TourRank: Utilizing Large Language Models for Documents Ranking with a Tournament-Inspired Strategy

Anonymous ACL submission

Abstract

Large Language Models (LLMs) are increasingly employed in zero-shot documents ranking, yielding commendable results. However, several significant challenges still persist in LLMs for ranking: (1) LLMs are constrained by limited input length, precluding them from processing a large number of documents simultaneously; (2) The output document sequence is influenced by the input order of documents, resulting in inconsistent ranking outcomes; (3) Achieving a balance between cost and ranking performance is quite challenging. To tackle these issues, we introduce a novel documents ranking method called TourRank, which is inspired by the tournament mechanism. This approach alleviates the impact of LLM's limited input length through intelligent grouping, while the tournament-like points system ensures robust ranking, mitigating the influence of the document input sequence. We test TourRank with different LLMs on the TREC DL datasets and the BEIR benchmark. Experimental results show that TourRank achieves state-of-the-art performance at a reasonable cost.

1 Introduction

Recently, Large Language Models (LLMs) have demonstrated great potential in numerous Natural Language Processing (NLP) tasks, especially under the zero-shot settings. Researchers and practitioners have also tried to leverage LLMs document ranking, a core task in information retrieval, under the zero-shot settings. Most of the existing LLM-based document ranking methods can be divided into three categories: the *Pointwise* approach that prompts LLMs to independetly assess the relevance of each candidate document (Sachan et al., 2022; Liang et al., 2022; Zhuang et al., 2023a; Guo et al., 2024); the *Pairwise* approach that use LLMs to compare each document against all the other documents (Qin et al., 2023); and the *Listwise* approach that instruct LLMs to generate a ranked list of document labels according to their relevance to the query (Sun et al., 2023; Ma et al., 2023; Pradeep et al., 2023a,b; Zhuang et al., 2023c).[1]

While these three approaches lead to different trade-offs between effectiveness and efficiency, the listwise approach, such as RankGPT, is considered as the preferred prompting strategy for the LLM-based zero-shot document ranking task. Unlike the pointwise approach, the listwise approach considers multiple documents simultaneously and thus yields better effectiveness in ranking. Meanwhile, listwise ranking eludes the quadratic growing cost exposed in the comparisons of all document pairs, resulting in improved efficiency.

Although the listwise approaches achieve a good trade-off between effectiveness and efficiency and thus are considered preferred prompting strategies for LLM-based document ranking, they also face certain challenges: (1) The maximum context length of LLMs limits the number of documents that can be compared in a single prompt; (2) The listwise generation process can not run in parallel, which makes it hard to return the final ranking list under a tight time constraint. (3) The ranking results are highly dependent on the initial order of the candidate documents in the input prompt.

To address these challenges, we need to develop a prompting strategy for LLM-based document ranking that can: (Requirement 1) establish a global ranking for about 100 candidate documents through multiple local comparisons of 2 to 10 documents in a single prompt; (Requirement 2) parallelize multiple LLM inferences to minimize the overall ranking process time; and (Requirement 3) effectively utilize the initial order of candidate documents set by the first-stage retrieval model without relying too heavily on it.

[1] See Appendix A.2 for a more detailed literature review of the pointwise, pairwise, and listwise approaches for LLM-based document ranking.

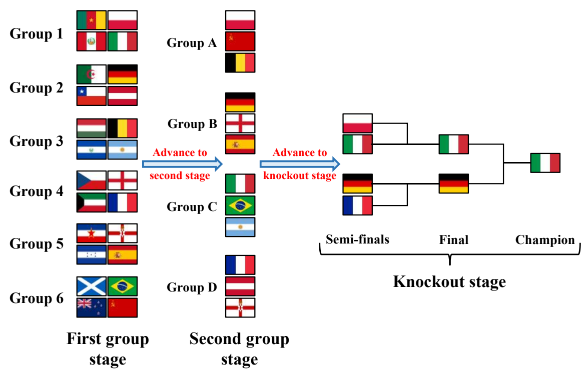


Figure 1: The 1982 FIFA World Cup.
In the first group stage, 24 teams were divided into six groups, and the top 2 out of 4 teams in each group qualified. In the second group stage, 12 teams were divided into 4 groups, and only the top 1 out of 3 teams in each group advanced. In knockout stages, only the winner in each two-team match progressed to the next stage.

Interestingly, we find that using LLMs and prompts to rank documents for a query can be analogous to ranking teams or athletes in a sports tournament, as the design of a sports tournament has similar requirements. A tournament in sports is a structured competition involving multiple teams or individual competitors who compete against each other in a series of matches or games, with the goal of determining a champion or ranking the participants. Figure 1 shows the format and results of an example tournament, the 1982 FIFA World Cup. The tournament consists of two group stages and two knockout stages (i.e., the semi-finals and the final). Analogous to Requirement 1, each group in the group stages and each two-team match in the knockout stages served as a local comparison; the results of these local matches determined which teams could advance to the next stage and their final rankings in the tournament. To expedite the ranking process, the World Cup organized multiple parallel matches across different groups. This parallelization allowed the tournament to progress efficiently and fit into a tight 4-week schedule, which meets Requirement 2. Regarding Requirement 3, the initial groupings were based on seeding and previous performance, providing an initial order of teams. However, the tournament did not solely rely on these seedings; each team's performance in the group stage and subsequent rounds determined their advancement and final rankings.

Therefore, inspired by the tournament mechanism, we propose a new zero-shot document ranking method called **TourRank**, which can fulfill the three requirements and mitigate the challenges in existing methods. In TourRank, we regard each candidate document as a participant in a multi-stage tournament. In each stage, we group the candidate documents and prompt the LLM to select the most relevant documents in each group to advance to the next stage. The LLM inferences across different groups in a single stage can be parallelized. We also design a grouping strategy, similar to the seeding strategy in sports tournaments, to make use of the initial document order provided by the first-stage retrieval model in ranking. In addition, to further improve the effectiveness and robustness, we design a point system to assign different points to each candidate document based on its ranking in each round tournament and perform multiple rounds of tournament. In this way, the ranking list can be obtained based on the final accumulated points in descending order.

To prove the effectiveness of our approach, We test TourRank and baselines on the TREC DL 19 (Craswell et al., 2020), TREC DL 20 datasets (Craswell et al., 2021), and 8 datasets from BEIR benchmark (Thakur et al., 2021).

To conclude, our contributions can be summarized as follows:

- We introduce TourRank, a novel zero-shot documents ranking method based on LLMs. By conducting multiple rounds of tournament, TourRank outperforms existing prompting strategies for zero-shot documents ranking.
- TourRank effectively mitigates the shortcomings of current methods, particularly their sensitivity to the initial candidate documents.
- TourRank strikes a commendable balance between inference cost and effectiveness, further solidifying its advantages.
- Our experimental results confirm that TourRank also achieves SOTA on the open-source models Mistral-7B (Jiang et al., 2023) and Llama-3-8B (MetaAI, 2024).

2 Related Works

Neural Network Approaches Recent advancements in document ranking have been achieved with the help of pre-trained language models like BERT (Devlin et al., 2018) and T5 (Raffel et al., 2020). Notably, Nogueira and Cho (2019) develop a multi-stage text ranking system using BERT, while Nogueira et al. (2020) and Zhuang et al. (2023b) employ T5 for document ranking.

LLMs Approaches Recent studies have utilized LLMs for ranking tasks, employing pointwise, pairwise, and listwise approaches. Pointwise methods, such as Query Generation (QG) (Sachan et al., 2022) and Binary Relevance Generation (B-RG) (Liang et al., 2022), use LLMs to compute the probability or likelihood of query-passage pairs. Pairwise approaches, such as Pairwise Ranking Prompting (PRP) (Qin et al., 2023), leverage LLMs to conduct pairwise comparisons and ranking of retrieved documents. RankGPT (Sun et al., 2023) is a listwise method that adopts a sliding window strategy for document ranking. There are also other listwise methods, like RankVicuna (Pradeep et al., 2023a) and RankZephyr (Pradeep et al., 2023b), which employ instruction-tuning for documents ranking. Setwise prompting (Zhuang et al., 2023c) enhances efficiency by reducing model inferences and prompt token consumption.

More introduction of existing works can be seen in the Appendix A.

3 Method: TourRank

In this section, we introduce our novel zero-shot ranking approach called TourRank, which is inspired by the tournament mechanism and includes multiple parallel tournaments. Similar to how players are ranked based on the accumulated points of multiple tournaments in descending order in a season, TourRank gets the ranking order of candidate documents based on the accumulated points of multi-round tournaments in descending order.

Next, we first delineate how a basic tournament works in TourRank. Then, we explain how to get the accumulated points of the candidate documents, which are subsequently utilized for document ranking. Lastly, we propose a specific grouping method to circumvent the constraints on the input length of LLMs and make full use of the initial ranking order.

3.1 A Basic Tournament

In one tournament of TourRank, we select N_K documents from N_1 candidates that are most relevant to the query in a stage-by-stage manner and each document gets a corresponding point after a whole tournament. As shown in Figure 2 (a), we choose the documents by stagewise selection ($N_1 \rightarrow N_2 \rightarrow \cdots \rightarrow N_{K-1} \rightarrow N_K$). In the k-th selection stage ($k \in \{1, 2, \cdots K-1\}$), the most top-$N_{k+1}$ relevant documents to the given query are selected from N_k documents to next selection stage. After the k-th selection stage, the points of the N_{k+1} selected documents are added by 1 to k. And the points of $N_k - N_{k+1}$ documents that are not selected are still $k-1$. In this way, after a full round of tournament, all candidate documents can get the corresponding points P_{T_r} which is expressed as Table 1. In our experiments, the number of candidate documents is 100, and the specific points of all 100 documents after one tournament are shown in Table 9 in Appendix G.

Number of Docs	Points of Docs
N_K	$K-1$
$N_{K-1} - N_K$	$K-2$
$\cdots$	$\cdots$
$N_k - N_{k+1}$	$k-1$
$\cdots$	$\cdots$
$N_1 - N_2$	0

Table 1: The points of all candidate documents after one tournament. For example, there are $N_k - N_{k+1}$ documents with a score of $k-1$. ($k \in \{1, 2, \cdots K-1\}$)

The parameter r of P_{T_r} represents the r-th round of tournaments. As shown in Figure 3, R rounds of tournaments can be performed in parallel, so we have $r \in \{1, 2, \cdots, R\}$.

3.2 Getting The Accumulated Points

Since the points obtained by one tournament are coarse, multiple tournaments are required to obtain more fine-grained document points. Figure 3 illustrates the process of multiple tournaments, where we can see that points of candidate documents $P_{T_r} (r \in \{1, \cdots, R\})$ are obtained after each round of the tournament.

Because there are many factors that affect the output content of LLMs, such as decoding strategy, temperature coefficient, and especially the order of documents input to LLMs, may introduce some bias, so each set of points $(P_{T_1}, \cdots, P_{T_R})$ obtained by R rounds of tournaments are a little bit different. If these points are added up, the bias of each round tournament could be reduced to some extent, and the accumulated points P_T, which is expressed as Equation (1), are more fine-grained and robust. So the final ranking list is obtained according to the accumulated points P_F in descending order. The analysis in Appendix D shows how exactly TourRank-r improves document ranking.

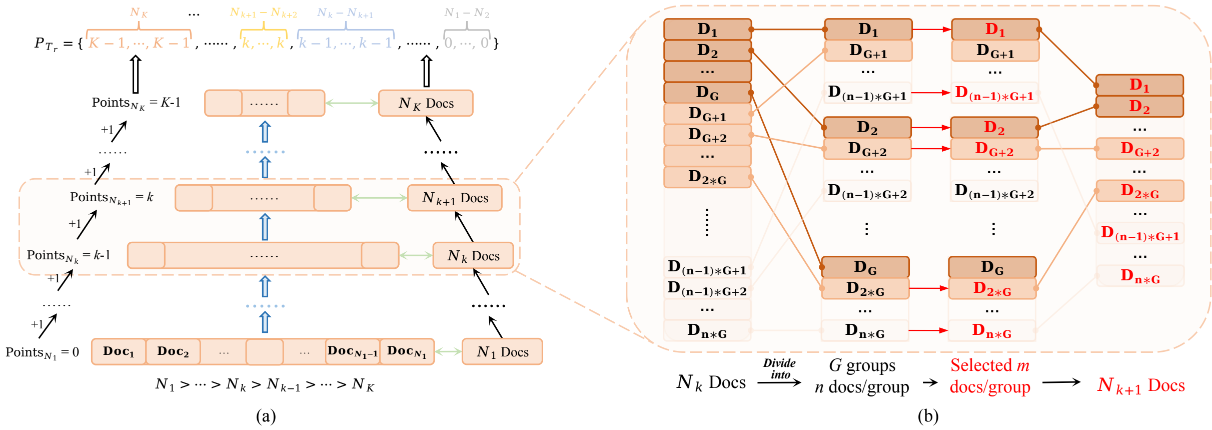


Figure 2: (a) A basic tournament. (b) The grouping strategy in the selection stage of the tournament.

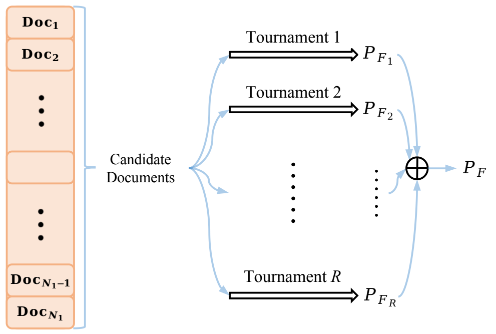


Figure 3: Get the accumulated points of all candidate documents through R tournaments.

$$P_T = \sum_{r=1}^{R} P_{T_r} \quad (1)$$

3.3 The Grouping and Selection Strategy

Considering the limitation of the input length of LLMs, in some stages of TourRank, such as the stage of selecting N_{k+1} documents from N_k candidates in Figure 2 (a), we may not be able to input all N_k documents into LLMs at once. Therefore, we take the approach of assigning N_k candidate documents to several groups and then inputting documents of each group into LLMs separately and simultaneously.

As shown in Figure 2 (b), the N_k documents are divided into G groups, each of which contains n documents. Here the relative order of N_k initial documents is given by the retrieval model, such as BM25 (Robertson et al., 2009), etc. When grouping in a sports tournament, the seeded players and the weaker players are evenly assigned into different groups to ensure the fairness of the competition. Similarly, we used a similar strategy to group the documents by evenly distributing the documents in the initial order into different groups as shown in Figure 2 (b). In this way, there will be some difference in the relevance of the documents within a group, making it easier for LLMs to select the more relevant documents.

Additionally, Liu et al. (2024) find that current language models do not robustly access and use information in long input contexts because of the position bias. In order to eliminate the bias of LLMs on document input order and achieve a robust ranking, the order of documents in each group will be shuffled before entering LLMs and the multiple tournaments will be performed as shown in Figure 3.

After grouping the documents, we select the most relevant m documents from the n ($m < n$) documents in each group. In Figure 2 (b), we mark the selected m documents in red in each group, and these documents advance to the next stage.

Eventually, through the k-th selection stage of the tournament, N_{k+1} more relevant documents are selected from the N_k documents to advance to the next stage. Benefiting from this smart grouping stage and multi-round tournaments mechanism, we solve the problem of limited input length of LLMs while achieving a more robust selection.

3.4 The Overall of TourRank

As the Pseudo-code of TourRank shown in **Algorithm 1**, we perform R parallel tournaments as the process in Figure 3 for the given query q and the candidate documents list D. In r-th round tournament, we first initialize the points of all N_1 candidate documents, that is $P_{T_r} = 0$ for N_1 documents. Then, we select and increase the points of the documents in a stage-by-stage way in which $K-1$ times selection stages are executed serially, and this is corresponds to Figure 2 (a). In k-th selection stage, we adopt a suitable grouping approach (Figure 2 (b)) to get the N_{k+1} documents which can advance to the next selection stage, while adding points to the selected N_{k+1} documents. After R rounds tournament, the points $P_{T_r}, r \in \{1, \cdots, R\}$ can be obtained. We can calculate the final points P_T according to Equation (1). Finally, we re-rank the candidate documents list according to the final points P_T in descending order.

Algorithm 1 The Pseudo-code of TourRank

Input: The query q and candidate documents list D
Perform R tournaments ***in parallel****, $r \in \{1, \cdots, R\}$:*
  Initialize the points as $\boldsymbol{P_{T_r}} = \mathbf{0}$ for N_1 documents.
  Perform k-th selection stages, for k in range$(1, K)$:
    Assign N_k documents to G groups and each group has n documents.
    Select m documents that are more relevant to the query q from n in each group ***in parallel****.*
    Get the selected N_{k+1} documents to advance to next stage.
    The points $\boldsymbol{P_{T_r}}$ of the selected N_{k+1} documents add 1.
  Get a set of points $\boldsymbol{P_{T_r}}$ for all N_1 documents.
After R times parallel tournaments, the final points $\boldsymbol{P_T}$ can be obtained according to Equation (1).
Rank the candidate documents D according to $\boldsymbol{P_T}$ in descending order.
Output: A ranked list of candidate documents D

The specific hyperparameters of TourRank can be seen in Table 8 in the Appendix G.

4 Experiments

Our experiments mainly focus on the following research questions:

- **RQ.1**: Whether TourRank outperforms existing ranking methods based on LLMs.
- **RQ.2**: Is TourRank sensitive to the candidate documents retrieved by different models and the initial order of documents, that is, does it have a robust ranking?
- **RQ.3**: Is TourRank easy to achieve a trade-off between effectiveness and resource consumption?
- **RQ.4**: Can TourRank achieve the best performance based on different LLMs (open-source and close-source)?

4.1 Experimental Settings

Datasets We conduct experiments to answer the above research questions on TREC DL datasets (Craswell et al., 2020, 2021) and BEIR benchmark (Thakur et al., 2021). **TREC** is a widely used benchmark in IR research. We use the test sets of TREC DL 19 and TREC DL 20, which contain 43 and 54 queries. **BEIR** is a heterogeneous zero-shot evaluation benchmark. Following Sun et al. (2023), we select 8 datasets for evaluation, including Covid, Touche, DBPedia, SciFact, Signal, News, Robust04, and NFCorpus.

Metrics In the next evaluations, we re-rank the top-100 documents retrieved by the first-stage retrieval model. If not specified, we use BM25 as the default retrieval model and PySerini for implementation.[2] We use NDCG@{5, 10, 20} as evaluation metrics.

Baselines We compare TourRank with several state-of-the-art baselines in documents ranking, including the supervised methods monoBERT (Nogueira and Cho, 2019) and monoT5 (Nogueira et al., 2020), and zero-shot methods based on LLMs: two *pointwise* methods, DIRECT(0, 10) (Guo et al., 2024), Binary Relevance Generation (B-RG) (Liang et al., 2022), one *pairwise* method PRP (Qin et al., 2023), and two *listwise* methods, Setwise (Zhuang et al., 2023c) and RankGPT (Sun et al., 2023). The detailed introductions of these baselines are in the Appendix B.

4.2 Experimental Results

Results on TREC DL datasets Table 2 shows the performance of different methods on TREC DL datasets. We compare NDCG@{5, 10, 20}, and the best top-4 results of zero-shot LLM methods are shaded. We reproduce all zero-shot LLM methods with gpt-3.5-turbo API. From the results, we can make the following findings:

(1) Our TourRank-10 outperforms all zero-shot ranking baselines. It is worth noting that after two tournaments (TourRank-2) the performance is much better than one tournament (TourRank-1), and TourRank-2 can significantly outperform RankGPT. This indicates that TourRank can achieve good results with fewer tournaments.

(2) Generally, the two pointwise methods tend to underperform in comparison to the pairwise method, PRP-Allpair. However, the PRP-Allpair falls short when compared to the listwise methods, Setwise.bubblesort and our TourRank-10. This indicates that listwise, which considers multiple documents simultaneously, is generally more effective among LLM-based zero-shot ranking methods.

(3) PRP-Allpair achieves about the same per-

[2] https://github.com/castorini/pyserini

Methods	TREC DL 19			TREC DL 20		
	NDCG@5	NDCG@10	NDCG@20	NDCG@5	NDCG@10	NDCG@20
BM25	52.78	50.58	49.14	50.67	47.96	47.21
Supervised Methods						
monoBERT (340M)	73.25	70.50	-	70.74	67.28	-
monoT5 (220M)	**73.77**	71.48	-	69.40	66.99	-
monoT5 (3B)	73.74	**71.83**	-	**72.32**	**68.89**	-
Zero-Shot LLM Methods						
DIRECT(0, 10)	54.22	54.59	54.15	55.17	55.35	54.73
B-RG	63.33	62.51	60.00	65.04	63.37	60.47
PRP-Allpair	70.43	68.18	64.61	69.75	66.40	64.03
Setwise.bubblesort	73.58	71.16	67.89	71.66	69.04	65.52
RankGPT	72.05	68.19	62.21	67.25	63.60	59.12
TourRank-1	70.95	66.23	62.49	66.65	63.74	60.59
TourRank-2	72.24	69.54	65.03	67.65	65.20	62.78
TourRank-10	**73.83**	**71.63**	**68.37**	**72.49**	**69.56**	**66.13**

Table 2: Performance comparison of different methods on TREC datasets. We reproduce all the zero-shot LLM methods with gpt-3.5-turbo API. The best-performing algorithms for supervised methods and zero-shot LLM methods are bolded, respectively. The best top-4 results of zero-shot LLM methods are shaded in each metric. TourRank-r represents that we perform r times tournaments.

formance as RankGPT on TREC DL 19, and outperforms RankGPT on TREC DL 20. Setwise.bubblesort outperforms RankGPT on both datasets and is second only to TourRank-10. However, PRP-Allpair and Setwise.bubblesort achieve relatively good results at the cost of much higher complexity and resource consumption than RankGPT and TourRank. We discuss the effectiveness and cost of them in Section 4.5.

(4) TourRank-10 achieves comparable results to the best supervised methods on TREC DL 19, and on TREC DL 20 TourRank-10 outperforms the best performing supervised method monoT5 (3B). It can be seen that TourRank is the only zero-shot method based on gpt-3.5-turbo API that can do this. We also perform TourRank with other close and open source LLMs in Section 4.6.

Results on BEIR benchmark Table 3 shows the NDCG@10 of different methods on 8 tasks of BEIR benchmark. The following are some valuable discussions:

(1) TourRank-10 achieves the best performance in 6 out of 8 tasks and the best average NDCG@10 across 8 tasks among zero-shot LLM methods.

(2) The average of TourRank-2 (49.46) outperforms RankGPT (49.37) in terms of NDCG@10 in Table 3, which, together with the better performance of TourRank-2 over RankGPT on TREC DL datasets shown in Table 2, prove that our TourRank algorithm can achieve good results with only a few times tournaments.

(3) Note that on the Touche task and Signal task, all supervised methods and zero-shot methods in Table 3 perform even worse than BM25. The NDCG@10 of all methods on these two tasks is low, only about 0.3. According to Thakur et al. (2024), the poor performance of neural retrieval models is mainly due to the large number of short texts and unlabeled texts in the Touche dataset.

The results on TREC datasets and BEIR benchmark jointly answer the **RQ.1**.

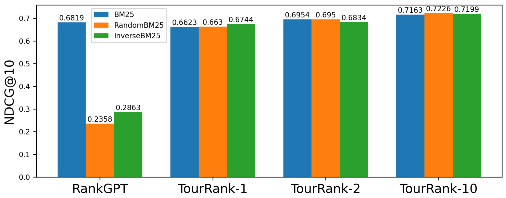
4.3 Sensitivity Analysis to Initial Ranking

We compare 3 different initial ranking: 1) **BM25**: Get top-100 documents by BM25; 2) **RandomBM25**: Shuffle the order of BM25; 3) **InverseBM25**: Reverse the order of BM25. Figure 4 shows the results of RankGPT and our TourRank based on 3 initial rankings and all these experiments are based on gpt-3.5-turbo API.

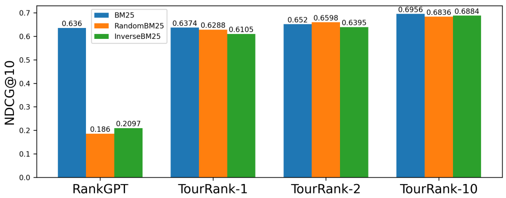
From Figure 4, we can see that RankGPT is very sensitive to the initial permutation of documents list. When the initial permutation is shuffled or reversed, the performance of RankGPT becomes much worse. This is caused by the ranking mechanism of RankGPT, which adjusts the overall permutation of documents list through the sliding window strategy. Sliding the window from bottom to top makes it easier for documents that are originally near the top to be ranked at top positions in the final permutation. Whereas documents that are at the bottom of the initial permutation need to be ranked at the top of every comparison in corresponding sliding window in order to be ranked at the top of

Methods	Covid	NFCorpus	Touche	DBPedia	SciFact	Signal	News	Robust04	Average
BM25	59.47	30.75	**44.22**	31.80	67.89	**33.05**	39.52	40.70	43.42
Supervised Methods									
monoBERT (340M)	70.01	36.88	31.75	41.87	71.36	31.44	44.62	49.35	47.16
monoT5 (220M)	78.34	37.38	30.82	42.42	73.40	31.67	46.83	51.72	49.07
monoT5 (3B)	**80.71**	**38.97**	32.41	**44.45**	**76.57**	32.55	**48.49**	**56.71**	**51.36**
Zero-Shot LLM Methods									
RankGPT	76.67	35.62	36.18	44.47	70.43	32.12	48.85	50.62	49.37
TourRank-1	77.17	36.35	29.38	40.62	69.27	29.79	46.41	52.70	47.71
TourRank-2	79.85	36.95	30.58	41.95	71.91	31.02	48.13	55.27	49.46
TourRank-10	**82.59**	**37.99**	29.98	**44.64**	**72.17**	30.83	**51.46**	**57.87**	**50.94**

Table 3: Performance (NDCG@10) comparison of different methods on BEIR benchmark. The best-performing algorithms for supervised methods and zero-shot LLM methods are bolded. TourRank-r represents that we perform r times tournaments.



(a) TREC DL 19



(b) TREC DL 20

Figure 4: The sensitivity analysis to initial ranking of TourRank and RankGPT on TREC DL 19 and TREC DL 20.

the final permutation, otherwise they are left at the bottom or middle of the whole documents list. So, this is the reason why RankGPT is very sensitive to the initial ranking.

However, our TourRank is quite robust to different initial orderings, as shown by the fact that shuffling and reversing the initial order has almost no effect on TourRank-r. The robustness of TourRank to the initial ranking benefits from the tournament mechanism presented in Figure 2. Each tournament is a selection over all candidate documents, not just a fine-tuning of the initial ranking like RankGPT.

4.4 Analysis to Different Retrieval Models

In addition to BM25, we also obtain top-100 documents based on two more powerful retrieval models, including a dense retriever model Contriever (Izacard et al., 2021) and a neural sparse retrieval model SPLADE++ ED (Formal et al., 2022), as the first-stage retrieval model. Then, we perform TourRank and RankGPT to re-rank the top-100 candidate documents retrieved by different retrieval models based on gpt-3.5-turbo API. The results in Table 4 show that TourRank-10 achieves SOTA ranking performance based on 3 kinds of different top-100 initial candidate documents. And TourRank-2 can also outperform RankGPT in general.

Methods	Top-100	TREC DL 19	TREC DL 20
BM25	-	50.58	47.96
RankGPT		68.19	63.60
TourRank-2	BM25	69.54	65.20
TourRank-10		**71.63**	**69.56**
Contriever	-	62.02	63.42
RankGPT		69.70	68.47
TourRank-2	Contriever	69.12	71.89
TourRank-10		**70.77**	**73.19**
SPLADE++ ED	-	73.08	71.97
RankGPT		74.56	70.75
TourRank-2	SPLADE++ ED	74.86	74.11
TourRank-10		**75.35**	**77.09**

Table 4: NDCG@10 of TourRank and RankGPT based on different retrieval models. Here we use gpt-3.5-turbo API for TourRank and RankGPT.

The results in Table 4 and the analysis in Section 4.3 jointly answer the **RQ.2**, that is, TourRank has the ability of robust ranking.

4.5 The Trade-Off between Effectiveness and Resource Consumption

Table 5 shows the approximation of the theoretical lowest time complexity of different methods and the number of documents LLMs need to receive. All the contents of Table 5 are based on the recommended parameters. More detailed discussions on precise time complexity and number of input documents are in Table 7 in the Appendix E. From Table 5, we can see that:

(1) PointWise has the lowest time complexity and the lowest number of documents received by LLMs, but the experimental results of DIRECT(0, 10) and B-RG in Table 2 show that PointWise exhibits poor performance.

(2) Although the pairwise method PRP-Allpair performs well in the experiments on TREC datasets, the number of input documents required by PRP-AllPair is $N^2 - N$, which will greatly increase the cost of ranking.

(3) Setwise.bubblesort performs very well on TREC DL datasets in Table 2 and is second only to TourRank-10. However, the multiple steps of Setwise have dependencies and cannot be run in parallel, resulting in the time complexity of Setwise being extremely high and unacceptable.

(4) Two listwise methods RankGPT and our TourRank take into account both the time complexity and the number of documents inputted to LLMs. The experimental results in Table 2 and 3 show that TourRank-2 can outperform RankGPT. From Table 5, we can see that TourRank-2 ($r = 2$) achieves this goal with about twice as many documents to LLMs as RankGPT but with lower time complexity. We also compare TourRank with running RankGPT multiple iterations in serial (Appendix F), and TourRank demonstrates better performance and lower consumption.

Methods	Time Complexity	No. of Docs to LLMs
PointWise	$O(1)$	N
PRP-Allpair	$O(1)$	$N^2 - N$
Setwise.bubblesort	$\approx O(\frac{1}{2}k * N)$	$\approx \frac{3}{2}k * N$
RankGPT	$\approx O(\frac{1}{10} * N)$	$\approx 2 * N$
TourRank-r	$O(K-1)$	$\approx 2r * N$

Table 5: A approximation of the theoretical lowest time complexity of various methods and the number of documents which are inputted to LLMs for each method. N is the number of candidate documents. Setwise rank the top-k ($k < N$) documents through bubblesort. $K - 1$ is the times of the selection stages in a tournament (Figure 2 (a)) and r is the times of tournaments in TourRank-r. (Note: The approximate contents in this table are based on the recommended parameters.)

The above experimental results and the analysis of the trade-offs between effectiveness and efficiency jointly answer the **RQ.3**.

4.6 TourRank Based on Other LLMs

We also perform TourRank based on the open-source models Mistral-7B (Jiang et al., 2023) and Llama-3-8B (MetaAI, 2024), and gpt-4-turbo API.

Table 6 shows the performance of TourRank with different LLMs. The top-100 candidate documents are retrieved by BM25. The results show that RankGPT performs far worse based on the open-source LLMs than based on OpenAI's APIs. However, the performances of our TourRank based on open-source LLMs are still good. Especially on TREC DL 19 dataset, the performance of TourRank-10 based on Llama 3 8B achieves 72.76, which is higher than the performance of TourRank-10 based on gpt-3.5-turbo (71.63). In addition, TourRank-5 with gpt-4-turbo outperforms all methods based on gpt-3.5-turbo in Table 2, which indicates that TourRank can achieve higher performance with fewer tournaments times r based on a stronger model.

Methods	LLMs	TREC DL 19	TREC DL 20
BM25	-	50.58	47.96
RankGPT	Mistral-7B	61.90	58.54
TourRank-2	Mistral-7B	65.47	61.52
TourRank-10	Mistral-7B	**68.93**	**65.53**
RankGPT	Llama-3-8B	59.48	54.47
TourRank-2	Llama-3-8B	68.95	65.62
TourRank-10	Llama-3-8B	**72.76**	**68.05**
RankGPT	gpt-4-turbo	72.67	69.48
TourRank-1	gpt-4-turbo	72.46	67.38
TourRank-5	gpt-4-turbo	**74.13**	**69.79**

Table 6: NDCG@10 of TourRank and RankGPT based on open-source LLMs, Mistral 7B and Llama 3 8B, and gpt-4-turbo API.

These experiments shows that TourRank can achieve good performance not only based on OpenAI's API, but also based on open source LLMs, answering the **RQ.4**.

5 Conclusions

We introduce TourRank, a novel zero-shot documents ranking method inspired by the tournament mechanism. TourRank addresses challenges in large language models for ranking, such as input length limitations and sensitivity to input order.

Our experiments show that TourRank outperforms existing LLM-based zero-shot ranking methods, balances effectiveness and cost. This demonstrates that TourRank is a promising approach for future research in zero-shot documents ranking.

Limitations

The performance of TourRank is inherently dependent on the capabilities of the underlying LLMs. If the LLMs can't follow the instructions well, it will be difficult to achieve good results.

Although multiple tournaments of TourRank can be performed in parallel in a multi-process manner, for example, based on the API of OpenAI, it is difficult to run the open-source models in a multi-process manner under the environment of limited computing resources.

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*.

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

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

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*, pages 2353–2359.

Fang Guo, Wenyu Li, Honglei Zhuang, Yun Luo, Yafu Li, Le Yan, and Yue Zhang. 2024. Generating diverse criteria on-the-fly to improve point-wise llm rankers. *arXiv preprint arXiv:2404.11960*.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*.

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*.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. 2022. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

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*.

MetaAI. 2024. Meta llama 3.

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

Rodrigo Nogueira, Zhiying Jiang, and Jimmy Lin. 2020. Document ranking with a pretrained sequence-to-sequence model. *arXiv preprint arXiv:2003.06713*.

OpenAI. Openai. `https://www.openai.com/`. Optional additional information.

Ronak Pradeep, Rodrigo Nogueira, and Jimmy Lin. 2021. The expando-mono-duo design pattern for text ranking with pretrained sequence-to-sequence models. *arXiv preprint arXiv:2101.05667*.

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*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

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

Devendra Singh Sachan, Mike Lewis, Mandar Joshi, Armen Aghajanyan, Wen-tau Yih, Joelle Pineau, and Luke Zettlemoyer. 2022. Improving passage retrieval with zero-shot question generation. *arXiv preprint arXiv:2204.07496*.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Pengjie Ren, Dawei Yin, and Zhaochun Ren. 2023. Is chatgpt good at search? investigating large language models as re-ranking agent. *arXiv preprint arXiv:2304.09542*.

Nandan Thakur, Luiz Bonifacio, Maik Fröbe, Alexander Bondarenko, Ehsan Kamalloo, Martin Potthast, Matthias Hagen, and Jimmy Lin. 2024. Systematic evaluation of neural retrieval models on the touché 2020 argument retrieval subset of beir. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

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 preprint arXiv:2104.08663*.

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, Honglei Zhuang, Bevan Koopman, and Guido Zuccon. 2023c. A setwise approach for effective and highly efficient zero-shot ranking with large language models. *arXiv preprint arXiv:2310.09497*.

Appendix

A More Related Works

A.1 Neural Network Approaches

Documents ranking has made significant progress, with the help of pre-trained language models, such as BERT (Devlin et al., 2018) and T5 (Raffel et al., 2020). Nogueira and Cho (2019) present a multi-stage text ranking system using BERT, introducing monoBERT and duoBERT models that offer a balance between quality and latency, achieving state-of-the-art results on MS MARCO and TREC CAR datasets. Nogueira et al. (2020) introduce a new method for document ranking using a pre-trained sequence-to-sequence model, T5, which outperforms classification-based models, especially in data-poor scenarios, and demonstrates the model's ability to leverage latent knowledge from pretraining for improved performance. Zhuang et al. (2023b) introduce RankT5, a method for fine-tuning the T5 model for text ranking using ranking losses, which shows significant performance improvements over models fine-tuned with classification losses and demonstrates better zero-shot ranking performance on out-of-domain data.

A.2 LLMs Approaches

Pointwise Approaches There are several works that employ various zero-shot pointwise rankers. Query Generation (QG) (Sachan et al., 2022) involves rescoring retrieved passages by leveraging a zero-shot question generation model. The model uses a pre-trained language model to compute the probability of the input question, conditioned on a retrieved passage. Binary Relevance Generation (B-RG) (Liang et al., 2022) proposes to utilize LLMs to make predictions on a query-passage pair, utilizing the likelihood of "Yes/No" responses for the computation of ranking scores. The Rating Scale $0-k$ Relevance Generation (RS-RG) (Zhuang et al., 2023a) incorporates fine-grained relevance labels into the prompts for LLM rankers to better differentiate documents of varying relevance levels to the query, thereby achieving more accurate rankings. Guo et al. (2024) propose a multi-perspective evaluation criteria-based ranking model to overcome the deficiencies of LLM rankers in standardized comparison and handling complex passages, thereby significantly enhancing the pointwise ranking performance. Guo et al. (2024) have also considered the Rating Scale $0-k$ Directly Score (DIRECT(0, k)) method. This approach prompts the LLM to directly generate the relevance score for each query-passage pair.

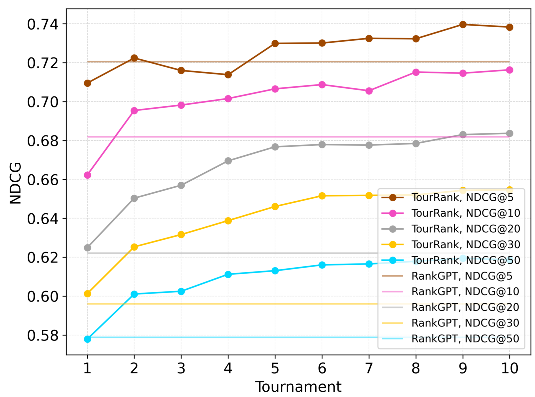
Pairwise Approaches Pradeep et al. (2021) design a pairwise component to enhance the early precision performance of the text ranking system by employing a pre-trained sequence-to-sequence model (such as T5 (Raffel et al., 2020)) to conduct pairwise comparisons and reranking of retrieved document pairs. Qin et al. (2023) introduce a method called Pairwise Ranking Prompting (PRP), which effectively enables LLMs to perform text ranking tasks by simplifying the prompt design and achieving competitive performance across multiple benchmark datasets.

Listwise Approaches LRL (Ma et al., 2023) enhances text retrieval reranking by employing a large language model as a zero-shot listwise reranker, utilizing a simple instruction template and a sliding window strategy to process multi-document information. Similarly, Sun et al. (2023) introduce a novel instructional permutation generation approach called RankGPT, utilizing a sliding window strategy to effectively enable LLMs (such as ChatGPT (OpenAI) and GPT-4 (Achiam et al., 2023)) to be used for relevance ranking tasks in information retrieval, achieving competitive and even superior results on popular IR benchmarks. In addition, both RankVicuna (Pradeep et al., 2023a) and RankZephyr (Pradeep et al., 2023b) utilize open-source LLMs and employ instruction fine-tuning to achieve zero-shot listwise document reranking, thereby enhancing the ranking performance of smaller LLMs. Zhuang et al. (2023c) propose a novel Setwise prompting approach to enhance the efficiency and effectiveness of LLMs in zero-shot document ranking tasks, by reducing the number of model inferences and prompt token consumption, which significantly improves computational efficiency while maintaining high ranking performance.

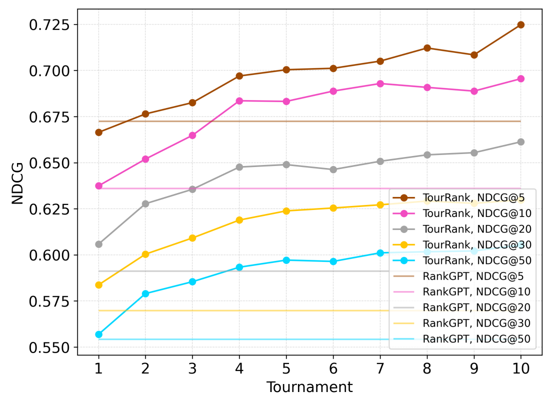
B Introduction of Baselines

B.1 Supervised Methods

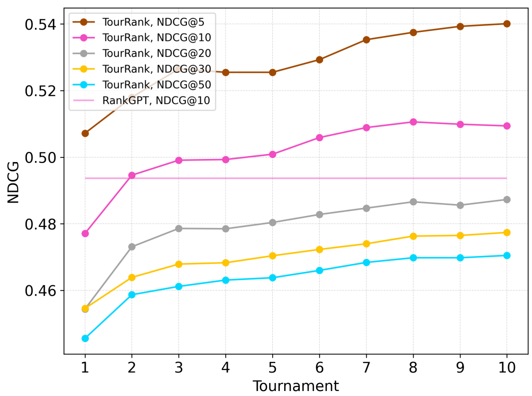
- **monoBERT** (Nogueira and Cho, 2019): A ranking method with a cross-encoder architecture based on BERT-large, trained on MS MARCO.
- **monoT5** (Nogueira et al., 2020): A ranking method that calculates the scores using T5 model.



(a) TREC DL 19



(b) TREC DL 20



(c) Average of 8 tasks on BEIR

Figure 5: The performance of TourRank with different times of tournaments. The abscissa is the times of tournaments, and the ordinate is NDCG@{5, 10, 20, 30, 50}. All the results are based on gpt-3.5-turbo API.

B.2 Zero-Shot Methods

- **DIRECT(0, 10)** (Guo et al., 2024): A pointwise method which gives the relevance scores ranging from 0 to 10 to each query-document pair in text format using LLMs. Then, rank the documents according to these scores in descending order.
- **Binary Relevance Generation (B-RG)** (Liang et al., 2022): A pointwise method which ranks the candidate documents according to the likelihood of "Yes or No" on a query-document pair.
- **PRP** (Qin et al., 2023): A pairwise method that reduces the burden on LLMs by using a technique called Pairwise Ranking Prompting.
- **Setwise** (Zhuang et al., 2023c): A listwise method that improves the efficiency of LLM-based zero-shot ranking. The authors introduce two setwise methods, setwise.bubblesort and setwise.heapsort. Because the former has better performances on experiments, we reproduce setwise.bubblesort in our experiments (Table 2). And we set $c = 3$ which is the best value for setwise.bubblesort. c is the number of compared documents in a prompt.
- **RankGPT** (Sun et al., 2023): A listwise method that uses a sliding window strategy to achieve listwise ranking based on LLMs. In the experiments, we observed some instability in the performance of RankGPT. So the values in Table 2 and Figure 4 are the average by running RankGPT 3 times.

C The Performance of TourRank-r

Figure 5 shows the trend of NDCG@{5, 10, 20, 30, 50} with the increase of the number of tournaments for TourRank on TREC datasets and BEIR benchmark. We can see that after the first two tournaments, TourRank-2 achieves relatively good results on all datasets, outperforming RankGPT on all corresponding metrics shown. Even in TourRank-10, the metrics still have the potential to continue to increase.

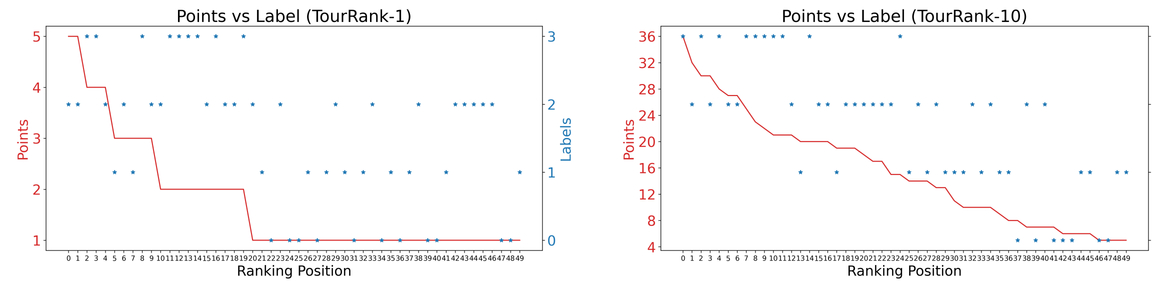


Figure 6: The relationship between the accumulated points P_T and the corresponding labels for TourRank-1 and TourRank-10. The query of this case is "how long is life cycle of flea" which is one of the queries in the TREC DL 19.

Since the number of tokens consumed by TourRank scales linearly with the number of tournaments, we can control the number of consumed tokens by controlling the number of tournaments. Thus, the balance between effectiveness and token consumption can be achieved.

D Case Study: How Does TourRank Improve the Performance of Documents Ranking?

In Figure 6, the horizontal coordinate represents the ranking position of top-50 documents, the red lines represent the accumulated points P_T of TourRank-1 and TourRank-10 respectively, and the blue star points represent the corresponding real labels (integers from 0 to 3).

It can be seen that the P_T of TourRank-1 is coarse, and the labels for the top-50 ranked documents are also relatively scattered. However, the accumulated points P_T of TourRank-10 become much more fine-grained after 10 tournaments, and the labels corresponding to top-50 documents are relatively concentrated. After testing, the NDCG@{10, 50} of the case query have increased from {0.7078, 0.8186} to {0.8715, 0.911}.

Therefore, as the times of tournaments increases, the accumulated points P_T become more fine-grained. This is how exactly TourRank improves the document ranking performance.

E The Discussions on Time Complexity and Number of Documents Inputted to LLMs

Table 7 is a more precise version of Table 5 which shows the theoretical lowest time complexity of various methods and the number of documents which are inputted to LLMs for each method. Then, we analysis the content in Table 7.

E.1 Time Complexity

PointWise and Pairwise Since PointWise scoring a single document and PRP-Allpair comparing a pair of documents can be performed in parallel, the theoretical lowest time complexity is $O(1)$. However, since pairwise methods need to compare about $O(N^2)$ pairs of documents, the theoretical minimum time complexity $O(1)$ is difficult to implement.

Setwise.bubblesort According to (Zhuang et al., 2023c), the time complexity of Setwise.bubblesort is $O(k * \frac{N}{c-1})$. Setwise rank the top-k ($k < N$) documents through bubblesort, and c is the documents compared in a prompt of Setwise. Considering that Setwise can achieve the best performance with $c = 3$, the time complexity is:

$$O(k * \frac{N}{c-1}) \approx O(\frac{1}{2}k * N) \quad (883)$$

RankGPT RankGPT uses sliding window strategy, so its time complexity is $O(\frac{N-\omega}{s})$. The best window size is $\omega = 20$ and the best step size is $s = 10$ in RankGPT. Based on the optimal parameters ($\omega = 20$ and $s = 10$) and considering that ω is often much smaller than N, the best time complexity of RankGPT is:

$$O(\frac{N-\omega}{s}) = O(\frac{N-20}{10}) \approx O(\frac{1}{10} * N) \quad (891)$$

TourRank-r One tournament includes $K - 1$ times selection stages shown in Figure 2, so the time complexity of one tournament is $O(K - 1)$. Because r rounds tournaments can be performed in parallel, the time complexity of TourRank-r is also $O(K - 1)$.

Methods	Time Complexity	No. of Docs to LLMs
PointWise	$O(1)$	N
PRP-Allpair	$O(1)$	$N^2 - N$
Setwise.bubblesort	$O(k * \frac{N}{c-1}) \approx O(\frac{1}{2}k * N)$	$k * \frac{N}{c-1} * c \approx \frac{3}{2}k * N$
RankGPT	$O(\frac{N-\omega}{s}) \approx O(\frac{1}{10} * N)$	$\omega * \frac{N-\omega}{s} \approx 2 * N$
TourRank-r	$O(K-1)$	$\left(\sum_{k=0}^{K-1} \frac{N}{2^k}\right) * r \approx 2r * N$

Table 7: This Table is a more precise version of Table 5. The theoretical lowest time complexity of various methods and the number of documents which are inputted to LLMs for each method. N is the number of candidate documents. Setwise rank the top-k ($k < N$) documents through bubblesort, and $c = 3$ is the documents compared in a prompt of Setwise. $\omega = 20$ is window size and $s = 10$ is step size in RankGPT. $K - 1$ is the times of the selection stages in a tournament (Figure 2 (a)) and r is the times of tournaments in TourRank-r. All the approximate contents in this table are based on the recommended parameters.

E.2 No. of Docs to LLMs

PointWise Since the PointWise method scores each document once, the number of documents inputted to LLMs is N.

Pairwise However, PRP-Allpair needs to form at least $\frac{N*(N-1)}{2}$ pairs for N candidate documents, and since one pair of documents is inputted to LLMs each time, the number of documents it inputs to LLMs is $N^2 - N$.

Setwise.bubblesort The time complexity of Setwise.bubblesort is $O(k*\frac{N}{c-1})$ and $c = 3$ documents is compared in a prompt, so the number of documents inputted to LLMs for Setwise.bubblesort is:

$$k * \frac{N}{c-1} * c \approx \frac{3}{2}k * N$$

RankGPT In RankGPT, we know that ω documents need to be inputted into each window, and a total $\frac{N-\omega}{s}$ intra-window ranking need to be performed, so the number of documents input to LLMs is $\omega * \frac{N-\omega}{s}$. The best window size ω given in RankGPT is 20 and the best step size s is 10. Based on the optimal parameters and considering that ω is often much smaller than N, the number of documents inputted into the LLMs of RankGPT is:

$$\omega * \frac{N-\omega}{s} = 20 * \frac{N-20}{10} \approx 2 * N$$

TourRank-r In TourRank, if close to half of the documents are selected to advance to the next selection stage in a tournament (that is, $m \approx \frac{1}{2}n$), the total number of documents input to LLMs is about:

$$N + \frac{N}{2} + \cdots + \frac{N}{2^{K-2}} = \sum_{k=0}^{K-1} \frac{N}{2^k}$$

$$= N * \frac{1 - \left(\frac{1}{2}\right)^{K-1}}{1 - \frac{1}{2}}$$

$$\approx 2 * N$$

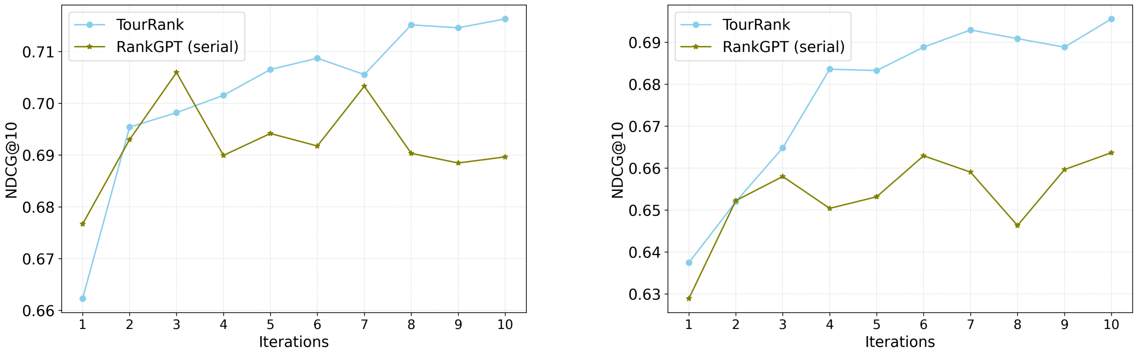
The TourRank-r performs r rounds tournaments, so the number of documents inputted to LLMs of TourRank is about:

$$\left(\sum_{k=0}^{K-1} \frac{N}{2^k}\right) * r \approx 2r * N$$

F Comparison Between Serial RankGPT and Parallel TourRank-r

We also run RankGPT multiple times in seriality called RankGPT (serial), that is, the documents order obtained by this iteration is used as the initial order for the next iteration. Figure 7 shows the comparison of RankGPT (serial) and our TourRank. We can see that on both TREC DL 19 and TREC DL 20 datasets, the NDCG@10 of RankGPT (serial) goes up for the first three iterations, but stops going up after that. This indicates that RankGPT will reach the upper limit after a few serial runs. However, after multiple iterations (or tournaments) of TourRank-r, the NDCG@10 still continues to rise and performs much better than RankGPT (serial).

RankGPT (serial) and TourRank after the same r iterations: (1) The number of documents inputted to LLMs are both about $2r * N$; (2) The time complexity $O(K - 1)$ of TourRank is also significantly



(a) TREC DL 19

(b) TREC DL 20

Figure 7: The comparison of NDCG@10 between running RankGPT multiple times in serial and running TourRank-r in parallel.

less than $O(\frac{r}{10} * N)$ of RankGPT (serial); (3) The performance of TourRank is significantly better than RankGPT (serial). These indicate that TourRank can achieve a better balance between effectiveness and efficiency.

G The Detail Hyperparameters of TourRank

The detail of hyperparameters of TourRank are shown in Table 8.

Table 9 shows the specific points of candidate document after 1 time tournament under the setting of our experiments.

H Prompts

Table 10 shows the prompt used in the grouping and selction stage (Figure 2 (b)) of TourRank.

Parameters	Explanation	Value
R	The rounds of tournament in TourRank.	10
K	One tournament contains $K-1$ times selection stages.	6
N_k	The number of candidate documents in k-th selection stages in a tournament. ($k \in \{1, \cdots, K\}$)	$N_1 = 100$ $N_2 = 50$ $N_3 = 20$ $N_4 = 10$ $N_5 = 5$ $N_6 = 2$
$G/n/m$	G: Divide candidate documents into G groups. n: Each group has n documents. m: Select m documents from each group.	$100 \to 50 : 5/20/10$ $50 \to 20 : 5/10/4$ $20 \to 10 : 1/20/10$ $10 \to 5 : 1/10/5$ $5 \to 2 : 1/5/2$

Table 8: Hyperparameters of TourRank.

Number of Docs	Points of Docs
$N_6 = 2$	5
$N_5 - N_6 = 3$	4
$N_4 - N_5 = 5$	3
$N_3 - N_4 = 10$	2
$N_2 - N_3 = 30$	1
$N_1 - N_2 = 50$	0

Table 9: The specific points of all documents after one tournament in our experimental settings.

system: You are an intelligent assistant that can compare multiple documents based on their relevancy to the given query.

user: I will provide you with the given query and n documents. Consider the content of all the documents comprehensively and select the m documents that are most relevant to the given query: $query$.

assistant: Okay, please provide the documents.

user: Document 1: Doc_1
assistant: Received Document 1.

user: Document 2: Doc_2
assistant: Received Document 2.

(User input more documents to assistant.)

user: The Query is: $query$. Now, you must output the top m documents that are most relevant to the Query using the following format strictly, and nothing else. Don't output any explanation, just the following format:
Document 3, ..., Document 1

Table 10: The prompt of the grouping and selection stage of TourRank.