FIRST: Faster Improved Listwise Reranking with Single Token Decoding

Revanth Gangi Reddy[1*] **JaeHyeok Doo**[1,2*] **Yifei Xu**[1,2*] **Md Arafat Sultan**[3] **Deevya Swain**[1,2]
Avirup Sil[3] **Heng Ji**[1]
[1]University of Illinois Urbana-Champaign [2]Lapis Labs [3]IBM Research AI
{revanth3,jdoo2,yifeix5,deevyas2,hengji}@illinois.edu
arafat.sultan@ibm.com avi@us.ibm.com

Abstract

Large Language Models (LLMs) have significantly advanced the field of information retrieval, particularly for reranking. Listwise LLM rerankers typically showcase superior performance and generalizability over conventional supervised approaches. However, existing LLM rerankers can be inefficient as they provide ranking output in the form of a generated ordered sequence of candidate passage identifiers. Further, they are trained using the standard language modeling objective, which treats all ranking errors uniformly, potentially at the cost of misranking highly relevant passages. Addressing these limitations, we introduce FIRST[1], a novel listwise LLM reranking approach that leverages the output logits of the first generated identifier to directly obtain a ranked ordering of the candidates. We further utilize a learning-to-rank loss for this model, which prioritizes ranking accuracy for the more relevant passages. Empirical results demonstrate that FIRST accelerates inference by 50% while maintaining robust ranking performance, with gains across the BEIR benchmark. Finally, to illustrate the practical effectiveness of listwise LLM rerankers, we investigate their application in providing relevance feedback for retrievers during inference. Our results show that LLM rerankers can provide a stronger distillation signal compared to cross-encoders, yielding substantial improvements in retriever recall after relevance feedback.

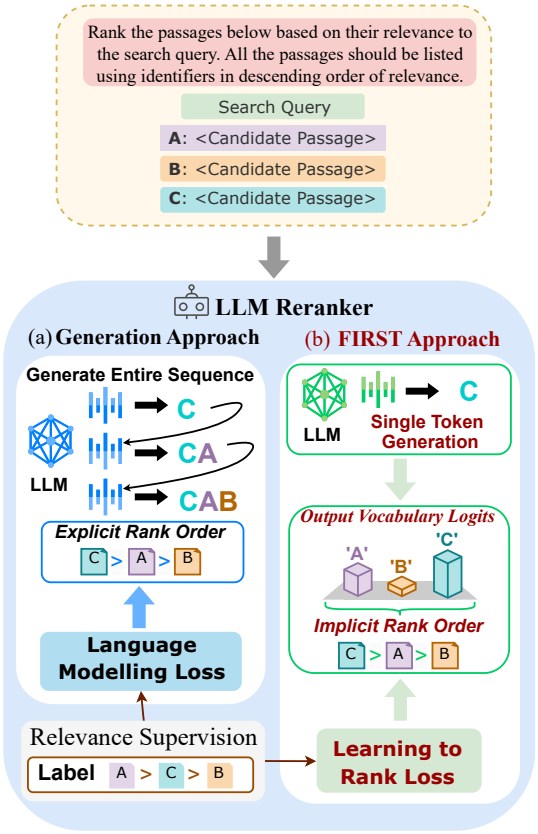


Figure 1: FIRST (b) directly ranks candidates using the output vocabulary logits for the first generated identifier, as opposed to the generation approach (a) of generating the entire ordered sequence. A learning-to-rank loss is incorporated during training to provide supervision to the model for ranking using single-token decoding.

1 Introduction

Given their vast linguistic knowledge and strong zero-shot capabilities (Wei et al., 2022), there has been a natural push to incorporate large language models (LLMs) into the search stack (Zhu et al., 2023; Wang et al., 2024). One of the core applications of LLMs in search involves ranking candidate passages for their relevance to a given query. Recent studies (Sun et al., 2023) have shown that instruction-tuned LLMs can outperform traditional supervised cross-encoders in zero-shot passage reranking (Nogueira et al., 2020; Zhuang et al., 2023b). In particular, listwise reranking approaches (Tang et al., 2023; Pradeep et al., 2023b) have received increased attention for their ability to score multiple passages simultaneously, as opposed to pointwise (Zhuang et al., 2023a,c) or pairwise (Qin et al., 2023) reranking, where scoring

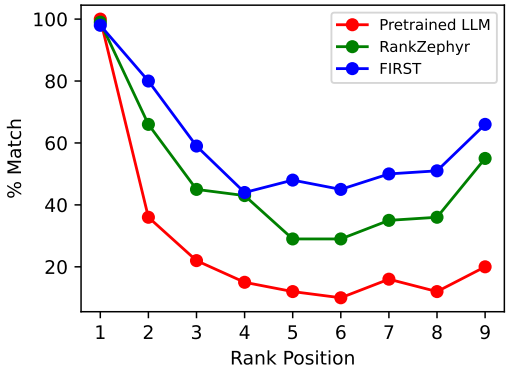
*Equal Contribution.
[1]https://github.com/gangiswag/llm-reranker

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 8642–8652
November 12-16, 2024

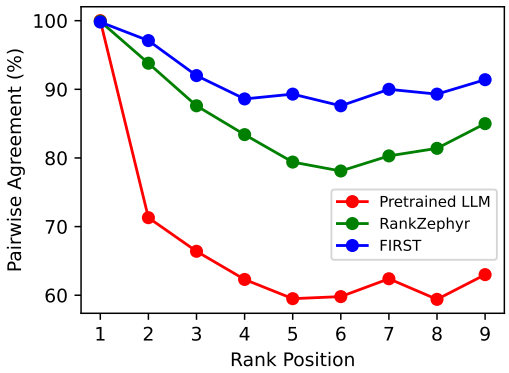
is performed in isolation. As Xian et al. (2023) have demonstrated, listwise reranking benefits from contextually comparing multiple passages at once, which helps calibrate relevance scoring better.

Listwise reranking with LLMs is typically framed as a generation task, where given a query and multiple candidate passages as input, the model outputs a ranked sequence of passage IDs. While Sun et al. (2023); Ma et al. (2023) use proprietary models, Pradeep et al. (2023a,b) demonstrate that open-source LLMs finetuned with GPT-3.5/GPT-4 (Achiam et al., 2023) annotated data can also achieve competitive performance. Pradeep et al. (2023b) introduce RankZephyr, which is trained using a standard language modeling objective, with the ranking sequence generated by GPT-4 as the target. Despite its promises, however, this approach has a number of key drawbacks. First, it involves generating entire sequences of passage IDs, which is arguably inefficient, and as we demonstrate through our study, is also unnecessary. Second, it penalizes errors uniformly across the ranking sequence – misjudging the rank of the most (and potentially only) relevant passage, for example, receives the same penalty as mistaking the ranks of two non-relevant passages. Intuitively, reranker training should prioritize accurately ranking top candidates over those that bear low relevance to the query.

The goal of this work is to overcome these limitations in LLM rerankers. Our investigation starts with the following question: Do the logits computed by existing LLM rerankers for their *first* generated identifier, which are meant to only predict the top-ranked candidate, also provide a calibrated estimate of the relative importances of *all* input candidates? In Figure 2, we show how the ranking indicated by the logits produced by RankZephyr (Pradeep et al., 2023b) in its first token position matches that of its fully generated ranking sequence. For each rank position $p \in [1, ..., 9]$, Figure 2a illustrates the degree of exact match between the two rankings in position p, and Figure 2b shows the average agreement over all different pairwise combinations of rank positions involving position p. We observe that RankZephyr's sequence-generation training objective also improves the quality of its logit-induced ranking by bringing it close to the sequence-based ranking, unlike the underlying pre-trained LLM. Crucially, this suggests that LLM rerankers can implicitly judge the relevance of candidate passages without needing to explicitly generate a ranking sequence. We seek to capitalize on this property to significantly accelerate inference process for listwise ranking, eliminating the need to generate a full sequence of IDs.



(a) Exact match



(b) Pairwise agreement rate

Figure 2: The degree to which the rank *generated* by an LLM reranker (RankZephyr (Pradeep et al., 2023b)) for a candidate agrees with the rank *implied by its computed logit* for the same candidate in the first (top-rank) token position, at different ranks. RankZephyr, originally fine-tuned with a sequence generation objective (in blue), shows a considerably higher similarity between the two above rankings than a pretrained LLM (in red).

To that end, we present FIRST[2], a novel approach that relies solely on the output logits of the first generated identifier to produce a listwise ranking of input candidates. FIRST employs a novel training strategy that directly incorporates a ranking loss into the supervision of LLM rerankers. The use of a learning-to-rank loss (Liu et al., 2009) also enables us to assign greater weights to more important ranks, unlike generation-based losses that treat all ranks in the output sequence uniformly. Figure 1 illustrates FIRST, which in our evaluation improves not only the efficiency of inference of LLM

[2]**F**aster **I**mproved **R**e-ranking with a **S**ingle **T**oken

rerankers due to single-token decoding, but also the quality of reranking by leveraging the more effective learning-to-rank supervision. As seen in Figure 2a, 2b, FIRST considerably increases exact match and pairwise agreement between the ranking sequence generated and the logit-based ranking. Experiments in §4.3 demonstrate that FIRST lowers the latency of LLM rerankers by 50%.

We further demonstrate the benefits of FIRST in downstream applications. Specifically, we study the impact of using LLM rerankers for pseudo-relevance feedback (ROCCHIO, 1971), wherein the output of a reranker is used to improve the recall of retrieval at inference. Prior work (Reddy et al., 2023; Sung et al., 2023) typically uses numeric point-wise scoring output from cross-encoders (Thakur et al., 2021a) as distillation supervision for relevance feedback. Here, we demonstrate (in §4.4) that a superior output from an LLM reranker, albeit in the form of an ordered sequence, can provide better relevance feedback that leads to greater improvement in retriever recall when distilled with ranking losses.

The main contributions of this work are:

- We introduce FIRST, a novel strategy for reranking with LLMs that obtains the ranking from only the output logits of the first generated identifier.
- By incorporating a learning-to-rank loss for supervision, FIRST improves ranking accuracy while lowering inference latency by 50%.
- Finally, we demonstrate the potential of LLM rerankers for relevance feedback, with improved retriever recall compared to using cross-encoders for inference-time distillation.

2 Related Work

2.1 Reranking with LLMs

Modern IR systems commonly employ a multi-stage pipeline, wherein an efficient initial retriever (Robertson et al., 2009; Karpukhin et al., 2020) selects a set of candidates from a vast corpus, which is then reranked by a more sophisticated reranker (Nogueira and Cho, 2019; Nogueira et al., 2020) to enhance precision. Methods leveraging cross-encoder models (Nogueira et al., 2020; Zhuang et al., 2023b) for rerankers have achieved notable success in improving ranking performance. Nonetheless, a principal limitation of such methodologies is their reliance on extensive in-domain human supervision, which leads to poor generalizability across different domains (Zhu et al., 2023). Recent efforts have explored mitigating this limitation by utilizing the zero-shot capabilities of LLMs for passage reranking (Ma et al., 2023; Sun et al., 2023). Building on this, Pradeep et al. (2023a,b) finetuned open-source LLMs to be capable of performing high-quality listwise reranking on par with proprietary models, such as GPT-4 (Achiam et al., 2023). However, existing works do not incorporate any traditional learning-to-rank strategies (Liu et al., 2009) when finetuning LLMs for listwise reranking. Further, they often overlook the considerable latency of reranking with LLMs. Our approach, FIRST, addresses both limitations by leveraging the output logits of the first generated identifier to directly obtain the rank order. FIRST successfully demonstrates that substantial efficiency gains are achievable without compromising accuracy in reranking with LLMs.

2.2 Learning to Rank

In IR literature, Learning to Rank (LTR) (Liu et al., 2009) aims to order items by their relevance to a particular query. LTR is an extensively explored research field, and multiple optimization techniques have been proposed that can be broadly categorized into three main approaches: pointwise, pairwise, and listwise. Given the item and query pair, pointwise approaches (Crammer and Singer, 2001; Li et al., 2007) determine relevance by a numerical score or binary judgment, which is later used for ranking. The pairwise approaches (Burges et al., 2005, 2006) measure the pairwise preferences between item pairs, being reportedly more effective than the pointwise method by capturing the relative importance of the items. Later, the training subjects were extended to a list of items, and the loss was defined over the entire item list (Cao et al., 2007; Xia et al., 2008; Taylor et al., 2008), allowing to obtain more fine-grained relative importance among the items. Recent studies (Nogueira et al., 2020; Zhuang et al., 2023b; Sun et al., 2023; Pradeep et al., 2023a,b) have applied pre-trained language models for passage reranking and observed significant performance gains. While Zhuang et al. (2023b) and Sun et al. (2023) employ LTR algorithms for finetuning, they only consider it for pointwise ranking. On the other hand, our approach adopts LTR algorithms for finetuning listwise LLM rerankers.

2.3 Listwise Reranking

Early exploration of leveraging pre-trained language models for document reranking relied on pointwise ranking (Sachan et al., 2022; Cho et al., 2023; Zhuang et al., 2023b). This involves extracting the generation probability of a relevance token, such as 'true' or 'yes', from the model when asked to determine the document's relevance to a query. Despite their supremacy over supervised ranking methods based on cross-encoders (Nogueira et al., 2020; Zhuang et al., 2023b), the isolated scoring mechanism of pointwise rerankers makes it difficult to calibrate relevance (Xian et al., 2023). Recent works (Ma et al., 2023; Sun et al., 2023) adopted listwise reranking to generate the ordered list of candidates directly, without needing any intermediate relevance scores. Compared to pointwise or pairwise counterparts (Qin et al., 2023), listwise reranking requires fewer runs as it takes multiple documents into account for a single window. When reranking multiple candidates making the prompt size more than max allowed input context length, listwise reranking adopts a sliding window strategy (Sun et al., 2023) with a fixed window and step size. However, due to the computationally demanding nature of LLMs, the improved results from listwise reranking come at the expense of increased latency. Recent work has tackled the latency problem of listwise reranking through efficient processing of candidate passages. Meng et al. (2024) introduced ranked list truncation, which optimizes the process by trimming reranking candidates, allowing for variable-length candidate lists that can be adapted per query. Parry et al. (2024) propose top-down partitioning, which introduces a parallelizable algorithm that effectively reduces redundancy in inference calls. Our method, FIRST, reduces the latency for each window in listwise reranking by lowering the number of output tokens required to be generated to one. FIRST complements existing strategies like ranked list truncation and top-down partitioning as each method targets a distinct yet complementary aspect of the listwise reranking workflow. We leave the empirical investigation of stacking these approaches together as an important direction for future work.

3 Methodology

In this section, we first discuss the fundamentals of listwise LLM reranking (§3.1). We then present FIRST, our own novel approach to the task (§3.2).

3.1 Listwise Reranking with LLMs

Given a list of retrieved passages $\mathcal{P} = \{p_1, p_2, ..., p_n\}$, the task of a reranker is to return k passages that are the most relevant to a query q. Due to input size limits, listwise reranking with LLMs often adopts a sliding window strategy with a window size of m passages ($m < n$) and a step size s (Sun et al., 2023). For each window, passages are denoted by unique identifiers t_i; the LLM reranker generates as output a sequence of identifiers in decreasing order of their relevance (e.g., $t_1 > t_3 > t_2$). The global process operates by first ranking the last m documents and then iteratively sliding the processing window s positions at a time until the beginning of the list is reached (Sun et al., 2023).

Recent work (Pradeep et al., 2023a,b) has drawn supervision for open-source listwise LLM rerankers (Tunstall et al., 2023) from larger proprietary models, such as $\text{GPT}_{3.5}$ and GPT_4. The relevance supervision in such cases comes in the form of a generated sequence $y = [y_1] > [y_2] ... > [y_m]$, where y_i is the identifier of a document that has been judged more relevant to the query q than y_j, for every $m \geq j > i$. The reranker is then trained with a language modeling objective, minimizing the error in predicting the true next token in the generation sequence:

$$\mathcal{L}_{LM} = -\sum_{i=1}^{|y|} \log(P_\theta(y_i|x, y_{<i})) \quad (1)$$

$P_\theta(y_i|x, y_{<i})$ here is the conditional probability of predicting the target y_i given the instruction prompt x and the preceding tokens $y_{<i}$.

3.2 FIRST: Ranking with a Single Token

The FIRST method operates under the hypothesis – which we validated in §1 – that LLMs can latently approximate the full ranked list during the generation of the first (top-ranked) passage identifier. FIRST simply extracts the output logits of candidate identifier tokens while generating the first identifier y_1 and returns the passage ranking in the order of decreasing logit values. Crucially, this process only involves computing the output logits of a single token during inference.

Since this ranking is based on output logits of individual tokens from the LLM's vocabulary, avoiding tokenizing passage identifiers into multiple tokens is key. Using numeric identifiers would limit

the number of candidates to ≤ 9 as byte pair encoding (Sennrich et al., 2016) tokenizes multiple-digit numbers into more than one token. We, therefore, adopt alphabetic identifiers instead, ranging from A to Z, as LLM rerankers typically consider up to 20 candidate passages in a single window.

Using FIRST directly with current LLM rerankers (Pradeep et al., 2023a,b), while showing promise in the evaluation of Figure 2, is still suboptimal, as these models are finetuned with a language modeling objective. Hence, we propose to leverage a learning-to-rank objective to provide targeted supervision to FIRST rerankers that can better equip them to rank using the first token's output logits. Formally, given m candidate passages $(p_1, p_2, ..., p_m)$, with t_i the identifier token of p_i and s_i the output vocabulary logit of passage identifier t_i during first token generation, let $r_i \in [1, 2, ..., m]$ be the true rank of p_i within the m candidates. We consider as our training objective a weighted version of RankNet (Burges et al., 2005) – a pairwise loss which considers the correctness of relative passage orders to formulate the learning-to-rank objective – as follows:

$$\begin{aligned}\mathcal{L}_{Rank} &= \sum_{i=1}^{m}\sum_{j=1}^{m} \frac{\mathbb{1}_{r_i<r_j}}{i+j}\log(1+\exp(s_i - s_j)) \\ &= \sum_{r_i<r_j} \frac{1}{i+j}\log(1+\exp(s_i - s_j))\end{aligned} \tag{2}$$

Here, the weight $1/(i+j)$ is the inverse mean rank of candidate pair (i, j), which prioritizes getting the ranks of higher-ranked candidates right over those of lower-ranked ones. Since the standard language modeling objective has also been used successfully to train listwise rerankers, we combine it with $\mathcal{L}_{Rank}$ to construct the following joint loss for our training:

$$\mathcal{L}_{Joint} = \mathcal{L}_{LM} + \lambda\mathcal{L}_{Rank} \tag{3}$$

where λ is a hyperparameter that controls the relative importance of the two losses. Note that while $\mathcal{L}_{Rank}$ is applied only to the output logits of the first generated token, $\mathcal{L}_{LM}$ is an aggregate over all tokens in the target ranking sequence. At inference, FIRST uses only the output vocabulary logits of the first generation token to obtain the ranked candidate identifier order.

4 Experiments

We first demonstrate in §4.2 that the proposed ranking loss improves the accuracy of listwise LLM reranking. Next, in §4.3, we measure the improvement in latency of inference from using FIRST. Finally, we show in §4.4 that leveraging listwise LLM rerankers for relevance feedback improves the recall of retrievers.

4.1 Setup

Model: We follow Pradeep et al. (2023b) to use Zephyr$_\beta$ (Tunstall et al., 2023) as our instruction-following LLM for listwise reranking. Zephyr$_\beta$ is a 7B LLM based on Mistral (Jiang et al., 2023) and instruction-tuned on chat datasets (Ding et al., 2023; Cui et al., 2023). We finetune Zephyr$_\beta$ for listwise reranking for three epochs with an effective batch size of 32, a learning rate of 5e-6 using bfloat16 precision, and leverage noisy embeddings (Jain et al., 2023). Training takes approximately 7 hours on four 40GB Nvidia A100 GPUs when used with DeepSpeed (Rasley et al., 2020). We randomly sample 300 queries from MS Marco as our development set, and use $\lambda = 10$ for scaling the weighted RankNet loss

Datasets: We use 40k GPT-4 labeled instances from Pradeep et al. (2023b) for fine-tuning LLM rerankers, which were created using 5k queries from MS MARCO (Nguyen et al., 2016a). Examples contain a variable number (≤ 20) of candidate passages that need to be reranked. For evaluation, we use the BEIR benchmark (Thakur et al., 2021b), which comprises test instances from MS MARCO and out-of-domain evaluation data from several scientific, biomedical, financial, and Wikipedia-based retrieval datasets[3].

Reranking Setup: We use Contriever (Gautier et al., 2022) for retrieving an initial set of candidates. The top 100 retrieved passages are then passed as input to the reranker. The listwise reranking process uses a sliding window strategy as in Sun et al. (2023); Pradeep et al. (2023b), with window size $m = 20$ and step size $s = 10$.

Baselines: We compare performance with a pointwise cross-encoder reranker from Thakur et al. (2021a), as well as RankVicuna (Pradeep et al., 2023a) and RankZephyr (Pradeep et al., 2023b), which are LLM-based listwise rerankers.

[3]We use the same BEIR subset as in Reddy et al. (2023).

Reranker	Training Data	Avg.	Climate FEVER	DBP-edia	FEVER	FiQA	Hotpot QA	MS Marco	NFC-orpus	NQ	Sci-docs	Sci-fact	Trec-COVID
None	MS Marco	45.9	23.7	41.3	75.8	32.9	63.8	40.7	32.8	49.8	16.5	67.7	59.6
Cross-Encoder	MS Marco	50.7	25.5	47.0	**81.9**	35.6	71.8	**47.0**	34.5	57.6	17.0	69.1	71.0
Rank Vicuna	GPT 3.5	50.7	**28.2**	50.0	81.0	35.9	73.5	36.7	33.1	58.6	18.4	70.5	71.3
Rank Zephyr	GPT 3.5+4	53.7	25.6	50.0	80.1	**42.2**	71.6	42.7	**37.7**	65.6	**20.5**	**76.7**	78.4
FIRST	GPT-4	**54.3**	26.7	**50.9**	81.7	**42.2**	**74.2**	44.4	37.4	**66.4**	20.4	74.6	**78.8**

Table 1: Performances of different rerankers (nDCG@10 in %) on BEIR (Thakur et al., 2021b). Top-100 retrieval results from Contriever (Gautier et al., 2022) are passed as input. Reranker: *None* indicates the retriever.

Training Strategy	Inference	Avg.	Climate FEVER	DBP-edia	FEVER	FiQA	Hotpot QA	MS Marco	NFC-orpus	NQ	Sci-docs	Sci-fact	Trec-COVID
LM	Generation	52.3	20.8	48.6	79.1	40.6	71.3	43.5	35.4	65.6	19.7	72.3	77.6
LM + RankNet	Generation	53.4	25.1	50.1	80.3	41.1	71.7	43.4	**37.6**	64.7	20.3	74.1	**79.4**
	FIRST	**54.3**	**26.7**	**50.9**	**81.7**	42.2	74.2	44.4	37.4	66.4	**20.4**	74.6	78.8
- Weighting	FIRST	53.8	23.7	50.1	79.0	**43.2**	**74.9**	**44.6**	36.8	**66.9**	19.7	**75.3**	77.5
- LM	FIRST	51.7	20.3	48.8	74.8	40.6	72.5	43.2	35.9	63.5	19.3	73.5	76.2

Table 2: Table showing the nDCG@10 (in %) on BEIR (Thakur et al., 2021b) for LLM listwise reranking when training with different strategies. *LM* corresponds to the traditional language modeling objective for training.

The cross-encoder was trained using 500k pairwise human-annotated instances from MS MARCO (Nguyen et al., 2016b). RankVicuna was finetuned using the RankGPT data (Sun et al., 2023), which contains GPT-3.5 labeled listwise reranking examples created from 100k MS MARCO queries. RankZephyr employs a two-stage training process that first finetunes with the RankGPT data and then with GPT-4 labeled listwise reranking examples created from 5k MS MARCO queries. We only use the smaller GPT-4 labeled instances due to compute constraints.

Dataset	RankNet	LambdaRank	ListNet
DBPedia	**50.9**	47.3	49.1
FiQA	42.2	43.2	**43.7**
NFCorpus	**37.4**	35.6	36.8
Scifact	74.6	**76.1**	74.4
Trec-COVID	**78.8**	75.0	75.5
Average	**56.7**	55.4	55.9

Table 3: Table showing the nDCG@10 (in %) on a subset of BEIR from incorporating different ranking losses when finetuning the listwise LLM reranker.

4.2 Ranking Performance

Table 1 shows nDCG@10 scores of different rerankers on BEIR (Thakur et al., 2021b), where each reranker was used to rerank the top-100 retrievals of Contriever. We first observe that FIRST outperforms RankZephyr despite being fine-tuned on considerably less data. Note that the cross-encoder achieves a very high score on MS MARCO as it was trained with in-domain human-annotated data, unlike the LLM rerankers.

Next, we report results from ablation studies involving different finetuning strategies in Table 2. The proposed joint loss significantly improves performance over finetuning with just the language modeling objective. We observe that the FIRST approach to inference, in addition to being considerably faster, also produces more accurate results than sequence generation. This suggests that error made early in the reranking with autoregressive sequence generation can propagate, leading to potentially suboptimal rankings. The benefit of adding the proposed inverse mean rank weighting to the existing RankNet loss is also evident. Interestingly, we observe that finetuning using only the weighted RankNet loss performs worse than using only the LM objective, which is perhaps unsurprising given the alignment of the latter with LLM pretraining.

Further, in addition to the weighted RankNet loss

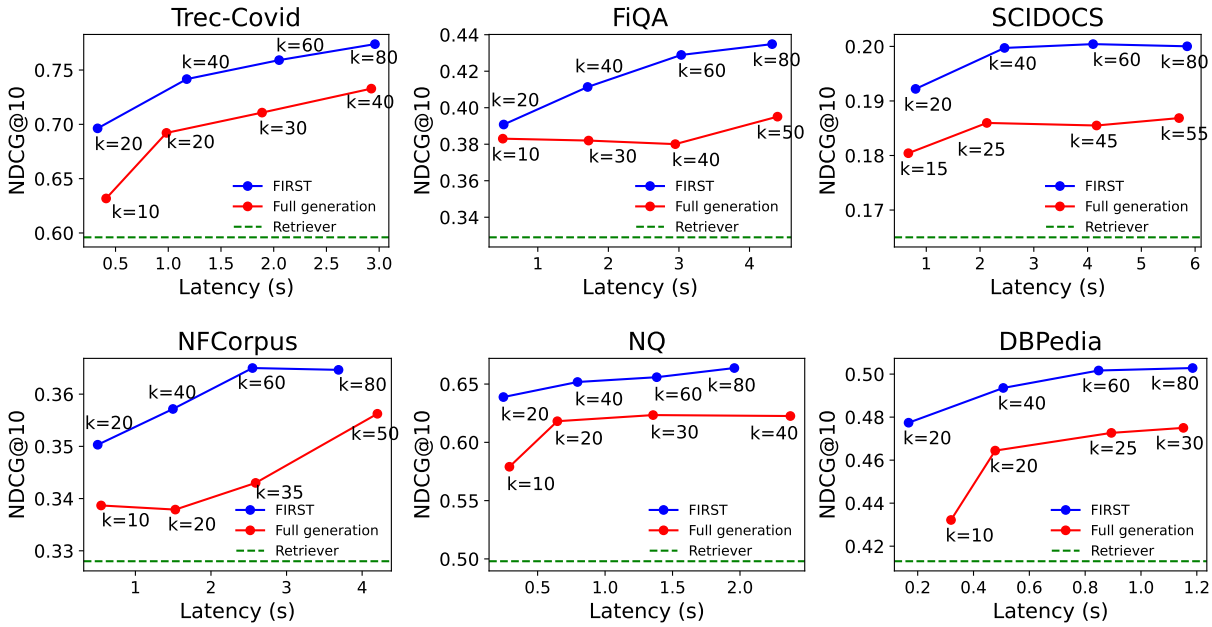


Figure 3: Ranking accuracy (nDCG@10) against the reranker's per query latency in seconds. k refers to the number of passages reranked for the corresponding latency. FIRST considerably outperforms sequence generation when constrained to a latency budget, as it is able to rerank significantly more candidates.

(eq. 2), we experimented with incorporating different ranking losses while finetuning the listwise reranker. Specifically, we considered the LambdaRank and ListNet losses. LambdaRank (Burges et al., 2006) is a pair-wise ranking loss that is similar to RankNet, but uses a weight proportional to the change in the target ranking metric (e.g. NDCG) that would result from swapping the positions of items in the pair. ListNet (Cao et al., 2007) is a listwise loss based on the cross entropy between two parameterized probability distributions of permutations. Table 3 shows the results on a subset of BEIR. We see that our weighted RankNet loss gives a better performance compared to using the LambdaRank and ListNet losses.

4.3 Comparing Latencies

One of the key stated advantages of FIRST is single-token decoding, which can be expected to improve latency considerably. To demonstrate this empirically, we compare the latencies of inference with FIRST and sequence generation[4]. Latency is measured on a 40GB Nvidia A100 GPU and averaged over 200 sampled queries.

We first compare the overall time taken for ranking candidate passages in a single window. Figure 4 plots the latency of FIRST and sequence generation against the window size m. While overall inference time increases for both approaches with more candidate passages in the window, the latency gap between the two grows as m increases. This is understandable, as the output length increases for sequence generation with the number of candidate passage identifiers, but not for FIRST.

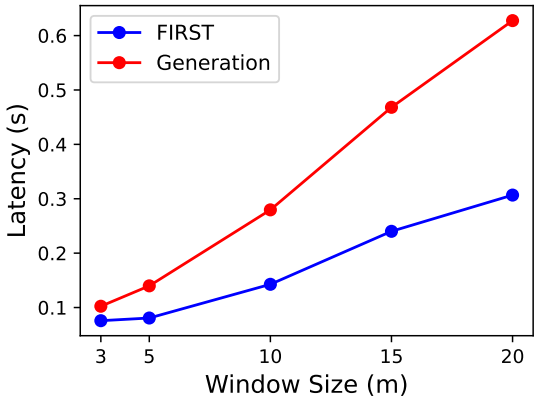


Figure 4: Plot comparing the single window inference latency for FIRST vs. generating the ranked sequence, for different numbers of candidate passages m.

In Figure 3, we further evaluate the reranking accuracy of the two approaches under specific latency requirements. We fix the number of the candidates $k = (20, 40, 60, 80)$ for FIRST and retrieve the corresponding number of candidates with sequence generation under identical latency requirements. Figure 3 shows the plots for six different

[4]For a fair comparison, we omitted the generation time of the identifier indicators ('[' and ']') for sequence generation.

Relevance Feedback	Average R@100	Climate FEVER	DBP-edia	FEVER	FiQA	Hotpot QA	MS Marco	NFC-orpus	NQ	Sci-docs	Sci-fact	Trec-COVID
None	66.8	57.4	54.1	94.9	65.6	77.7	89.1	30.0	92.5	37.8	94.7	40.7
CE (KL Div.)	69.0	**59.5**	57.3	**95.5**	65.6	80.4	**90.5**	31.9	94.2	40.1	95.2	51.5
LLM (RankNet)	71.2	58.8	58.4	95.2	**72.7**	79.8	89.3	34.5	95.6	43.1	**96.1**	59.4
CE + LLM	**72.0**	59.4	**59.8**	**95.5**	71.8	**81.2**	89.7	**35.9**	**96.1**	**44.1**	95.9	**62.2**

Table 4: Table showing recall@100 (in %) on BEIR (Thakur et al., 2021b) using the updated query vector for second-stage retrieval after relevance feedback. Results for *None* correspond to the first-stage retrieval using Contriever. Relevance feedback from cross-encoder (CE) uses the KL divergence loss as in Reddy et al. (2023), while that from listwise LLM reranker uses the weighted RankNet loss (Eq. 2) for optimizing the query vector.

datasets from BEIR, where we observe FIRST to consistently outperform sequence generation while maintaining the same per-query reranking latency. Clearly, FIRST can rerank more candidates k in the same amount of time, which leads to the observed performance gains.

4.4 Relevance Feedback with LLM Rerankers

Here, we demonstrate that the better ranking performance from LLM-based rerankers, when compared to cross-encoders, is advantageous for downstream applications. Specifically, we consider the task of providing relevance feedback (ROCCHIO, 1971) for improving the retrieval recall. Relevance feedback using rerankers at inference involves optimizing the retriever's query representation at test-time using the reranker's output for the retrieval results. Reddy et al. (2023); Sung et al. (2023) update the query representation from dense retrievers, like Contriever (Gautier et al., 2022), by gradient descent based on KL divergence loss between the query vector and cross-encoder reranker scoring distributions over the retrieved passages. Since rerankers are typically more performant than retrievers, the updated query representation, when used for second-stage retrieval, can improve recall upon the previously retrieved results. We refer the reader to Reddy et al. (2023) for more details.

While cross-encoder rerankers provide floating-point scores that can be used as distillation supervision, listwise rerankers output an ordered sequence of the candidates. Hence, the typically used KL divergence loss cannot be applied for relevance feedback in this setting. In this regard, we investigate how listwise rerankers can be leveraged for relevance feedback, and whether they can provide bigger improvements for second-stage retrieval recall compared to cross-encoders. We experiment with using the weighted RankNet loss (in eq. 2) to use the ranked ordering from listwise rerankers as distillation supervision for relevance feedback.

For our experiments, we follow the same setup as Reddy et al. (2023) with Contriever for initial retrieval and evaluation on BEIR (Thakur et al., 2021b). Distillation using the cross-encoder with KL divergence loss has a learning rate of 0.005 and 100 gradient updates, while that using the LLM reranker with the weighted RankNet loss has a learning rate of 0.001 and 20 gradient updates. Table 4 shows recall@100 numbers from second-stage retrieval after different relevance feedback strategies. We observe that relevance feedback from the LLM reranker significantly improves recall compared to the cross-encoder reranker. We attribute this to the superior ranking performance of LLM rerankers (as seen in Table 1), thereby providing higher quality relevance feedback. Moreover, we see that using the LLM reranker feedback in addition to that from the cross-encoder (CE+LLM) leads to further gains. This improvement could be explained as the diversity of feedback signals from the two rerankers, i.e. floating-point scores for cross-encoder vs ranking sequence for listwise reranker, providing a more comprehensive distillation supervision and demonstrating the huge potential of listwise rerankers for relevance feedback.

5 Conclusion

In this work, we introduce FIRST, a novel strategy for listwise LLM reranking. FIRST leverages the output logits of the first generated identifier to obtain a ranking for the candidates, as opposed to the typical approach of generating the entire ranked ordering sequence of candidate passage identifiers. We demonstrated that our single-token decoding approach reranks a considerably larger number of

candidates compared to inference with ordered sequence generation in the same time, leading to larger gains when reranking under a latency constraint. FIRST also demonstrates ranking performance benefits from incorporating a learning-to-rank loss during training, allowing for prioritizing more important ranks. By addressing both the training and inference inefficiencies of existing LLM listwise reranking approaches, FIRST represents a significant step forward in the development of advanced re-ranking techniques using LLMs.

Limitations

While FIRST benefits from leveraging GPT-4 labeled data for training, we have not experimented with using human-annotated pairwise examples in supervised datasets such as MS Marco to further improve performance. Moreover, our experiments here are on English data on account of the underlying LLM being predominantly monolingual. An interesting extension would be to finetune a multilingual LLM for listwise reranking to demonstrate the benefit of our approach in other languages. Further, we use alphabets as passage identifiers since the window size for listwise reranking is typically $\leq$20. However, we expect finetuning using other vocabulary tokens as identifiers should enable leveraging a larger set of candidate identifiers in case the window size needs to be further increased.

Acknowledgements

We acknowledge Ron Arel, Rishub Tamirisa and Andy Zhou from Lapis Labs for helping with access to NCSA Delta compute. We would also like to thank members of the BlenderNLP group for valuable comments and feedback. We are grateful to Ronak Pradeep for releasing the training data and code for RankZephyr. This research is based upon work supported DARPA ITM Program No. FA8650-23-C-7316 and the Agriculture and Food Research Initiative (AFRI) grant no. 2020-67021-32799/project accession no.1024178 from the USDA National Institute of Food and Agriculture. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

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*, ICML '05, pages 89–96, New York, NY, USA. ACM.

Christopher Burges, Robert Ragno, and Quoc Le. 2006. Learning to rank with nonsmooth cost functions. In *Advances in Neural Information Processing Systems*, volume 19. MIT Press.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, ICML '07, pages 129–136, New York, NY, USA. ACM.

Sukmin Cho, Soyeong Jeong, Jeong yeon Seo, and Jong C Park. 2023. Discrete prompt optimization via constrained generation for zero-shot re-ranker. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 960–971.

Koby Crammer and Yoram Singer. 2001. Pranking with ranking. In *Advances in Neural Information Processing Systems*, volume 14. MIT Press.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Wei Zhu, Yuan Ni, Guotong Xie, Zhiyuan Liu, and Maosong Sun. 2023. Ultrafeedback: Boosting language models with high-quality feedback. *Preprint*, arXiv:2310.01377.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. 2023. Enhancing chat language models by scaling high-quality instructional conversations. *arXiv preprint arXiv:2305.14233*.

Izacard Gautier, Caron Mathilde, Hosseini Lucas, Riedel Sebastian, Bojanowski Piotr, Joulin Armand, and Grave Edouard. 2022. Unsupervised dense information retrieval with contrastive learning. *Transactions on Machine Learning Research*.

Neel Jain, Ping-yeh Chiang, Yuxin Wen, John Kirchenbauer, Hong-Min Chu, Gowthami Somepalli, Brian R Bartoldson, Bhavya Kailkhura, Avi Schwarzschild, Aniruddha Saha, et al. 2023. Neftune: Noisy embeddings improve instruction finetuning. In *The Twelfth International Conference on Learning Representations*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego

de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

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

Ping Li, Qiang Wu, and Christopher Burges. 2007. Mcrank: Learning to rank using multiple classification and gradient boosting. In *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc.

Tie-Yan Liu et al. 2009. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3):225–331.

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

Chuan Meng, Negar Arabzadeh, Arian Askari, Mohammad Aliannejadi, and Maarten de Rijke. 2024. Ranked list truncation for large language model-based re-ranking. *CoRR*, abs/2404.18185.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016a. Ms marco: A human generated machine reading comprehension dataset. In *CoCo@ NIPs*.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016b. Ms marco: A human generated machine reading comprehension dataset. *CoRR*, abs/1611.09268.

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

Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document ranking with a pre-trained sequence-to-sequence model. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 708–718.

Andrew Parry, Sean MacAvaney, and Debasis Ganguly. 2024. Top-down partitioning for efficient list-wise ranking. *Preprint*, arXiv:2405.14589.

Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023a. Rankvicuna: Zero-shot listwise document reranking with open-source large language models. *arXiv preprint arXiv:2309.15088*.

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

Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, Xuanhui Wang, et al. 2023. Large language models are effective text rankers with pairwise ranking prompting. *arXiv preprint arXiv:2306.17563*.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 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*, pages 3505–3506.

Revanth Gangi Reddy, Pradeep Dasigi, Md Arafat Sultan, Arman Cohan, Avirup Sil, Heng Ji, and Hannaneh Hajishirzi. 2023. Inference-time re-ranker relevance feedback for neural information retrieval. *arXiv preprint arXiv:2305.11744*.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

J ROCCHIO. 1971. Relevance feedback information retrieval. *The Smart Retrieval System-Experiments in Automatic Document Processing*, pages 313–323.

Devendra Sachan, Mike Lewis, Mandar Joshi, Armen Aghajanyan, Wen-tau Yih, Joelle Pineau, and Luke Zettlemoyer. 2022. Improving passage retrieval with zero-shot question generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3781–3797.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 2023. Is chatgpt good at search? investigating large language models as re-ranking agents. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14918–14937.

Mujeen Sung, Jungsoo Park, Jaewoo Kang, Danqi Chen, and Jinhyuk Lee. 2023. Optimizing test-time query representations for dense retrieval. In *The 61st Annual Meeting Of The Association For Computational Linguistics*.

Raphael Tang, Xinyu Zhang, Xueguang Ma, Jimmy Lin, and Ferhan Ture. 2023. Found in the middle: Permutation self-consistency improves listwise ranking in large language models. *arXiv preprint arXiv:2310.07712*.

Michael Taylor, John Guiver, Stephen Robertson, and Tom Minka. 2008. Softrank: optimizing non-smooth

rank metrics. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, WSDM '08, page 77–86, New York, NY, USA. Association for Computing Machinery.

Nandan Thakur, Nils Reimers, Johannes Daxenberger, and Iryna Gurevych. 2021a. Augmented SBERT: Data augmentation method for improving bi-encoders for pairwise sentence scoring tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 296–310, Online. Association for Computational Linguistics.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021b. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models. *arXiv preprint arXiv:2104.08663*.

Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourrier, Nathan Habib, Nathan Sarrazin, Omar Sanseviero, Alexander M. Rush, and Thomas Wolf. 2023. Zephyr: Direct distillation of lm alignment. *Preprint*, arXiv:2310.16944.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024. Large search model: Redefining search stack in the era of llms. In *ACM SIGIR Forum*, volume 57, pages 1–16. ACM New York, NY, USA.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent abilities of large language models. *Transactions on Machine Learning Research*.

Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. 2008. Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 1192–1199, New York, NY, USA. Association for Computing Machinery.

Ruicheng Xian, Honglei Zhuang, Zhen Qin, Hamed Zamani, Jing Lu, Ji Ma, Kai Hui, Han Zhao, Xuanhui Wang, and Michael Bendersky. 2023. Learning list-level domain-invariant representations for ranking. *Advances in Neural Information Processing Systems*, 36.

Yutao Zhu, Huaying Yuan, Shuting Wang, Jiongnan Liu, Wenhan Liu, Chenlong Deng, Zhicheng Dou, and Ji-Rong Wen. 2023. Large language models for information retrieval: A survey. *arXiv preprint arXiv:2308.07107*.

Honglei Zhuang, Zhen Qin, Kai Hui, Junru Wu, Le Yan, Xuanhui Wang, and Michael Berdersky. 2023a. Beyond yes and no: Improving zero-shot llm rankers via scoring fine-grained relevance labels. *arXiv preprint arXiv:2310.14122*.

Honglei Zhuang, Zhen Qin, Rolf Jagerman, Kai Hui, Ji Ma, Jing Lu, Jianmo Ni, Xuanhui Wang, and Michael Bendersky. 2023b. Rankt5: Fine-tuning t5 for text ranking with ranking losses. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2308–2313.

Shengyao Zhuang, Bing Liu, Bevan Koopman, and Guido Zuccon. 2023c. Open-source large language models are strong zero-shot query likelihood models for document ranking. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 8807–8817.