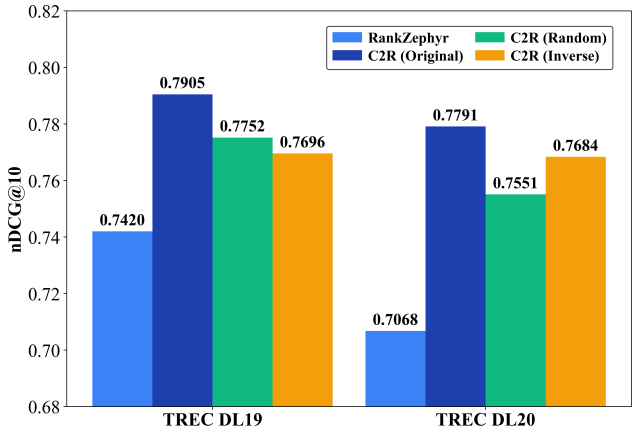

Figure 3: C2R's performance across different initial passage orderings, compared with the RankZephyr baseline. C2R demonstrates high robustness with minimal internal variation, while consistently outperforming the baseline.

Compressed Length	DL19	DL20	Covid	News
32	0.7382	0.7188	0.7254	0.4990
8	**0.7905**	**0.7791**	**0.8217**	**0.5623**

Table 5: Impact of different compressed sequence lengths on C2R's performance (nDCG@10) across various datasets.

2024). To assess C2R's robustness to this effect, we evaluated its performance using three different ordering schemes. We tested the Original sequence as provided by the BM25 retriever, a Random sequence from a shuffled retrieval list, and an Inverse sequence reversed from the original order.

As shown in Figure 3, C2R demonstrates notable robustness. On TREC DL19, the performance drop from the original order to the random and inverse orders is modest. A similar pattern is observed on TREC DL20. Critically, even under the most challenging 'Inverse' and 'Random' conditions, C2R's performance remains substantially higher than the strong RankZephyr baseline. This indicates that C2R effectively learns to assess passage relevance based on content rather than being overly reliant on initial positioning, a key attribute for a reliable reranking model.

4.6 Robustness to Passage Length Variation

Another key challenge in reranking is handling the wide variation in passage length. Our analysis in Table 6 shows that C2R's performance advantage over the full-text RankMistral model is not uniform, but correlates with passage length characteristics. The improvements are particularly significant for both very long and short passages.

For instance, on long-text datasets like Robust04 and News, where average token counts are highest, C2R achieves its most significant relative nDCG@10 gains of +50.9% and +51.6%, respectively. This demonstrates its effectiveness in mitigating the "lost-in-the-middle" problem common in full-text models. Concurrently, C2R also secures

Dataset	Avg. Len.	Median Len.	Max Len.	ΔnDCG@10
Covid	332.9	298.0	4846	+5.3%
NFCorpus	377.7	365.0	2804	+16.0%
DBPedia	96.2	97.0	2397	+40.1%
Touché-20	466.8	272.0	6607	+14.8%
SciFact	395.0	366.0	2167	+9.9%
Signal-1M	24.5	22.0	80	+41.9%
News	955.9	837.5	25590	+51.6%
Robust04	1013.8	804.0	361494	+50.9%

Table 6: Relationship between passage length characteristics and C2R's performance gain (ΔnDCG@10) over RankMistral on BEIR datasets. Lengths are measured in tokens.

massive improvements on short-text datasets like Signal-1M and DBPedia, with gains of +41.9% and +40.1%. This highlights its ability to distill crucial signals even from concise texts. We attribute this robust performance to our end-to-end training, which enables the compressor to act as a ranking-aware relevance filter that extracts salient information and discards noise regardless of document length.

5 Related Work

Listwise reranking has emerged as a dominant paradigm for applying Large Language Models to information retrieval (Sharifymoghaddam et al. 2025; Sun et al. 2023; Zhu et al. 2023; Rathee, MacAvaney, and Anand 2025). By jointly processing candidate passages to directly generate a relevance-ordered list (Ma et al. 2023; Sun et al. 2023; Tamber, Pradeep, and Lin 2023), it captures inter-passage relationships beyond pointwise or pairwise methods.

Recent work improves efficiency by rethinking the sliding-window inference framework, introducing tree-based (Yoon et al. 2024), top-down (Parry, MacAvaney, and Ganguly 2024), or truncated (Meng et al. 2024) architectures. Training-free approaches divide candidates into smaller groups for parallel ranking and then aggregate results using tournament-style scoring (Chen et al. 2025) or implicit pairwise synthesis (Dedov 2025). Other studies optimize intra-window processing via compact passage embeddings (Liu et al. 2025a,b), first-token probability cues (Gangi Reddy et al. 2024), or self-calibrated scoring for global consistency (Ren et al. 2025). Another active line distills strong rerankers (e.g., RankGPT) into smaller listwise models (Liu et al. 2024).

6 Conclusion

We introduce C2R, a novel framework enhancing LLM-based listwise reranking efficiency and effectiveness. Our experiments on TREC DL and BEIR benchmarks show C2R achieves substantial speedups and competitive, often superior, ranking performance compared to full-text models. This work validates that a carefully designed, ranking-aware compression can be more effective than processing the unfiltered text, paving the way for the practical deployment of powerful listwise rerankers in applications and opening new possibilities for other token-intensive NLP tasks.

Acknowledgments

This research was supported by the National Natural Science Foundation of China (NSFC) under Grant 72071029, 72231010, and the Graduate Research Fund of the School of Economics and Management of Dalian University of Technology (No. DUTSEMDRFKO1).

References

Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

An, S.; Ma, Z.; Lin, Z.; Zheng, N.; Lou, J.-G.; and Chen, W. 2024. Make your llm fully utilize the context. *Advances in Neural Information Processing Systems*, 37: 62160–62188.

Asadi, N.; and Lin, J. 2013. Effectiveness/efficiency tradeoffs for candidate generation in multi-stage retrieval architectures. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, 997–1000.

Bajaj, P.; Campos, D.; Craswell, N.; Deng, L.; Gao, J.; Liu, X.; Majumder, R.; McNamara, A.; Mitra, B.; Nguyen, T.; et al. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

Cambazoglu, B. B.; Zaragoza, H.; Chapelle, O.; Chen, J.; Liao, C.; Zheng, Z.; and Degenhardt, J. 2010. Early exit optimizations for additive machine learned ranking systems. In *Proceedings of the third ACM international conference on Web search and data mining*, 411–420.

Chen, Y.; Liu, Q.; Zhang, Y.; Sun, W.; Ma, X.; Yang, W.; Shi, D.; Mao, J.; and Yin, D. 2025. Tourrank: Utilizing large language models for documents ranking with a tournament-inspired strategy. In *Proceedings of the ACM on Web Conference 2025*, 1638–1652.

Chen, Z.; Pradeep, R.; and Lin, J. 2024. An Early FIRST Reproduction and Improvements to Single-Token Decoding for Fast Listwise Reranking. *arXiv preprint arXiv:2411.05508*.

Chen, Z.; Pradeep, R.; and Lin, J. 2025. Accelerating Listwise Reranking: Reproducing and Enhancing FIRST. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 3165–3172.

Craswell, N.; Mitra, B.; Yilmaz, E.; and Campos, D. 2021. Overview of the TREC 2020 deep learning track. arXiv:2102.07662.

Craswell, N.; Mitra, B.; Yilmaz, E.; Campos, D.; and Voorhees, E. M. 2020. Overview of the TREC 2019 deep learning track. arXiv:2003.07820.

Dao, T.; Fu, D.; Ermon, S.; Rudra, A.; and Ré, C. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35: 16344–16359.

Dedov, E. 2025. JointRank: Rank Large Set with Single Pass. In *Proceedings of the 2025 International ACM SIGIR Conference on Innovative Concepts and Theories in Information Retrieval (ICTIR)*, 208–217.

Gangi Reddy, R.; Doo, J.; Xu, Y.; Sultan, M. A.; Swain, D.; Sil, A.; and Ji, H. 2024. FIRST: Faster Improved Listwise Reranking with Single Token Decoding. In Al-Onaizan, Y.; Bansal, M.; and Chen, Y.-N., eds., *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 8642–8652. Miami, Florida, USA: Association for Computational Linguistics.

Gao, Y.; Xiong, Y.; Gao, X.; Jia, K.; Pan, J.; Bi, Y.; Dai, Y.; Sun, J.; Wang, H.; and Wang, H. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2(1).

Ge, T.; Hu, J.; Wang, L.; Wang, X.; Chen, S.-Q.; and Wei, F. 2023. In-context autoencoder for context compression in a large language model. *arXiv preprint arXiv:2307.06945*.

Günther, M.; Ong, J.; Mohr, I.; Abdessalem, A.; Abel, T.; Akram, M. K.; Guzman, S.; Mastrapas, G.; Sturua, S.; Wang, B.; et al. 2023. Jina embeddings 2: 8192-token general-purpose text embeddings for long documents. *arXiv preprint arXiv:2310.19923*.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; Chen, W.; et al. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2): 3.

Jiang, A. Q.; Sablayrolles, A.; Mensch, A.; Bamford, C.; Chaplot, D. S.; de las Casas, D.; Bressand, F.; Lengyel, G.; Lample, G.; Saulnier, L.; Lavaud, L. R.; Lachaux, M.-A.; Stock, P.; Scao, T. L.; Lavril, T.; Wang, T.; Lacroix, T.; and Sayed, W. E. 2023. Mistral 7B. arXiv:2310.06825.

Krestel, R.; and Fankhauser, P. 2012. Reranking web search results for diversity. *Information retrieval*, 15(5): 458–477.

Liang, P.; Bommasani, R.; Lee, T.; Tsipras, D.; Soylu, D.; Yasunaga, M.; Zhang, Y.; Narayanan, D.; Wu, Y.; Kumar, A.; et al. 2022. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*.

Liu, J.; Ma, Y.; Zhao, R.; Zheng, J.; Ma, Q.; and Kang, Y. 2025a. ListConRanker: A Contrastive Text Reranker with Listwise Encoding. *arXiv preprint arXiv:2501.07111*.

Liu, N. F.; Lin, K.; Hewitt, J.; Paranjape, A.; Bevilacqua, M.; Petroni, F.; and Liang, P. 2023. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*.

Liu, Q.; Wang, B.; Wang, N.; and Mao, J. 2025b. Leveraging passage embeddings for efficient listwise reranking with large language models. In *Proceedings of the ACM on Web Conference 2025*, 4274–4283.

Liu, W.; Ma, X.; Zhu, Y.; Zhao, Z.; Wang, S.; Yin, D.; and Dou, Z. 2024. Sliding windows are not the end: Exploring full ranking with long-context large language models. *arXiv preprint arXiv:2412.14574*.

Ma, X.; Zhang, X.; Pradeep, R.; and Lin, J. 2023. Zero-shot listwise document reranking with a large language model. *arXiv preprint arXiv:2305.02156*.

MacAvaney, S.; and Soldaini, L. 2023. One-shot labeling for automatic relevance estimation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2230–2235.

Mao, Y.; He, P.; Liu, X.; Shen, Y.; Gao, J.; Han, J.; and Chen, W. 2021. RIDER: Reader-Guided Passage Reranking for Open-Domain Question Answering. *arXiv preprint arXiv:2101.00294*.

Matveeva, I.; Burges, C.; Burkard, T.; Laucius, A.; and Wong, L. 2006. High accuracy retrieval with multiple nested ranker. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, 437–444.

Meng, C.; Arabzadeh, N.; Askari, A.; Aliannejadi, M.; and De Rijke, M. 2024. Ranked list truncation for large language model-based re-ranking. In *Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval*, 141–151.

Nogueira, R.; Jiang, Z.; and Lin, J. 2020. Document ranking with a pretrained sequence-to-sequence model. *arXiv preprint arXiv:2003.06713*.

Nogueira, R.; Yang, W.; Cho, K.; and Lin, J. 2019. Multi-stage document ranking with BERT. *arXiv preprint arXiv:1910.14424*.

Parry, A.; MacAvaney, S.; and Ganguly, D. 2024. Top-down partitioning for efficient list-wise ranking. *arXiv preprint arXiv:2405.14589*.

Petroni, F.; Piktus, A.; Fan, A.; Lewis, P.; Yazdani, M.; De Cao, N.; Thorne, J.; Jernite, Y.; Karpukhin, V.; Maillard, J.; et al. 2020. KILT: a benchmark for knowledge intensive language tasks. *arXiv preprint arXiv:2009.02252*.

Pradeep, R.; Sharifymoghaddam, S.; and Lin, J. 2023a. Rankvicuna: Zero-shot listwise document reranking with open-source large language models. *arXiv preprint arXiv:2309.15088*.

Pradeep, R.; Sharifymoghaddam, S.; and Lin, J. 2023b. RankZephyr: Effective and Robust Zero-Shot Listwise Reranking is a Breeze! *arXiv preprint arXiv:2312.02724*.

Qin, Z.; Jagerman, R.; Hui, K.; Zhuang, H.; Wu, J.; Yan, L.; Shen, J.; Liu, T.; Liu, J.; Metzler, D.; et al. 2023. Large language models are effective text rankers with pairwise ranking prompting. *arXiv preprint arXiv:2306.17563*.

Rasley, J.; Rajbhandari, S.; Ruwase, O.; and He, Y. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 3505–3506.

Rathee, M.; MacAvaney, S.; and Anand, A. 2025. Guiding retrieval using llm-based listwise rankers. In *European Conference on Information Retrieval*, 230–246. Springer.

Rau, D.; Wang, S.; Déjean, H.; Clinchant, S.; and Kamps, J. 2025. Context embeddings for efficient answer generation in retrieval-augmented generation. In *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining*, 493–502.

Ren, R.; Wang, Y.; Zhou, K.; Zhao, W. X.; Wang, W.; Liu, J.; Wen, J.-R.; and Chua, T.-S. 2025. Self-calibrated listwise reranking with large language models. In *Proceedings of the ACM on Web Conference 2025*, 3692–3701.

Robertson, S.; Zaragoza, H.; et al. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4): 333–389.

Sachan, D. S.; Lewis, M.; Joshi, M.; Aghajanyan, A.; Yih, W.-t.; Pineau, J.; and Zettlemoyer, L. 2022. Improving passage retrieval with zero-shot question generation. *arXiv preprint arXiv:2204.07496*.

Sharifymoghaddam, S.; Pradeep, R.; Slavescu, A.; Nguyen, R.; Xu, A.; Chen, Z.; Zhang, Y.; Chen, Y.; Xian, J.; and Lin, J. 2025. RankLLM: A Python Package for Reranking with LLMs. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 3681–3690.

Stoehr, N.; Cheng, P.; Wang, J.; Preotiuc-Pietro, D.; and Bhowmik, R. 2023. Unsupervised contrast-consistent ranking with language models. *arXiv preprint arXiv:2309.06991*.

Sun, W.; Yan, L.; Ma, X.; Wang, S.; Ren, P.; Chen, Z.; Yin, D.; and Ren, Z. 2023. Is ChatGPT good at search? investigating large language models as re-ranking agents. *arXiv preprint arXiv:2304.09542*.

Tamber, M. S.; Pradeep, R.; and Lin, J. 2023. Scaling down, litting up: Efficient zero-shot listwise reranking with seq2seq encoder-decoder models. *arXiv preprint arXiv:2312.16098*.

Tang, R.; Zhang, X.; Ma, X.; Lin, J.; and Ture, F. 2023. Found in the middle: Permutation self-consistency improves listwise ranking in large language models. *arXiv preprint arXiv:2310.07712*.

Thakur, N.; Reimers, N.; Rücklé, A.; Srivastava, A.; and Gurevych, I. 2021. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models. *arXiv preprint arXiv:2104.08663*.

Won, S.; Kwak, H.; Shin, J.; Han, J.; and Jung, K. 2023. Break: Breaking the dialogue state tracking barrier with beam search and re-ranking. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2832–2846.

Xiao, S.; Liu, Z.; Zhang, P.; Muennighoff, N.; Lian, D.; and Nie, J.-Y. 2024. C-pack: Packed resources for general chinese embeddings. In *Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval*, 641–649.

Yoon, S.; Choi, E.; Kim, J.; Yun, H.; Kim, Y.; and Hwang, S.-w. 2024. ListT5: Listwise Reranking with Fusion-in-Decoder Improves Zero-shot Retrieval. In Ku, L.-W.; Martins, A.; and Srikumar, V., eds., *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2287–2308. Bangkok, Thailand: Association for Computational Linguistics.

Zhu, Y.; Yuan, H.; Wang, S.; Liu, J.; Liu, W.; Deng, C.; Chen, H.; Liu, Z.; Dou, Z.; and Wen, J.-R. 2023. Large language models for information retrieval: A survey. *arXiv preprint arXiv:2308.07107*.