Rank-without-GPT: Building GPT-Independent Listwise Rerankers on Open-Source Large Language Models

Crystina Zhang,[1]* Sebastian Hofstätter,[2] Patrick Lewis,[2] Raphael Tang,[3] Jimmy Lin[1]

[1]University of Waterloo [2]Cohere [3]Comcast Applied AI
[1]{xinyucrystina.zhang, jimmylin}@uwaterloo.ca
[2]{sebastian, patrick}@cohere.com [3]raphael_tang@comcast.com

Abstract. *Listwise* rerankers based on large language models (LLMs) are the zero-shot state of the art. However, current work in this direction all depend on GPT models, making them a single point of failure in scientific reproducibility. In this work, we lift this pre-condition and build effective listwise rerankers without any form of dependency on GPT for the first time. Our passage retrieval experiments show that our best listwise reranker surpasses the listwise rerankers based on GPT-3.5 by 13% and achieves 97% effectiveness of the ones based on GPT-4. Our results also show that the existing training datasets, which were expressly constructed for *pointwise* ranking, are insufficient for building such listwise rerankers. Instead, high-quality listwise ranking data is required and crucial, calling for further work on building human-annotated listwise data resources.

1 Introduction

Given a user query, the objective of *text retrieval* is to fetch an ordered list of documents among potentially billions of documents. Mainstream solutions to this problem follow a multi-stage ranking pipeline. In the first stage, *retrievers* are designed to efficiently retrieve the top-k candidates from the entire collection, followed in further stages by the application of *rerankers*, which refine the ranking of the returned candidate documents.

Rerankers are traditionally constructed in a *pointwise* paradigm, where given a query, the rerankers produce a relevance score for each passage independently, and the final ranking is formed by sorting passages by their relevance scores. Recently, attracted by the strong generative power of large language models (LLMs) and their capacity to consume long-context inputs, a new paradigm of neural rerankers has been proposed using *listwise* ranking [16, 28, 22, 29]. These models consume a combined list of passages at a time and directly outputs the reordered ranking list.[1]

Not only does listwise reranker achieve better scores on IR benchmarks [29], it also provides a new perspective to passage reranking paradigm: it questions the necessity to convert the ranking task into a classification task, and instead frames it as a pure text generation task that could be solved end-to-end in a generalized text-to-text fashion [24]. For the first time, the model directly generates the entire ranking list in the form of text, instead of requiring multiple disjoint inference passes of the model as in pointwise [18,

[*] Work is done during internship at Cohere.

[1] Note that this is different from the listwise loss [3]. See details in Section 2.2.

19] or pairwise rerankers [23, 21]. This integrates passage retrieval into the unified framework established in NLP, and shows its potential to be merged seamlessly with other text-to-text tasks and leveraging existent prompting techniques [30, 14].

However, while existing work on listwise reranking demonstrates the promising application of this new ranking paradigm, their success *crucially depends* on GPT models, which are either directly used as reranker models at inference time [16, 28] or indirectly used as teacher models at the training time [22]. Such exclusive dependence results in a single point of failure in scientific reproducibility. Moreover, it raises the concern that the current progress on listwise rerankers is only applicable to the proprietary GPT-based models instead of the general LLMs. In this work, we seek to reduce the reliance of listwise rerankers on GPT models and diversify the solution options for constructing such models. Results show that, for the first time, our best listwise reranker built without any form of GPT dependence surpasses the rerankers based on GPT-3.5 by 13% and achieves 97% effectiveness of ones based on GPT-4, measured by nDCG@10 on two passage retrieval datasets.

In this process, we found the current IR training data, which was constructed in order to train pointwise rerankers, is far from sufficient for training listwise rerankers (Fig. 1, Section 3.1), yielding worse results than using data generated by BM25, a non-neural lexical technique in IR. While silver ranking data generated by current rerankers serves as a good approximation of the gold ranking, the performance of listwise rerankers increases linearly with training data ranking quality — a relationship that has not yet plateaued (Section 5). This indicates that the models are likely to further benefit from training data of higher quality, calling for future work on building human-annotated datasets purpose-designed for listwise training.

The main purpose of our work is to advocate diverse solutions for future listwise reranking research. Our contributions are as follows: (1) We are first to show that the listwise rerankers, without any form of dependency on the GPT models, could outperform the listwise rerankers based on GPT-3 or 3.5 and perform on par with the ones based on GPT-4; (2) We find that the ranking quality in the training data is crucial in constructing efficient listwise rerankers, which might be the bottleneck of the current capacity of the listwise rerankers; (3) We demonstrate that listwise reranker fine-tuning is data-efficient, where an effective listwise reranker can be built using 5k queries, each associated with a list of passages ranked in high quality, showing that it is feasible to build a human-annotated listwise dataset for this purpose.

2 Background

2.1 Pointwise Reranking

Given a query q and a passage p_i at the same time, pointwise rerankers h_{pw} produce a real score $s_i := h_{\text{pw}}(q, p_i)$ indicating the relevance of the passage p_i to the query q. The model is optimized using cross entropy [18, 19] or contrastive loss [11, 20, 33, 15, 31], based on binary relevance judgments from human annotators. At inference time, given the top-k passages $\{p_i\}_{i=1}^{k}$ returned by the previous-stage retriever, the model computes the relevance scores $\{s_i\}_{i=1}^{k}$ for each p_i independently. The final passages are then ranked by decreasing the magnitude of their corresponding relevance scores.

2.2 Listwise Reranking

As opposed to pointwise rerankers, which rank passages according to their individual predicted relevance scores to the query, listwise rerankers are designed to directly predict the final *ranking* of a list of passages as a whole, This not only allows the models to inter-reference the candidate passages to better determine their order, but also frames the passage retrieval task as text generation and thus fuse well with the existent techniques based on generative models. Using an LLM as a listwise reranker is concurrently studied in RankGPT [28] and LRL [16], where both works use GPT-based models.

We formulate listwise rerankers under the same preliminaries as the pointwise one: given the instruction prompt s, the query q, and an input sequence of top-k passages $\{p_i\}_{i=1}^k$, the listwise-ranking LLM h_{lw} returns the final ranked passages $\hat{\mathcal{P}} := h_{\text{lw}}(q, \{p_i\}_{i=1}^k; s)$, where $\hat{\mathcal{P}}$ is a permutation (reranking) of $\{p_i\}_{i=1}^k$.

Sliding window. Limited by the maximum input length, we can feed only 10–20 passages to the LLM at a time. To rerank a longer list, e.g. typically top-100 passages, both RankGPT and LRL adopt a *sliding window strategy*, where we slide a window of size n from the end to the front of the list and rerank the documents in the window, striding by m documents per step. In each stride, the top-$(n - m)$ documents are preserved and form the next sliding window, together with the next m documents.

Fine-tuning listwise-ranking LLMs. Used directly out of the box, current open-source LLMs often generate ill-formed outputs from listwise prompts [23, 22], where few valid ranking results can be inferred. Thus, our work focuses on the condition of fine-tuning LLMs, which helps the models follow the instructions and generate valid outputs. However, we found that the current human-annotated training data for IR is insufficient for this purpose, which we elaborate in Section 3.1.

Difference from listwise loss. Note that the listwise ranking mentioned in this work is different from the listwise loss in information retrieval (IR; [3]), where models still generate the score for each passage independently, although the loss is computed by leveraging scores of a list of documents. The term listwise in this work refers to that the model is capable of processing a list of documents at the same time.

3 Method

3.1 Training Data for Listwise Reranker

The difference in the output format of the two above rerankers by nature requires different types of training data. Past experience shows that a large-scale professionally annotated dataset with binary judgments, e.g., MS MARCO [1], is sufficient in fine-tuning pointwise rerankers. These pointwise datasets consist of queries, documents, and binary query–document labels, annotated to denote document relevance to the query. Unannotated documents are considered irrelevant by default. (Fig. 1, Block I, Block III)

However, there are challenges in constructing gold rankings using current resources for two main reasons. First, there are many false-negative passages. Taking MS MARCO

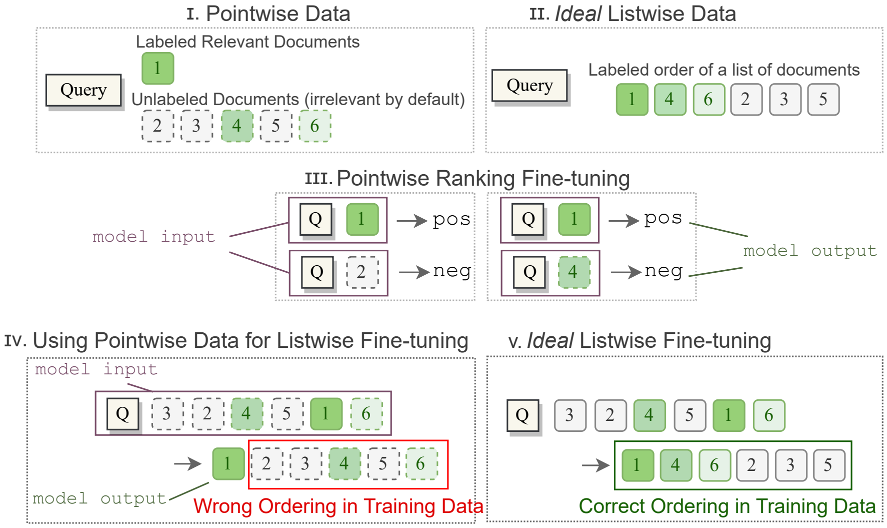

I. Pointwise Data
Labeled Relevant Documents
Query
1
Unlabeled Documents (irrelevant by default)
2 3 4 5 6
II. *Ideal* Listwise Data
Query
Labeled order of a list of documents
1 4 6 2 3 5
III. Pointwise Ranking Fine-tuning
model input
Q 1 → pos
Q 2 → neg
Q 1 → pos
Q 4 → neg
model output
IV. Using Pointwise Data for Listwise Fine-tuning
model input
Q 3 2 4 5 1 6
→ 1 2 3 4 5 6
model output
Wrong Ordering in Training Data
V. *Ideal* Listwise Fine-tuning
Q 3 2 4 5 1 6
→ 1 4 6 2 3 5
Correct Ordering in Training Data

Fig. 1: The issue with using current pointwise ranking data in listwise training. Numbers in the boxes indicate different passages. The grey boxes indicate irrelevant passages and the green ones indicate relevant ones. The saturation level indicates the relevance: the more saturating the green is, the more relevant the passages are. Boxes with dash borders indicate unlabeled passages, which are considered irrelevant in the current convention. Thus, the green boxes with dash borders are the false negative passages.

as an example, which is the largest training data in text retrieval, there is only one labeled passage per query on average. In a list of, say, twenty retrieved passages, only one at most is known to be in the correct position (the first one), whereas the positions of the other nineteen are unknown. This may result in an extremely noisy ordering. Second, true relevance is nuanced and *graded* (multilevel) rather than binary, as TREC evaluation sets show. Binary relevance ignores nuances in the true relevance levels and discards the correct order of relevant passages, thus resulting in a suboptimal ordering of the passage list. We illustrate these two issues in Fig. 1, Block IV.

To verify the above hypothesis that the ordering of the ranking list is crucial for fine-tuning listwise rerankers, we designed two sets of experiments:

Pointwise ground truth (P-GT). We construct a list by placing the labeled relevant documents in the front, which are then followed by the non-relevant ones ordered arbitrarily. This is used as a sanity baseline, showing the effectiveness when only using the human-annotated training data in the pointwise ranking manner.

Silver ranking. We use the ranking results of several existent ranking systems to approximate the gold ranking. Specifically, we select the following ranking systems, which are listed in order of increasing ranking capability and thus generating listwise training data with increasing ranking quality:

a) **BM25**: Passages are ranked by BM25 [25], a traditional unsupervised retrieval algorithm based on lexical matching.
b) **Fine-tuned Contriever (Contriever+ft)**: Passages are ranked by Contriever [12] that has been further fine-tuned on MS MARCO. We used the checkpoint released by the original work.[2]
c) **`co.rerank`**: Passages are ranked by model `rerank-english-v2.0`[3] through Cohere rerank API.[4]

3.2 Prompt

We adopt the same prompt as RankGPT for a fair comparison of the results:

Input Prompt Template:

```
USER: I will provide you with {num} passages, each
indicated by a numerical identifier []. Rank the
passages based on their relevance to the search
query: {query}.
[1] {title 1} {passage 1}
[2] {title 2} {passage 2}
...
[{num}] {passage {num}}
Search Query: {query}.
Rank the {num} passages above based on their
relevance to the search query. All the passages
should be included and listed using identifiers, in
descending order of relevance. The output format
should be [] > [], e.g., [4] > [2]. Only respond
with the ranking results, do not say any word
or explain.
```

Example Completion:

```
[4] > [5] > [2] > [3] > [1]
```

4 Experimental Setup

4.1 Models

Most of the experiments in the work are conducted on the open-source LLM, Code-LLaMA-Instruct [26],[5]. We experiment with all released model sizes: 7B, 13B, and 34B. In ablation studies, we compare the results to Vicuna-v1.5,[6] another model that is based on Llama 2 but fine-tuned on ShareGPT, instructional data generated by GPT.

[2] `https://huggingface.co/facebook/contriever-msmarco`

[3] `https://huggingface.co/Cohere/rerank-english-v2.0/tree/main`

[4] `https://cohere.com/rerank`

[5] `https://huggingface.co/codellama`

[6] `https://huggingface.co/lmsys/vicuna-7b-v1.5`

4.2 Data

Training data preparation. The training data are prepared from MS MARCO v1 corpus [1], which contains 8.8 million passages. We sampled n training queries from the $100k$ training data of RankVicuna ($n \in \{2k, 5k, 10k, 20k\}$), then reordered the list of documents per query in the four settings mentioned in Section 3.1.

Evaluation datasets. We select TREC-DL-19 and TREC-DL-20 [6, 5] to evaluate the in-domain effectiveness. Both datasets are built from the TREC Deep Learning Track and share the same corpus with MS MARCO v1 [1]. In Section 5.4, we report results reranking top-100 candidates returned by BM25 [25] and RepLLaMA [15]. We report scores of nDCG@10 following the dataset standard. In Section 5.4, we additionally report Judged@10 on some experiment settings, the ratio of judged passages in the top-10 of the ranking list.

4.3 Configurations

In this work, we use FastChat [32][7] for the model training and inference. FlashAttention [9, 8] is applied to all experiments. We turned on gradient checkpointing when fine-tuning 34B models. When not specified, we fine-tune the model with batch size 128. The maximum input length is set as 4,096. In all experiments, we keep the total number of steps the same, thus the number of fine-tuning epochs depends on the number of training datapoints. The model is fine-tuned for 4 epochs when using 20k training data, 8 epochs when using 10k training data, so on and on.

In experiments using QLoRA, we set LoRA rank as 64, alpha as 16, dropout rate as 0.05, maximum gradient norm as 0.3, and a constant learning rate of 1×10^{-4} following previous works [10]. LoRA is applied on `q_proj` and `v_proj` layers. In experiments that fine-tune the entire LLM, we use a learning rate of 2×10^{-5} with the cosine learning schedule. All experiments are run on 8 A100 GPUs with 80 GB memory. With QLoRA, training 7B models takes around 5 hours when fine-tuning 20k training data for 4 epochs.

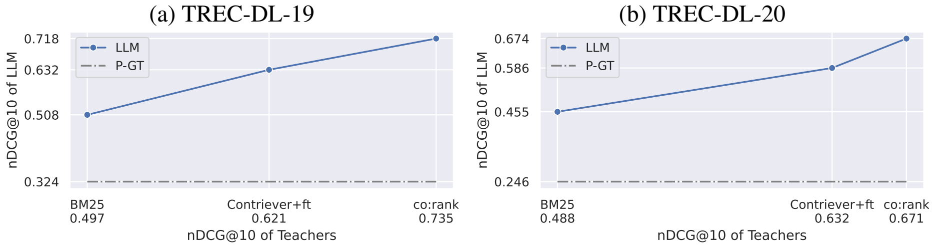
5 Results and Analysis

5.1 Training Data Quality

We first show that the current pointwise labeled data alone could not serve the need of fine-tuning generative LLM as listwise rerankers. While the ranking results produced by current rerankers could be used as an approximation of the gold ranking, listwise rerankers are likely to further benefit from human-labeled listwise data in higher quality.

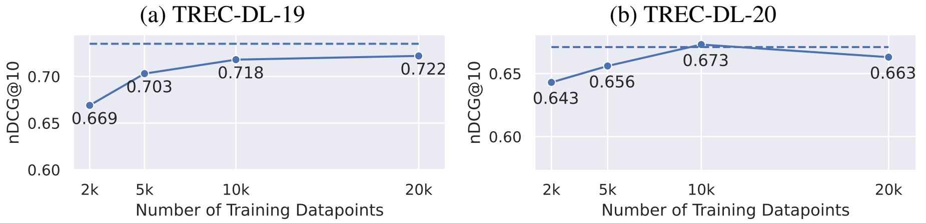
Fig. 2 shows the results on TREC-DL-19 and TREC-DL-20 of the listwise rerankers when fine-tuned on different training data. The x-axis is the nDCG@10 of the pointwise rerankers that generate the training data, and the y-axis is the nDCG@10 of the listwise rerankers fine-tuned on the corresponding data. The horizontal dash line is the result when the model is only fine-tuned on the ground-truth pointwise data.

[7] `https://github.com/lm-sys/FastChat`


(a) TREC-DL-19

(b) TREC-DL-20

Fig. 2: nDCG@10 on TREC-DL-19 and TREC-DL-20 when fine-tuned on data prepared on methods described in Section 3.1. **P-GT**: Pointwise ground truth.


(a) TREC-DL-19

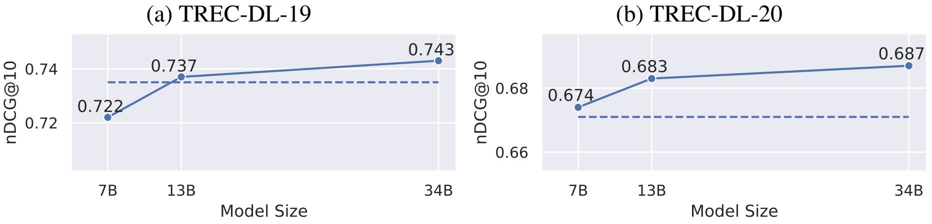
(b) TREC-DL-20

Fig. 3: Results regarding the increasing number of training data generated by `co.rerank`. Dash lines refer to the result of `co.rerank`.

Clearly, listwise rerankers fine-tuned only the pointwise data yield inferior ranking quality, evidenced by that the grey line is greatly lower than others. When fine-tuned on the silver ranking data, the scores of the listwise rerankers follow closely to the scores of pointwise rerankers (e.g., scores on pointwise vs. corresponding listwise reranker: 0.497 vs. 0.508, 0.621 vs. 0.632, 0.735 vs. 0.718). On the one hand, this shows that the quality of rankings data is crucial when fine-tuning the listwise rerankers; on the other hand, the listwise student is able to keep up with even one of the best current teachers without showing a trend of plateau. This hints that the potential capacity of the listwise rankers may not be fully excavated and may be bounded by the quality of current training data. That is, if higher-quality listwise training data were available (e.g., by human labeling), the listwise rankers might show higher ranking capacity.

5.2 Training Data Quantity

Having proved that higher-quality data is necessary to obtain effective listwise rerankers, we ask the next question: *how much data is required?* Fig. 3 compares the model effectiveness with an increasing amount of fine-tuning data. For a fair comparison, models are fine-tuned for the same number of steps when varying training data quantity: where models are fine-tuned for 8 epochs with 10k datapoints, it is then fine-tuned for 40, 16, and 4 epochs when using 2k, 5k, and 20k datapoints, where each datapoint consists of one query and 20 passages. Therefore, training with fewer datapoints only saves the



(a) TREC-DL-19

(b) TREC-DL-20

Fig. 4: Result regarding different sizes (x-axis) of the model, all fine-tuned on $10k$ data. Dash lines refer to the result of `co.rerank`.

anticipated human labor effort for annotation but not the training time. Experiments are based on Code-LLaMA-Instruct in size 7B.

As Fig. 3 shows, training on 5k training datapoints already yield 97% of the effectiveness compared to using 10k data points, while increasing the amount of data from 10k to 20k only brings marginal improvement in the case of TREC-DL-19 and no positive effect on TREC-DL-20. That is, 100k high-quality query–passage pairs (5k queries with 20 passages per query) serve the need of effectively fine-tuning listwise rerankers. This is in the same scale with fine-tuning pointwise rerankers, where RankLLaMA [15] consumes 300k query–passage pairs from MS MARCO.

5.3 Model Size

The experiments above are all based on 7B-sized models. We then examine the effect of scaling up the models. As expected, the effectiveness of the listwise rerankers increases with the language model size. Fig. 4 shows the trend of the ranking quality with respect to the model size, where the model of 13B already outperforms the teacher, and increasing the model size to 34B brings additional improvement.

5.4 Comparisons with Other Baselines

Finally, we compare our listwise rerankers to other methods in the field, evaluated on TREC-DL-19 and TREC-DL-20. Results are shown in Table 1. The baselines are grouped into three categories: (1) non-listwise rerankers based on non-LLM models (e.g., BERT); (2) non-listwise rerankers based on LLM, including methods based on query likelihood [27], pairwise [23] and pointwise reranking [15]; (3) listwise rerankers [22, 16, 28, 29], which all depend on GPT models.

Unlabeled top-reranked passages. Although TREC-DL data have comparatively dense human judgments,[8] we observe that listwise rerankers bring more unlabeled passages to the top of the reranked list compared to the pointwise ones. For example, on TREC-DL-19, the Judged@10 of listwise rerankers are between 0.88 to 0.94, whereas the Judged@10 of RankLLaMA is over 0.98.

[8] 120 judgments per query on TREC-DL-19; 211 judgments per query on TREC-DL-20

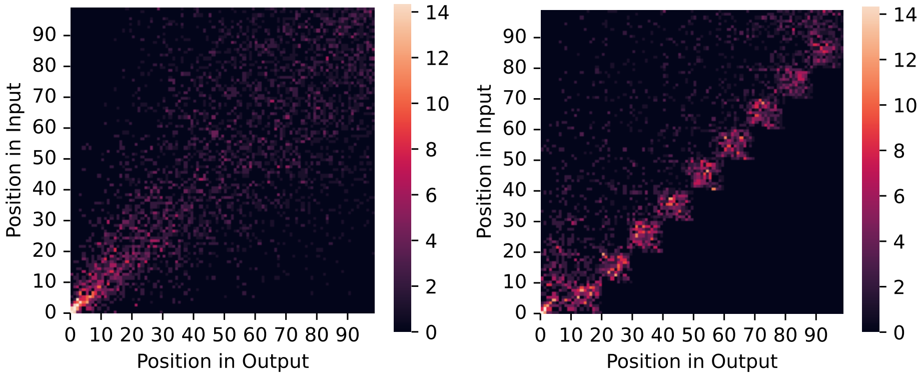
Table 1: Comparison of listwise reranker fine-tuned on data generated by `co.rerank` to other methods in the field, evaluated on TREC-DL-19 and TREC-DL-20. The *italicized* scores in bracket are evaluated on enriched query–passage relevance judgment, with Judged@10 = 1. Results of RankVicuna, LRL, and RankGPT-3.5 are copied from the original paper [22, 16, 28]. Results of RankGPT-4 reranking BM25 top-100 are copied from [29]. * indicates a significant difference from rankVicuna (row 7, based on Viccuna and distilled from GPT-3), † indicates a significant difference from LRL (row 8, based on GPT-3), and ‡ indicates a significant difference from RankGPT-3.5 (row 9, based on GPT-3.5), and none of the results is significantly different from RankGPT-4 (row 10, based on GPT-4). All significant tests are based on a two-tailed t-test with p < 0.01.

	Model		**GPT-independent**	**Model Size**	**Previous Stage**	**top-*k***	**TREC-DL-19** nDCG@10	**TREC-DL-20** nDCG@10
			non-listwise methods (not LLM-based)					
(1)	monoBERT	BERT	✓	110M	BM25	1000	72.3	72.2
(2)	monoT5	T5	✓	3B	BM25	100	71.8	68.9
(3)	rankT5	T5	✓	3B	BM25	100	71.2	69.5
			non-listwise methods (LLM-based)					
(4)	UPR	FLAN-T5-XXL	✓	11B	BM25	100	62.0	60.3
(5)	PRP-Sliding-10	FLAN-UL2	✓	20B	BM25	100	72.7	70.5
(6)	RankLLaMA	LLaMA	✓	7B	RepLLaMA	100	75.3 (*76.1*)	76.7 (*76.2*)
			listwise methods					
(7)	RankVicuna	Vicuna	✗	7B	BM25	100	66.8	65.5
(8)	LRL	GPT-3	✗	?	BM25	100	65.8	62.2
(9)	RankGPT-3.5	GPT-3.5	✗	?	BM25	100	65.8	62.9
(10)	RankGPT-4	GPT-4	✗	?	BM25	100	75.7	71.0
(11)	*Rank-wo-GPT*	Code-LLaMA-Instruct	✓	7B	BM25	100	71.8 (*70.8*)*†‡	67.4 (*66.7*)*†‡
(12)	*Rank-wo-GPT*		✓	7B	RepLLaMA	100	73.0 (*75.2*)*†‡	70.0 (*71.7*)*†‡
(13)	*Rank-wo-GPT*		✓	13B	BM25	100	73.7*†‡	68.3*†‡
(14)	*Rank-wo-GPT*		✓	34B	BM25	100	74.3*†‡	68.7*†‡

For a fair comparison, we manually annotated the missing query–passage relevance judgments from the top-10 of the lists returned by some of the rerankers, including both pointwise and listwise ones from rows (6, 11, 12). The labels are on the same scale as the original graded judgment (i.e., from 0 to 3, with larger numbers indicating higher relevance). These added labels, together with the initial ones, form the new judgment set, which we refer to as "*enriched judgments*".

Scores evaluated on our enriched judgments set are *italicized* in parentheses. We observe that the added judgment made a nontrivial difference to the evaluation results. Most prominently, the nDCG@10 on row (12) increased from 73.0 to 75.2 after filling in the missing relevance. Intact judgments also amend the over-rated rankings, for example, on row (11), the scores decreased with more labels. In the rest of this section, we compare results evaluated on the enriched judgments.

Comparison to GPT-based listwise rerankers. Comparing rows (11–14) to rows (7–10), we found even the smallest listwise reranker (7B) is significantly higher compared to previous listwise rerankers based on GPT-3 and GPT-3.5, and insignificantly lower than rerankers based on GPT-4.



(a) Pointwise Reranker

(b) Listwise Reranker

Fig. 5: Compare the position of relevant passages before and after reranking by RankLLaMA and Rank-wo-GPT both reranking RepLLaMA top-100. x-axis: passage positions in the reranked list; y-axis: passage positions in the first-stage list. Best viewed in color.

Comparison to LLM-based pointwise rerankers. While the pointwise rerankers are fine-tuned on the optimal human-annotated data, we find our listwise models, fine-tuned under data non-optimized for its purpose, perform close to the best pointwise rerankers in the same model size on TREC-DL-19. Comparing row (12) to row (6), where both listwise and pointwise rerankers are based on the same size of models (7B) and reranking the same candidates from the first-stage retriever, there is only a small gap between the nDCG@10 on TREC-DL-19, with *insignificant* difference (two-tailed t-test, with $p < 0.01$), although there is a larger gap on TREC-DL-20: 71.7 vs. 76.2 on rows (12, 6), with a significant difference, which requires future work to close the gap.

5.5 Analysis on Sliding Window Strategy

While the sliding window strategy is a natural resort to apply listwise ranking on a passage list longer than the model input capacity, it is unknown yet how well it aggregates the list in each pass.

To start answering this question, we plot the ranking positions of relevant passages before and after reranking. Fig. 5 compares the position difference when using the pointwise and listwise rerankers, the models on rows (6) and (12) in Table 1. In each heatmap, the y-axis indicates the passage position in the first-stage ranking (i.e., RepLLaMA) and the x-axis indicates the position after reranking by RankLLaMA (Fig. 5a) or Rank-wo-GPT (Fig. 5b).

Comparing the heatmaps, we observe a prominent pattern in the listwise heatmap (Fig. 5b) that there is a chain of bright clusters in the square shape along the diagonal line. This indicates that a large number of relevant documents are “trapped” in the local block, promoted only within the current or the next pass of the sliding window. We find this phenomenon common for relevant passages at all relevant levels.

Table 2: Results when using Code-LLaMA-Instruct and Vicuna as the initial LLM, and when fine-tuning Vicuna with QLoRA or all parameters (Full). All models are in size 7B and fine-tuned on 10k datapoints for 8 epochs.

	Model	**DL-19**	**DL-20**
(1) QLoRA	Code-LLaMA-Instruct	0.718	0.674
(2) QLoRA	Vicuna-v1.5	0.728	0.683
(3) Full	Vicuna-v1.5	0.727	0.674

The brightness density in the upper matrix indicates the frequency of relevant passages promoted over a long distance over the list. Compared to pointwise, where the scatters distribute symmetrically along the diagonal matrix, listwise heatmap shows more scatters clustered in left-most columns, $x \in [0, 20]$, indicating that the top-ranked passages by listwise rerankers still come from a wider range of positions in the first-stage results compared to the pointwise methods regardless of the large number of passages trapped, as aforementioned.

6 Ablation Studies

LLM with GPT-based instruction fine-tuning. To investigate if more GPT-like instruction fine-tuning would further benefit the listwise ranking results, we ran the same experiment on Vicuna-v1.5. As shown in rows (1, 2) in Table 2, while fine-tuning based on Vicuna achieved slightly better results on both datasets, the difference is not significant. Thus, we conclude that starting from a GPT-free LLM yields satisfactory effectiveness compared to a more GPT-like LLM.

Fine-tuning Full Model vs. QLoRA. In previous experiments, we fine-tuned the LLM using QLoRA instead of the entire LLM model to alleviate the GPU memory and disk requirement. Here, we compared the effectiveness of the two fine-tuning strategies on Vicuna.[9] As shown in rows (2, 3) in Table 2, fine-tuning with QLoRA yields similar effectiveness as fine-tuning all parameters on both datasets, with the same amount of training data and the fine-tuning epochs.

7 Related Work

In the past few years, the question of how generative models could bring benefits to information retrieval has been an area of intense study, with a number of differing and complementary techniques emerging. The strong generative performance of LLMs has been leveraged for retrieval by generating a large volume of synthetic datasets on domains: InPars [2, 13], and Promptagator [7].

[9] We conducted the same experiment in Code-LLaMA-Instruct; However, the results were not in the correct scale. Thus we use Vicuna as a replacement in this ablation.

In parallel, researchers have investigated whether LLMs could be used directly as retrievers or rerankers: SGPT [17] first shows that the GPT-based decoder models, are effective when used as bi-encoder in retrieval tasks. UPR [27] uses the query likelihood as the ranking score. PRP [23] shows that the LLM can effectively determine the comparative relevance regarding the query, given a pair of documents. Recently, RepLLaMA [15] demonstrates that fine-tuning LLAMA in the traditional paradigm of bi-encoder and pointwise cross-encoder surpasses smaller models.

Finally, a line of work that is mostly relevant to our work regards LLMs as black boxes and only uses the final generative output for ranking: RankGPT [28] and LRL [16] studied listwise rerankers concurrently, demonstrating their effectiveness using GPT-3, GPT-3.5, and GPT-4. RankVicuna [22] then showed that the method could be applied to a smaller-sized open-source LLM (e.g. Vicuna [4] in 7B, 13B) by distilling from GPT-3.5. PSC [29] proposed a permutation self-consistency prompting method, which alleviates the positional bias and largely improves the effectiveness of the listwise ranking.

8 Conclusions and Future Work

In this work, we study how to construct effective *GPT-free* listwise rerankers based on open-source LLM models. Experiments on two passage retrieval datasets show that our listwise rerankers, without any form of dependency on GPT, can substantially outperform the ones built on GPT-3 and perform on par with the ones built on GPT-4. In this process, we find that current pointwise training data in IR is not sufficient in fine-tuning listwise rerankers. Instead, training data comprised of high-quality ranked document lists is required and crucial. While the training data generated by current pointwise rerankers could be used as a nice approximation, the models are likely to benefit more from higher-quality listwise training data that are built from human annotations.

We hope this work sets the stage for future research on listwise ranking methods by bringing more diversity of solutions to the research in this area. Additionally, we hope it paves the path for future work on addressing text retrieval in the text generation paradigm, where it could be formatted in the same way as the other text-to-text tasks, and thus better integrated into the unified system.

Acknowledgments

This research was supported in part by the Canada First Research Excellence Fund and the Natural Sciences and Engineering Research Council (NSERC) of Canada. Computational resources were provided by Compute Ontario and Compute Canada.

References

1. Bajaj, P., Campos, D., Craswell, N., Deng, L., Gao, J., Liu, X., Majumder, R., McNamara, A., Mitra, B., Nguyen, T., et al.: MS MARCO: A human generated machine reading comprehension dataset. arXiv:1611.09268 (2016)

2. Bonifacio, L., Abonizio, H., Fadaee, M., Nogueira, R.: InPars: Unsupervised dataset generation for information retrieval. In: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 2387–2392 (2022)
3. Cao, Z., Qin, T., Liu, T.Y., Tsai, M.F., Li, H.: Learning to rank: From pairwise approach to listwise approach. In: Proceedings of the 24th International Conference on Machine Learning. p. 129–136. ICML '07, Association for Computing Machinery, New York, NY, USA (2007)
4. Chiang, W.L., Li, Z., Lin, Z., Sheng, Y., Wu, Z., Zhang, H., Zheng, L., Zhuang, S., Zhuang, Y., Gonzalez, J.E., Stoica, I., Xing, E.P.: Vicuna: An open-source chatbot impressing GPT-4 with 90%* ChatGPT quality (2023)
5. Craswell, N., Mitra, B., Yilmaz, E., Campos, D.: Overview of the TREC 2020 deep learning track. arXiv:2102.07662 (2021)
6. Craswell, N., Mitra, B., Yilmaz, E., Campos, D., Voorhees, E.M.: Overview of the TREC 2019 deep learning track. arXiv:2003.07820 (2020)
7. Dai, Z., Zhao, V.Y., Ma, J., Luan, Y., Ni, J., Lu, J., Bakalov, A., Guu, K., Hall, K.B., Chang, M.W.: Promptagator: Few-shot dense retrieval from 8 examples. arXiv:2209.11755 (2022)
8. Dao, T.: FlashAttention-2: Faster attention with better parallelism and work partitioning. arXiv:2307.08691 (2023)
9. Dao, T., Fu, D.Y., Ermon, S., Rudra, A., Ré, C.: FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In: Advances in Neural Information Processing Systems (2022)
10. Dettmers, T., Pagnoni, A., Holtzman, A., Zettlemoyer, L.: QLoRA: Efficient finetuning of quantized LLMs. arXiv:2305.14314 (2023)
11. Gao, L., Dai, Z., Callan, J.: Rethink training of BERT rerankers in multi-stage retrieval pipeline. In: Advances in Information Retrieval: 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28 – April 1, 2021, Proceedings, Part II. p. 280–286. Springer-Verlag, Berlin, Heidelberg (2021)
12. Izacard, G., Caron, M., Hosseini, L., Riedel, S., Bojanowski, P., Joulin, A., Grave, E.: Unsupervised dense information retrieval with contrastive learning. arXiv:2112.09118 (2021)
13. Jeronymo, V., Bonifacio, L., Abonizio, H., Fadaee, M., Lotufo, R., Zavrel, J., Nogueira, R.: InPars-v2: Large language models as efficient dataset generators for information retrieval. arXiv:2301.01820 (2023)
14. Liu, N.F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., Liang, P.: Lost in the middle: How language models use long contexts. arXiv:2307.03172 (2023)
15. Ma, X., Liang, W., Yang, N., Furu, W., Lin, J.: Fine-tuning LLaMA for multi-stage text retrieval. arXiv:2309.15088 (2023)
16. Ma, X., Zhang, X., Pradeep, R., Lin, J.: Zero-shot listwise document reranking with a large language model. arXiv:2305.02156 (2023)
17. Muennighoff, N.: SGPT: GPT sentence embeddings for semantic search. arXiv:2202.08904 (2022)
18. Nogueira, R., Cho, K.: Passage re-ranking with BERT. arXiv:1901.04085 (2019)
19. Nogueira, R., Jiang, Z., Lin, J.: Document ranking with a pretrained sequence-to-sequence model. arXiv:2003.06713 (2020)
20. Pradeep, R., Liu, Y., Zhang, X., Li, Y., Yates, A., Lin, J.: Squeezing water from a stone: A bag of tricks for further improving cross-encoder effectiveness for reranking. In: Advances in Information Retrieval. pp. 655–670. Springer International Publishing, Cham (2022)
21. Pradeep, R., Nogueira, R., Lin, J.: The expando-mono-duo design pattern for text ranking with pretrained sequence-to-sequence models. arXiv:2101.05667 (2021)
22. Pradeep, R., Sharifymoghaddam, S., Lin, J.: RankVicuna: Zero-shot listwise document reranking with open-source large language models. arXiv:2309.15088 (2023)

23. Qin, Z., Jagerman, R., Hui, K., Zhuang, H., Wu, J., Shen, J., Liu, T., Liu, J., Metzler, D., Wang, X., et al.: Large language models are effective text rankers with pairwise ranking prompting. arXiv:2306.17563 (2023)
24. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J.: Exploring the limits of transfer learning with a unified text-to-text transformer. J. Mach. Learn. Res. **21**(1) (jan 2020)
25. Robertson, S., Zaragoza, H., et al.: The probabilistic relevance framework: BM25 and beyond. Foundations and Trends in Information Retrieval (2009)
26. Roziere, B., Gehring, J., Gloeckle, F., Sootla, S., Gat, I., Tan, X.E., Adi, Y., Liu, J., Remez, T., Rapin, J., et al.: Code Llama: Open foundation models for code. arXiv:2308.12950 (2023)
27. Sachan, D., Lewis, M., Joshi, M., Aghajanyan, A., Yih, W.t., Pineau, J., Zettlemoyer, L.: Improving passage retrieval with zero-shot question generation. In: Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing. pp. 3781–3797. Abu Dhabi, United Arab Emirates (Dec 2022)
28. Sun, W., Yan, L., Ma, X., Ren, P., Yin, D., Ren, Z.: Is ChatGPT good at search? Investigating large language models as re-ranking agent. arXiv:2304.09542 (2023)
29. Tang, R., Zhang, X., Ma, X., Lin, J., Ture, F.: Found in the middle: Permutation self-consistency improves listwise ranking in large language models. arXiv:2310.07712 (2023)
30. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q.V., Zhou, D., et al.: Chain-of-thought prompting elicits reasoning in large language models. Advances in Neural Information Processing Systems **35**, 24824–24837 (2022)
31. Zhang, C., Li, M., Lin, J.: CELI: Simple yet effective approach to enhance out-of-domain generalization of cross-encoders. In: Duh, K., Gomez, H., Bethard, S. (eds.) Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers). pp. 188–196. Association for Computational Linguistics, Mexico City, Mexico (Jun 2024)
32. Zheng, L., Chiang, W.L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E.P., Zhang, H., Gonzalez, J.E., Stoica, I.: Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. arXiv:2306.05685 (2023)
33. Zhuang, H., Qin, Z., Jagerman, R., Hui, K., Ma, J., Lu, J., Ni, J., Wang, X., Bendersky, M.: RankT5: Fine-tuning T5 for text ranking with ranking losses. In: Proc. of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR) (2023)