Accelerating Listwise Reranking: Reproducing and Enhancing FIRST

Zijian Chen
s42chen@uwaterloo.ca
University of Waterloo
Waterloo, Ontario, Canada

Ronak Pradeep
rpradeep@uwaterloo.ca
University of Waterloo
Waterloo, Ontario, Canada

Jimmy Lin
jimmylin@uwaterloo.ca
University of Waterloo
Waterloo, Ontario, Canada

Abstract

Large language models (LLMs) have emerged as powerful listwise rerankers but remain prohibitively slow for many real-world applications. What's more, training on the language modeling (LM) objective is not intrinsically aligned with reranking tasks. To address these challenges, FIRST, a novel approach for listwise reranking, integrates a learning-to-rank objective and leverages only the logits of the first generated token for reranking, significantly reducing computational overhead while preserving effectiveness. We systematically evaluate the capabilities and limitations of FIRST. By extending its evaluation to TREC Deep Learning collections (DL19–23), we show that FIRST achieves robust out-of-domain effectiveness. Through training FIRST on a variety of backbone models, we demonstrate its generalizability across different model architectures, and achieve effectiveness surpassing the original implementation. Further analysis of the interaction between FIRST and various first-stage retrievers reveals diminishing returns akin to traditional LLM rerankers. A comprehensive latency study confirms that FIRST consistently delivers a 40% efficiency gain over traditional rerankers without sacrificing effectiveness. Notably, while LM training implicitly improves zero-shot single-token reranking, our experiments also highlight potential conflicts between LM pre-training and subsequent fine-tuning on the FIRST objective. These findings pave the way for more efficient and effective listwise reranking in future applications. Our code is available at: https://rankllm.ai.

CCS Concepts

• **Information systems** → **Language models**; **Retrieval efficiency**; **Retrieval effectiveness**.

Keywords

Information Retrieval; Listwise Reranking; Large Language Models

ACM Reference Format:
Zijian Chen, Ronak Pradeep, and Jimmy Lin. 2025. Accelerating Listwise Reranking: Reproducing and Enhancing FIRST. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '25), July 13–18, 2025, Padua, Italy*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3726302.3730287

SIGIR '25, Padua, Italy

ACM ISBN 979-8-4007-1592-1/2025/07
https://doi.org/10.1145/3726302.3730287

1 Introduction

Large language models (LLMs) have emerged as powerful tools for information retrieval, including the task of document reranking [26]. Recent studies have demonstrated that LLMs, used as zero-shot listwise rerankers, can surpass traditional supervised approaches without requiring extensive relevance judgments [10, 13, 21].

However, associated with LLM rerankers is their high inference latency, presenting challenges for practical deployment. Traditional listwise reranking approaches frame the reranking problem as a generation task, requiring LLMs to produce complete permutations of the document identifiers as the output ranking, trained using language modeling loss against the correct permutation. The combination of auto-regressive generation in transformers and the substantial size of LLMs leads to concerning latency issues in practical applications [26].

To address these efficiency challenges, Zhuang et al. [28] proposed examining the relative magnitudes of first-token logits for reranking, eliminating the need to generate entire identifier permutations. Following up, recent work by Reddy et al. [19] introduced FIRST (**F**aster **I**mproved Listwise **R**eranking with **S**ingle **T**oken Decoding), which not only leverages single-token reranking, but also combines it with a new training objective. The authors claimed that the language modeling objective is fundamentally suboptimal for reranking as it uniformly penalizes incorrect ranking across all positions, failing to emphasize the importance of correctly ranking the most relevant documents. By incorporating a learning-to-rank loss alongside their single-token strategy, FIRST promised more efficient reranking without compromising effectiveness.

While results from FIRST were promising, several questions remain about its broader applicability and practical impact. In this work, we present a comprehensive reproduction and analysis of the method, guided by the following research questions:

RQ1 How does FIRST generalize to different LLM backbones beyond the original Zephyr$_\beta$ implementation?

RQ2 How does FIRST generalize to other out-of-domain datasets?

RQ3 To what extent can LLMs trained solely on language modeling perform FIRST-style reranking in a zero-shot setting?

RQ4 How does FIRST perform when paired with different first-stage retrievers?

RQ5 What are the concrete effectiveness–efficiency tradeoffs between FIRST and traditional full-generation?

RQ6 What are the effectiveness–efficiency tradeoffs between different window and step sizes when using FIRST?

Through investigating these questions, we validated the robustness of FIRST across a diverse range of backbone models and datasets, achieving 40% latency improvements while maintaining effectiveness. Moreover, our investigation yielded several insights about the relationship between language modeling and the effectiveness of FIRST: while language modeling training implicitly improves zero-shot single-token reranking capabilities, we discovered that language modeling pre-training may paradoxically hinder subsequent FIRST fine-tuning. These findings provided important implications for model training strategies and contributed to our understanding of efficient reranking approaches.

2 Background and Related Work

Multi-Stage Ranking. Given a corpus of documents denoted $C = \{d_1, d_2, \cdots, d_n\}$ and a query q, *retrieving* refers to the task of finding an ordered list $\mathcal{R}$ of k most relevant documents from C, in descending relevance with respect to q, where $k << |C|$. *Reranking* refers to the downstream task that reorders $\mathcal{R}$, if necessary, to a more accurate ranking. This retrieve–rerank procedure common in modern systems is referred to as *multi-stage ranking*, where the retrieve step often uses a more computationally efficient approach, followed by a more accurate but expensive rerank step [14]. The retrieving system is referred to as a *first-stage retriever* in this setting, while the reranking system is referred to as a *reranker*.

Reranking with Large Language Models. Recent work has demonstrated that LLMs can serve as effective rerankers [16, 18]. Such approaches can be categorized into pointwise, pairwise, or listwise.

Earlier work predominantly utilized the *pointwise* approach, where the LLM assesses each query–document pair independently, computing a likelihood or binary relevance judgment in isolation [27] of the other documents. On the other hand, *pairwise* approaches leverage the LLM to compare the relevance of two documents at a time, given the same query [18].

More recently, RankGPT [21] experimented with a *listwise* approach, where the LLM is prompted with a query and a list of documents, generating a complete permutation of the documents based on their relevance to the query. RankZephyr by Pradeep et al. [16] continued this theme by instruction-tuning Zephyr$_\beta$ [24] to perform such listwise document reranking. Concurrently, Rank-without-GPT [25] explored instruction tuning leveraging non-GPT teacher models. Newer work [22] has also explored more efficient listwise reranking through smaller encoder–decoder models.

In this study, we focus on utilizing LLMs as listwise rerankers.

Listwise Reranking with FIRST. Although listwise reranking approaches have demonstrated strong effectiveness, they are computationally intensive, as they typically rely on sequence generation to produce an entire permutation of document identifiers. FIRST by Reddy et al. [19] addresses these inefficiencies by utilizing only the logits from the first token in the output sequence to determine the rank order of candidate documents, rather than generating a complete ranked sequence.

Further, FIRST incorporates a learning-to-rank objective during training, prioritizing ranking accuracy for top candidates over less relevant ones, rather than focusing solely on the language modeling objective. This modification by Reddy et al. promised to give more effective supervision during training.

Together, FIRST offers a more efficient reranking model that achieves comparable or superior ranking quality with substantially reduced computational demands. In this work, we refer to LLM rerankers that employ full-sequence generation like RankZephyr [16] as "traditional LLM rerankers", while we designate models utilizing the single-token approach as "FIRST rerankers".

3 Methods

We begin by summarizing the methodology introduced by Reddy et al. [19], while also formalizing the problem and notation.

Listwise Reranking Using Sliding Windows. Recall the reranking problem: given a candidate document list $\mathcal{R} = \{d_1, d_2, \ldots, d_n\}$ for a query q, reorder $\mathcal{R}$ based on the document relevance to q. Due to context window constraints in LLMs, this reordering typically cannot be accomplished in a single step. Following Sun et al. [21], we employ a sliding window approach with window size m and step size s. The window processes m documents at a time, moving from the end of the list toward the front with a stride of s documents. At each step, the LLM is prompted to reorder the documents within the current window according to their relevance to q.

FIRST Objective. Conventional listwise reranking approaches require LLMs to generate a complete permutation of the candidate document identifiers and are trained using a language modeling objective against the correct permutation. In contrast, FIRST derives the rank ordering solely from the relative magnitude of the output logits of the first generated identifier token, eliminating the need for full-sequence generation. To formally state the objective of FIRST:

- t_i denotes the identifier token for document d_i
- p_i represents the logits for generating t_i as the first (most relevant) identifier token
- $r_i \in \{1, \ldots, m\}$ indicates the true rank of document d_i among m candidates

FIRST incorporates a weighted pairwise learning-to-rank loss defined as:

$$\mathcal{L}_{Rank} = \sum_{r_i < r_j} \frac{1}{i+j} \log(1 + \exp(p_i - p_j))$$

where the weight term $\frac{1}{i+j}$ prioritizes accurate ranking of higher-ranked documents, aiming to address the issue of uniformly penalizing incorrect ranking across all positions in language modeling.

The final training objective combines this ranking loss with the traditional language modeling loss $\mathcal{L}_{LM}$: $\mathcal{L}_{Joint} = \mathcal{L}_{LM} + \lambda \mathcal{L}_{Rank}$ where λ is a hyperparameter set to 10 in the original work. We will refer to this as the "FIRST objective".

4 Reproducing FIRST

First, we reproduced the results from Reddy et al. [19] using the procedure detailed in their work.

4.1 Setup

Model and Training. We initialized from Zephyr$_\beta$ [24], a 7B LLM instruction-tuned from Mistral-7B-Instruct-v0.1 [9] on chat datasets,

Dataset	(1) **FIRST-Reddy**	(2) **FirstZephyr$_\beta$**	(3) **FirstRankZephyr**	(4) **FirstMistral**	(5) **FirstLLaMA**
Climate-FEVER	**0.2672**	0.2314	0.2519	0.2417	0.2434
DBPedia-Entity	**0.5092**^{5}	0.4908^{5}	0.4761^{5}	0.5033^{5}	0.4387
FEVER	0.8164	0.8215	0.7927	**0.8413**^{3}	0.8301^{3}
FiQA	0.4223	0.4509^{5}	0.4263	**0.4778**^{1,3,5}	0.4162
HotpotQA	0.7424^{5}	0.7620^{5}	0.7506^{5}	**0.7705**^{5}	0.7017
MSMARCO	0.4425	0.4383	0.4275	**0.4512**	0.4316
NFCorpus	0.3725	0.3729	0.3555	**0.3816**^{5}	0.3481
NQ	0.6638^{5}	0.6928^{3,5}	0.6535	**0.6985**^{3,5}	0.6290
SCIDOCS	0.2047	0.2064	0.1874	**0.2110**	0.1848
SciFact	0.7459	0.7680	0.7495	**0.7769**	0.7489
TREC-COVID	**0.7913**^{3,5}	0.7683^{5}	0.7552^{5}	0.7666^{5}	0.6565
Average	0.5435	0.5458	0.5297	**0.5564**	0.5117

Table 1: Comparison of nDCG@10 across the datasets selected by Reddy et al. [19] from BEIR and MS MARCO, on models trained on different backbones with the FIRST objective. Contriever was used as the first-stage retriever. Models are numbered from (1) to (5). The superscripts indicate statistically significant improvements (paired Student's t-test with $p \leq 0.01$ with Bonferroni correction). e.g., ^{1} indicates that the entry is significantly higher than the entry in column 1 of that row.

and we fine-tuned using the joint objective $\mathcal{L}_{Joint}$ with $\lambda = 10$. Training lasted for 3 epochs, using:

- Effective batch size: 32
- Learning rate: 5e-6
- Noisy embeddings [8]
- Sliding window size (m): 20
- Step size (s): 10

All model training was performed on 4 NVIDIA RTX A6000's. The trained checkpoint is referred to as FirstZephyr$_\beta$.

Training Data. We utilized the same dataset as Reddy et al. [19]: 40K GPT-4 labeled rerank instances from Pradeep et al. [16]. Note that to ensure single-token identification, Reddy et al. had converted the original dataset to use alphabetical identifiers rather than numerical identifiers.[1]

Retriever. Following the original paper, we employed Contriever [11] as our first-stage retriever, selecting the top 100 documents for LLM reranking.

Baseline Evaluation. We evaluated on the same data used in the original FIRST study, which comprises of several subsets of BEIR [23] and MS MARCO [1].

4.2 Results and Discussion

Table 1 compares the reranking quality across various models trained with the FIRST objective, evaluated on the benchmark datasets used in the original study. We compared two key implementations: FIRST-Reddy, the official checkpoint[2] released by Reddy et al. running on our machines, and FirstZephyr$_\beta$, our reproduction trained according to Reddy et al.'s procedures detailed above. Our reproduced model achieved comparable average effectiveness to the original checkpoint, while demonstrating modest improvements across most datasets. This validates both our implementation and the reproducibility of the FIRST approach.

Despite these validations, our reproduction also revealed a subtle but notable tokenization issue. Since FIRST reranks documents based on the first generated token—a capitalized letter identifier such as "A" or "B"—it assumes that these tokens remain stable during inference. However, we observed that these identifiers occasionally appear in slightly different forms, most commonly another token of the same letter but preceded by a whitespace. Consequently, the same document can be assigned multiple rankings within the top logits, appearing both as "A" and " A" (notice the whitespace), effectively duplicating an identifier. This issue was not anecdotal. When running FIRST-Reddy on the TREC-COVID dataset, 5 out of 50 queries contained duplicate identifiers in their top 20 ranked logits; recall that 20 is also the window size used. While we addressed this problem through post-processing by filtering out repeated identifiers, this phenomenon highlights a fundamental limitation: alphabetical identifiers, despite being single tokens, may not be the most reliable choice for reranking tasks.

5 Enhancing FIRST

5.1 Varying the Model Backbone

RQ1 How does FIRST generalize to different LLM backbones beyond the original Zephyr$_\beta$ implementation?

We fine-tuned various other prominent LLMs of comparable size, besides Zephyr$_\beta$, while keeping other settings constant:

- FirstMistral: fine-tuned from Mistral-7B-Instruct-v0.3[3]
- FirstLLaMA: fine-tuned from LLaMA-3.1-8B-Instruct[4]
- FirstRankZephyr: fine-tuned from RankZephyr [16]

[1] The alphabetical version is available at: https://huggingface.co/datasets/rryisthebest/rank_zephyr_training_data_alpha
[2] https://huggingface.co/rryisthebest/First_Model
[3] https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3
[4] https://ai.meta.com/blog/meta-llama-3-1

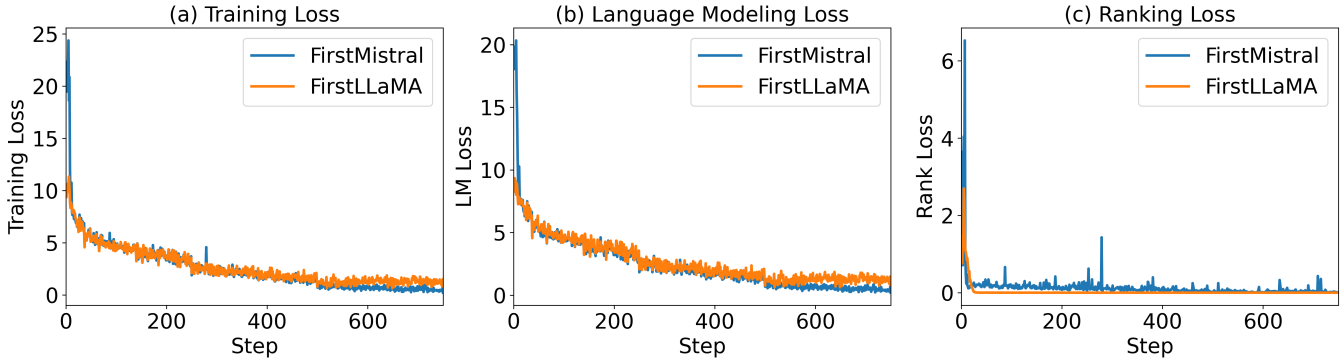

Figure 1: Comparison of training, language modeling, and ranking losses for FirstMistral and FirstLLaMA during training.

Note that FirstRankZephyr was fine-tuned from RankZephyr, which in turn has been previously trained as a listwise reranker on $\mathcal{L}_{LM}$.

As demonstrated in Table 1, FirstMistral achieved the highest nDCG@10 scores on 8 out of 11 of the datasets selected by Reddy et al. [19], with significant improvements over most other models on FiQA (determined via paired Student's t-test with $p \leq 0.01$, using Bonferroni correction). This superior effectiveness can likely be attributed to its initialization from a more recent version of Mistral 7B (v0.3), while FIRST-Reddy, FirstZephyr$_\beta$, and FirstRankZephyr were fine-tuned from Zephyr$_\beta$, which in turn was fine-tuned from an earlier Mistral 7B version (v0.1).

To better understand FirstLLaMA's significantly lower effectiveness on most datasets (in contrast to FirstMistral), we tracked and compared its training process with FirstMistral, as illustrated in Figure 1. The analysis monitored three metrics: the combined training loss ($\mathcal{L}_{Joint}$), its language modeling component ($\mathcal{L}_{LM}$), and its ranking component ($\lambda\mathcal{L}_{Rank}$). While FirstLLaMA exhibited faster convergence in ranking loss, FirstMistral demonstrated more efficient convergence in language modeling loss, which dominated the overall training loss. This phenomenon could suggest that the joint objective hyperparameters (e.g., the weighting factor λ) and overall loss design may interact in an architecture-specific manner, that the FIRST settings chosen for Zephyr$_\beta$ (and the Mistral-7B family) might not be optimal for LLaMA-3.1-8B-Instruct. Consequently, although FIRST appears to generalize across the backbones selected above, practitioners should be prepared to re-tune objective weights and learning schedules when adapting to new backbones to fully unlock each model's potential.

In addition, effectiveness across the eleven BEIR datasets is far from uniform: FirstMistral achieves its largest gains on FiQA, significantly outperforming most other FIRST models, whereas on TREC-COVID the margin narrows considerably. Therefore, when deploying FIRST, it is essential to evaluate on benchmarks that reflect the target setting's characteristics (e.g., query complexity, corpus size, topic domain), ensuring that the FIRST model used aligns with application-specific requirements.

5.2 Evaluating on TREC Deep Learning

RQ2 How does FIRST generalize to other out-of-domain datasets?

We extended the evaluation beyond the datasets selected by Reddy et al. to include five TREC Deep Learning Track test collections. The TREC 2019 and 2020 Deep Learning Tracks (DL19, DL20) [2, 5] draw from the MS MARCO v1 passage corpus, which contains approximately 8.8 million passages. In contrast, the TREC 2021, 2022, and 2023 Deep Learning Tracks (DL21, DL22, DL23) [3, 4, 6] are based on the substantially larger MS MARCO v2 passage corpus, containing around 138 million passages.

Table 2 presents a comprehensive evaluation of model effectiveness across TREC DL19–23. The results reinforced our previous findings from Sections 4 and 5.1: FirstMistral maintained its superior scores across these new test collections, while FIRST-Reddy and FirstZephyr$_\beta$ demonstrated comparable effectiveness, and FirstLLaMA continued to show relatively lower effectiveness.

Notably, with the exception of FirstLLaMA, models trained with the FIRST objective achieved effectiveness comparable to RankZephyr$_g$, a model trained on $\mathcal{L}_{LM}$, with FirstMistral even surpassing RankZephyr$_g$'s average effectiveness. Since TREC DL19–23 were not included in the original study, these results provide additional validation for Reddy et al.'s central claim that the FIRST approach, their attempt to efficient reranking, does not compromise reranking effectiveness.

5.3 Evaluating Zero-Shot FIRST

RQ3 To what extent can LLMs trained solely on language modeling perform FIRST-style reranking in a zero-shot setting?

A key insight from Reddy et al. was that RankZephyr, despite being trained solely with a language modeling objective ($\mathcal{L}_{LM}$), exhibited zero-shot ability to rank documents using the logits of the first identifier only; its capability to rerank using the FIRST strategy was notably stronger than in the pre-trained model, suggesting that fine-tuning on $\mathcal{L}_{LM}$ implicitly improves single-token ranking. To investigate this phenomenon further, we evaluated two open-source listwise rerankers trained exclusively on $\mathcal{L}_{LM}$—RankZephyr and RankVicuna [15, 16]—on the TREC Deep Learning Track test collections (TREC DL19–23), using only the first-token logits at inference time.

As evident in row RankZephyr$_l$ of Table 2, RankZephyr achieved effectiveness on par with dedicated FIRST models when using only first-token logits for reranking, even achieving significantly higher

Model	DL19	DL20	DL21	DL22	DL23	Average
(1) FIRST-Reddy	0.7476	$0.7986^{5,8,9}$	$\mathbf{0.7709}^{3,5,6,8,9}$	$0.6944^{5,8,9}$	$0.5630^{6,8,9}$	0.7149
(2) FirstZephyr$_\beta$	0.7576	0.7550	0.7439	$0.6767^{5,8,9}$	$0.5533^{3,6,8,9}$	0.6973
(3) FirstRankZephyr	0.7315	0.7363	0.7145	$0.6442^{8,9}$	0.5106	0.6674
(4) FirstMistral	0.7678	$0.7901^{5,8}$	$0.7694^{3,5,6,8,9}$	$\mathbf{0.7030}^{2,3,5,8,9}$	$\mathbf{0.5743}^{3,5,6,8,9}$	**0.7209**
(5) FirstLLaMA	0.7363	0.7263	0.6941	0.6042^{8}	0.4987^{8}	0.6519
(6) RankZephyr$_l$	0.7369	0.7370^{8}	0.7103^{5}	0.6165	0.4874^{8}	0.6576
(7) RankZephyr$_g$	$\mathbf{0.7760}^{9}$	$\mathbf{0.8140}^{5,8,9}$	$0.7605^{5,8,9}$	0.6669^{9}	0.5658^{8}	0.7166
(8) RankVicuna$_l$	0.7302	0.7103	0.6809	0.5794	0.4689	0.6339
(9) RankVicuna$_g$	0.6894	0.7081	0.6947	0.5521	0.4844	0.6257

Table 2: Comparison of nDCG@10 across TREC DL19–23 on models trained on the FIRST objective, as well as RankZephyr and RankVicuna that were trained on $\mathcal{L}_{LM}$. For RankZephyr, RankZephyr$_l$ denotes the model reranking using the logits of the first identifier only, and RankZephyr$_g$ denotes the model reranking by generating the full permutation of document identifiers. RankVicuna$_l$ and RankVicuna$_g$ are defined similarly for RankVicuna. Models are numbered from (1) to (9). The superscripts indicate statistically significant improvements (paired Student's t-test with $p \leq 0.01$ with Bonferroni correction). e.g., ^{1} indicates that the entry is significantly higher than the entry in row 1 of that dataset.

effectiveness than FirstLLaMA on DL21, despite not being explicitly trained for this objective. This observation validates Reddy et al.'s hypothesis that training on $\mathcal{L}_{LM}$ implicitly improves the model's ability to perform single-token rerank. In fact, this is even more pronounced in the case of RankVicuna, where RankVicuna$_l$ outperformed RankVicuna$_g$; that is, a model trained on $\mathcal{L}_{LM}$ was even more effective on average when reranking using a single-token.

However, the relationship between $\mathcal{L}_{LM}$ and FIRST is slightly more nuanced. We compared two models:

1. FirstRankZephyr: sequentially fine-tuned first on $\mathcal{L}_{LM}$ (starting from Zephyr$_\beta$ to create RankZephyr) and then on the FIRST objective
2. FirstZephyr$_\beta$: fine-tuned directly from Zephyr$_\beta$ using the FIRST objective

The results in Tables 1 and 2 show that FirstZephyr$_\beta$ consistently outperformed FirstRankZephyr across most datasets, with significant improvements on the NQ dataset. That is, while $\mathcal{L}_{LM}$ training improves zero-shot FIRST effectiveness, it may actually hinder subsequent fine-tuning with the FIRST objective. This result challenges the intuitive assumption that language model pre-training necessarily benefits downstream FIRST training.

5.4 Varying First-Stage Retrievers

RQ4 How does FIRST perform when paired with different first-stage retrievers?

We conducted experiments with three first-stage retrievers, in combination with FirstMistral, our most effective FIRST model, on TREC DL19 and DL20. These retrievers represent diverse approaches to information retrieval: a classical lexical method BM25 [20], a sparse neural retriever SPLADE++ EnsembleDistil [7], and a dense neural retriever RepLLaMA [12]. All three retrievers were also used as baselines in the RankZephyr study [16].

Method	DL19	DL20
BM25	0.5058	0.4796
BM25 → FirstMistral	0.7277	0.6971
Improvement	+43.9%	+45.4%
SPLADE++ EnsembleDistil	0.7308	0.7197
SPLADE++ EnsembleDistil → FirstMistral	0.7678	0.7901
Improvement	+5.1%	+9.8%
RepLLaMA	0.7384	0.7195
RepLLaMA → FirstMistral	0.7587	0.7682
Improvement	+2.8%	+6.8%

Table 3: Comparison of nDCG@10 for FirstMistral across different first-stage retrievers, evaluated on DL19 and DL20.

The results in Table 3 revealed two key patterns. First, stronger initial retrieval consistently results in improved post-reranking effectiveness with FirstMistral. Second, we observed patterns of diminishing returns from better first-stage retrievers, as reflected in the decreasing percentage improvements. That is, while starting with better retrieval improved post-reranking effectiveness, the marginal improvement contributed by FIRST tapers off once the initial retrieval is already strong.

These findings align with patterns reported in RankZephyr [16], suggesting that the relationship between first-stage retriever quality and final ranking effectiveness remains consistent, regardless of whether the reranker employs traditional listwise reranking with full generation or FIRST's single-token approach.

Dataset	FirstMistral	RankMistral
Climate-FEVER	0.2417	0.2411
DBPedia-Entity	0.5033	0.5088
FEVER	0.8413	0.8223
FiQA	0.4778	0.4537
HotpotQA	0.7705	0.7349
MSMARCO	0.4512	0.4351
NFCorpus	0.3816	0.3828
NQ	0.6985	0.6835
SCIDOCS	0.2110	0.2108
SciFact	0.7769	0.7743
TREC-COVID	0.7666	0.7840
DL19	0.7678	0.7772
DL20	0.7901	0.7949
DL21	0.7694	0.7603
DL22	0.7030	0.6980
DL23	0.5743	0.5537
Average	0.6078	0.6010

Table 4: Comparison of nDCG@10 for FirstMistral and RankMistral across the datasets selected by Reddy et al., as well as TREC DL19–23. We did not observe any $p \leq 0.01$ with paired Student's t-test on the datasets.

5.5 Comparing Effectiveness–Efficiency Tradeoffs

RQ5 What are the concrete effectiveness–efficiency tradeoffs between FIRST and traditional full-generation?

Effectiveness. To rigorously assess the effectiveness of the FIRST objective against the traditional language modeling objective $\mathcal{L}_{LM}$, we conducted experiments on two sets of pre-trained models. In each set, we compared models initialized from the same base checkpoint but fine-tuned with different objectives. For our first comparison, we evaluated FirstZephyr$_\beta$ and RankZephyr, both initialized from Zephyr$_\beta$ but fine-tuned on the FIRST objective and $\mathcal{L}_{LM}$ respectively. Results in Table 2 show that both models achieved comparable effectiveness, with RankZephyr (RankZephyr$_g$) demonstrating a slight advantage on average. To validate these findings, we conducted a second comparison between FirstMistral and RankMistral, both initialized from Mistral-7B-Instruct-v0.3 but again fine-tuned on FIRST and $\mathcal{L}_{LM}$ respectively. As shown in Table 4, both models demonstrated consistent effectiveness across all datasets. These results provide robust evidence that, despite using only the first token, the FIRST objective maintains its ranking effectiveness.

Efficiency. To quantify the computational efficiency gains of FIRST's single-token approach versus full-sequence generation, we measured inference latency using NVIDIA Nsight Systems,[5] as well as the input and output token counts across the TREC DL19–23 datasets. All experiments were conducted on a single NVIDIA RTX 4090 to reduce confounding factors stemming from orchestration across multiple GPUs.

[5]https://developer.nvidia.com/nsight-systems

	RankZephyr		RankMistral	
	(g)	(l)	(g)	(l)
DL19 (43 queries)				
# In	18247.5	18293.8	18323.3	18419.2
# Out	711	18	711	18
Latency	3.13 s	1.92 s	3.13 s	1.89 s
Speedup	-	38.7%	-	39.6%
DL20 (54 queries)				
# In	18119.9	18182.6	18109.0	18189.4
# Out	711	18	711	18
Latency	3.10 s	1.83 s	3.06 s	1.79 s
Speedup	-	41.0%	-	41.5%
DL21 (53 queries)				
# In	15681.9	15708.4	15482.7	15552.4
# Out	711	18	711	18
Latency	2.59 s	1.55 s	2.48 s	1.51 s
Speedup	-	40.2%	-	39.1%
DL22 (76 queries)				
# In	15812.1	15834.9	15674.3	15741.1
# Out	711	18	711	18
Latency	2.76 s	1.67 s	2.64 s	1.58 s
Speedup	-	39.5%	-	40.2%
DL23 (82 queries)				
# In	15753.1	15791.5	15666.3	15736.7
# Out	711	18	711	18
Latency	2.58 s	1.59 s	2.52 s	1.57 s
Speedup	-	38.3%	-	37.7%

Table 5: Comparison of average input token counts (# In), average output tokens (# Out), and average reranking latency per query on DL19–23. Under RankZephyr, (g) denotes the model reranking by generating the full permutation of document identifiers, and (l) denotes reranking using the logits of the first identifier only. Similar notations are used for RankMistral. Speedup shows the percentage decrease in latency from (g) to (l) in the corresponding model.

Table 5 compares the average number of input tokens, output tokens per query, and the average inference latency per query between full-sequence generation and FIRST's single-token approach across two models, RankZephyr and RankMistral. The results demonstrated substantial computational savings, with FIRST reducing inference time by 40% per query across the five datasets.

These efficiency gains, consistent across different model architectures, confirm that FIRST is a competitive listwise reranking

approach, offering faster inference while maintaining the effectiveness demonstrated in previous experiments. Such latency benefits are particularly critical as these models are increasingly deployed in real-world serving settings, where response time directly impacts user experience and infrastructure costs [17].

5.6 Varying Window and Step Sizes

RQ6 What are the effectiveness–efficiency tradeoffs between different window and step sizes when using FIRST?

A keen reader may have noticed that in Table 5, the average number of output tokens is consistently 711 for full-sequence generation and 18 for FIRST. Indeed, this stems from the fixed window size of 20 and step size of 10. FIRST rerankers are trained to output 2 tokens per rerank: one for the alphabetical identifier and one for the closing identifier. Given that we rerank the top 100 retrieved documents using a window size of 20 and step size of 10, precisely 9 rerank LLM calls are performed per query, yielding $2 \times 9 = 18$ output tokens. A similar calculation can be done for full-sequence generation rerankers.

That is, varying the window and step sizes will directly impact the efficiency per query, and intuitively also the effectiveness. To further understand the effectiveness–efficiency tradeoffs FIRST has to offer, we conducted an ablation study across different parameterizations of (window size/step size) on FIRST. Following RankZephyr [16], we compared parameterizations of (20/10), (10/5), (2/1), and our results align with those from RankZephyr: (20/10) yields higher effectiveness than (10/5) and (2/1), as presented in Table 6, with significant improvements over (2/1) on most datasets. In fact, (20/10) is also more efficient as the bigger window size requires fewer rerank LLM calls.

However, this outcome is somewhat expected, courtesy of the nDCG@10 metric rewarding models that effectively rank the *top 10 documents*: among the 3 parameterizations, only (20/10) has a large enough window size to promote 10 documents from the last 10 candidates to the top 10 candidates. Thus, to further investigate the effect of parameterization in fairness of nDCG@10, we introduced two additional parameterizations: (20/5) and (20/2), both having a long window size while varying step sizes. As shown in Table 6, (20/10), (20/5), (20/2) achieve comparable effectiveness, with (20/5) demonstrating modest advantage on average. However, (20/5) increases latency per query by approximately 85% compared to (20/10), while (20/2) incurs more than 4 times the latency. Given this steep efficiency cost, we regard (20/10), the original parameterization in RankZephyr and FIRST, as the parameterization of choice, balancing efficiency and effectiveness.

w/s	# In	# Out	Latency	nDCG@10	# Calls
DL19 (43 queries)					
a: (2/1)	33878.4	198	3.50 s	0.7309	99
b: (10/5)	21205.2	38	2.10 s	0.7412	19
c: (20/10)	18419.2	18	1.89 s	0.7678	9
d: (20/5)	34928.6	34	3.46 s	0.7727	17
e: (20/2)	84834.9	82	8.35 s	**0.7749**	41
DL20 (54 queries)					
a: (2/1)	33794.8	198	3.38 s	0.7193	99
b: (10/5)	20773.4	38	2.07 s	0.7589^{a}	19
c: (20/10)	18189.4	18	1.79 s	0.7901^{a}	9
d: (20/5)	34351.5	34	3.38 s	$\mathbf{0.7914}^{a,b}$	17
e: (20/2)	82861.2	82	8.06 s	0.7865^{a}	41
DL21 (53 queries)					
a: (2/1)	31701.0	198	3.18 s	0.6847	99
b: (10/5)	18073.7	38	1.80 s	0.7589^{a}	19
c: (20/10)	15552.4	18	1.51 s	0.7694^{a}	9
d: (20/5)	29425.5	34	2.91 s	$\mathbf{0.7746}^{a}$	17
e: (20/2)	70905.1	82	6.89 s	0.7656^{a}	41
DL22 (76 queries)					
a: (2/1)	31372.1	198	3.15 s	0.5718	99
b: (10/5)	18161.0	38	1.81 s	0.6985^{a}	19
c: (20/10)	15741.1	18	1.58 s	$\mathbf{0.7030}^{a}$	9
d: (20/5)	29741.2	34	2.93 s	0.6996^{a}	17
e: (20/2)	71805.9	82	7.07 s	0.7029^{a}	41
DL23 (82 queries)					
a: (2/1)	31403.5	198	3.16 s	0.4728	99
b: (10/5)	18186.0	38	1.86 s	0.5705^{a}	19
c: (20/10)	15736.7	18	1.57 s	0.5743^{a}	9
d: (20/5)	29714.3	34	3.04 s	0.5809^{a}	17
e: (20/2)	71777.1	82	7.17 s	$\mathbf{0.5811}^{a}$	41

Table 6: Comparison of (window size/step size) for RankMistral at (2/1), (10/5), (20/10), (20/5), and (20/2) on the DL19–23 datasets. # In, # Out, Latency, # Calls are the average number of input tokens, output tokens, latency, and number of rerank LLM calls per query on top 100 retrieved documents, respectively. Different (window size/step size) parameterizations are numbered from "a" to "e". The superscripts in the nDCG@10 column indicate statistically significant improvements (paired Student's t-test with $p \leq 0.01$ with Bonferroni correction). e.g., ^{a} indicates that the entry is significantly higher than the entry in row "a" of that dataset.

6 Conclusion

Through a comprehensive study guided by six research questions, we expanded understanding of the capabilities of FIRST across multiple dimensions, validating its promise as a more efficient yet effective alternative to traditional LLM reranking.

More specifically, we demonstrated that FIRST successfully generalizes beyond its original Zephyr$_\beta$ implementation to other LLM backbones (**RQ1**), with robust effectiveness across diverse topics in out-of-domain TREC Deep Learning datasets (**RQ2**). This strong generalization validated the practical utility of FIRST across diverse retrieval scenarios and suggested that its single-token approach captures generalizable relevance signals.

Further, we showed that while LM training does improve zero-shot single-token reranking capabilities (**RQ3**), it may paradoxically

hinder subsequent FIRST fine-tuning, raising questions about the relationship between language modeling and ranking objectives.

Regarding first-stage retrievers (**RQ4**), we observed that FIRST exhibited patterns of diminishing returns with stronger retrievers, mirroring results from traditional LLM rerankers. This suggests that despite changes in both inference method and training objective, the fundamental dynamics of multi-stage retrieval remain consistent.

Finally, our latency analysis—both in comparison with traditional full-sequence generation (**RQ5**) and across different parameterizations for FIRST (**RQ6**)—provided a clear quantification of the concrete efficiency–effectiveness tradeoffs. We showed that FIRST reduces inference time by 40% compared to full-sequence generation while maintaining comparable effectiveness, and validated the optimal choice of window and step sizes in the original work.

To conclude, our findings demonstrated the reproducibility, robustness, and effectiveness of FIRST, suggesting that full-sequence generation may be unnecessarily verbose for ranking tasks. However, challenges remain, particularly the tokenization inconsistencies with alphabetical identifiers and the underlying tension between language modeling and ranking objectives, as evidenced by the relatively poor effectiveness of FirstRankZephyr and FirstLLaMA. These issues point to promising research directions: future work should explore ways to better integrate ranking objectives more explicitly into language model training, refine tokenization strategies to ensure stable document identifiers, and further investigate the interplay between learning-to-rank losses and language modeling to improve the effectiveness of single-token reranking.

Acknowledgments

This research was supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada. Additional funding is provided by Microsoft via the Accelerating Foundation Models Research program.

References

[1] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. 2018. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. arXiv:1611.09268 [cs.CL]

[2] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2020. Overview of the TREC 2020 Deep Learning Track. In *Proceedings of the Twenty-Ninth Text REtrieval Conference Proceedings (TREC 2020)*.

[3] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Jimmy Lin. 2021. Overview of the TREC 2021 Deep Learning Track. In *Proceedings of the Thirtieth Text REtrieval Conference (TREC 2021)*.

[4] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, Jimmy Lin, Ellen M. Voorhees, and Ian Soboroff. 2022. Overview of the TREC 2022 Deep Learning Track. In *Proceedings of the Thirty-First Text REtrieval Conference (TREC 2022)*.

[5] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. 2019. Overview of the TREC 2019 Deep Learning Track. In *Proceedings of the Twenty-Eighth Text REtrieval Conference Proceedings (TREC 2019)*.

[6] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Hossein A. Rahmani, Daniel Campos, Jimmy Lin, Ellen M. Voorhees, and Ian Soboroff. 2023. Overview of the TREC 2023 Deep Learning Track. In *Proceedings of the Thirty-Second Text REtrieval Conference (TREC 2023)*.

[7] Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 2022. From Distillation to Hard Negative Sampling: Making Sparse Neural IR Models More Effective. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Madrid, Spain) *(SIGIR '22)*. Association for Computing Machinery, New York, NY, USA, 2353–2359. doi:10.1145/3477495.3531857

[8] Neel Jain, Ping yeh Chiang, Yuxin Wen, John Kirchenbauer, Hong-Min Chu, Gowthami Somepalli, Brian R. Bartoldson, Bhavya Kailkhura, Avi Schwarzschild, Aniruddha Saha, Micah Goldblum, Jonas Geiping, and Tom Goldstein. 2023. NEFTune: Noisy Embeddings Improve Instruction Finetuning. arXiv:2310.05914 [cs.CL]

[9] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7B. arXiv:2310.06825 [cs.CL]

[10] Carlos Lassance, Ronak Pradeep, and Jimmy Lin. 2024. Naverloo @ TREC Deep Learning and NeuCLIR 2023: As Easy as Zero, One, Two, Three — Cascading Dual Encoders, Mono, Duo, and Listo for Ad-Hoc Retrieval. In *Proceedings of the Thirty-Second Text REtrieval Conference (TREC 2023)*. NIST.

[11] Yibin Lei, Liang Ding, Yu Cao, Changtong Zan, Andrew Yates, and Dacheng Tao. 2023. Unsupervised Dense Retrieval with Relevance-Aware Contrastive Pre-Training. In *Findings of the Association for Computational Linguistics: ACL 2023*, Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, Toronto, Canada, 10932–10940. doi:10.18653/v1/2023.findings-acl.695

[12] Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. 2024. Fine-Tuning LLaMA for Multi-Stage Text Retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Washington DC, USA) *(SIGIR '24)*. Association for Computing Machinery, New York, NY, USA, 2421–2425. doi:10.1145/3626772.3657951

[13] Xueguang Ma, Xinyu Zhang, Ronak Pradeep, and Jimmy Lin. 2023. Zero-Shot Listwise Document Reranking with a Large Language Model. arXiv:2305.02156 [cs.IR]

[14] Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. 2019. Multi-Stage Document Ranking with BERT. arXiv:1910.14424 [cs.IR]

[15] Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023. RankVicuna: Zero-Shot Listwise Document Reranking with Open-Source Large Language Models. arXiv:2309.15088 [cs.IR]

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

[17] Ronak Pradeep, Nandan Thakur, Sahel Sharifymoghaddam, Eric Zhang, Ryan Nguyen, Daniel Campos, Nick Craswell, and Jimmy Lin. 2025. Ragnarök: A Reusable RAG Framework and Baselines for TREC 2024 Retrieval-Augmented Generation Track. In *Proceedings of the 47th European Conference on Information Retrieval (ECIR 2025), Part I*. Lucca, Italy, 132–148.

[18] Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Le Yan, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, Xuanhui Wang, and Michael Bendersky. 2024. Large Language Models are Effective Text Rankers with Pairwise Ranking Prompting. arXiv:2306.17563 [cs.IR]

[19] Revanth Gangi Reddy, JaeHyeok Doo, Yifei Xu, Md Arafat Sultan, Deevya Swain, Avirup Sil, and Heng Ji. 2024. FIRST: Faster Improved Listwise Reranking with Single Token Decoding. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (Eds.). Association for Computational Linguistics, Miami, Florida, USA, 8642–8652. doi:10.18653/v1/2024.emnlp-main.491

[20] Stephen Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends in Information Retrieval* 3, 4 (2009), 333–389.

[21] 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. arXiv:2304.09542 [cs.CL]

[22] Manveer Singh Tamber, Ronak Pradeep, and Jimmy Lin. 2023. Scaling Down, LiTting Up: Efficient Zero-Shot Listwise Reranking with Seq2seq Encoder-Decoder Models. arXiv:2312.16098 [cs.IR]

[23] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A Heterogenous Benchmark for Zero-shot Evaluation of Information Retrieval Models. arXiv:2104.08663 [cs.IR]

[24] 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. arXiv:2310.16944 [cs.LG]

[25] Xinyu Zhang, Sebastian Hofstätter, Patrick Lewis, Raphael Tang, and Jimmy Lin. 2023. Rank-without-GPT: Building GPT-Independent Listwise Rerankers on Open-Source Large Language Models. arXiv:2312.02969 [cs.CL]

[26] Yutao Zhu, Huaying Yuan, Shuting Wang, Jiongnan Liu, Wenhan Liu, Chenlong Deng, Haonan Chen, Zheng Liu, Zhicheng Dou, and Ji-Rong Wen. 2024. Large Language Models for Information Retrieval: A Survey. arXiv:2308.07107 [cs.CL]

[27] Honglei Zhuang, Zhen Qin, Kai Hui, Junru Wu, Le Yan, Xuanhui Wang, and Michael Bendersky. 2024. Beyond Yes and No: Improving Zero-Shot LLM Rankers via Scoring Fine-Grained Relevance Labels. arXiv:2310.14122 [cs.IR]

[28] Shengyao Zhuang, Honglei Zhuang, Bevan Koopman, and Guido Zuccon. 2024. A Setwise Approach for Effective and Highly Efficient Zero-shot Ranking with Large Language Models. arXiv:2310.09497 [cs.IR]