Active Learners as Efficient PRP Rerankers

Anonymous ACL submission

Abstract

Pairwise Ranking Prompting (PRP) elicits pairwise preference judgments from an LLM, which are then aggregated into a ranking, usually via classical sorting algorithms. However, judgments are noisy, order-sensitive, and sometimes intransitive, so sorting assumptions don't match the setting. Because sorting aims to recover a full permutation, truncating it to meet a call budget does not produce a dependable top-K. We thus reframe PRP reranking as active learning from noisy pairwise comparisons and show active rankers are drop-in replacements that improve NDCG@10 per call in the call-constrained regime. Our noise-robust framework also introduces a randomized-direction oracle that uses a single LLM call per pair. This approach converts systematic position bias into zero-mean noise, enabling unbiased aggregate ranking without the cost of bidirectional calls.[1]

1 Introduction

LLMs are increasingly used for reranking in Retrieval-Augmented Generation (RAG): given a query and a candidate list, rerankers aggregate LLM pairwise preferences into an ordered top-K subset that strongly affects downstream answer quality (Zhou et al., 2025; Zhu et al., 2023; Dong et al., 2024; Sun et al., 2025). Major cloud providers now offer reranking as managed services, making call efficiency a first-class concern: LLM invocations dominate cost and latency, and the goal is a sorted prefix, not the full list.

Commonly, PRP is paired with classical sorting algorithms (Qin et al., 2024; Sun et al., 2023): PRP supplies noisy preference judgments while sorting determines which pairs to query. This is structurally mismatched: sorting assumes transitive comparisons, while LLM judgments are stochastic and can violate transitivity. Sorting thus wastes budget polishing an unstable permutation rather than improving the top-K.

LLM order effects, where swapping document presentation order can flip the judge's choice between documents, further complicate matters (Shi et al., 2024; Yin et al., 2025; Jeong et al., 2025). Standard PRP queries both prompt directions at 2 calls per pair (Qin et al., 2024; Wu et al., 2025), yet preference cycles persist.

We therefore frame PRP reranking as active learning from noisy pairwise comparisons, choosing adaptively which pairs to query to maximize top-K quality within a budget. This connects to the literature on best-K identification under stochastic feedback (Mohajer et al., 2017; Heckel et al., 2016; Shah and Wainwright, 2018; Ren et al., 2020; Luo et al., 2024). We also evaluate a cheaper oracle: randomizing the prompt direction yields a one-call estimate that converts position bias into zero-mean noise.

We study two questions: **(Q1)** *Does active ranking outperform state-of-the-art PRP rerankers at fixed budget (NDCG@10)?* **(Q2)** *Does randomized-direction prompting improve the NDCG@10–cost trade-off beyond scheduling alone?* The best-performing active scheduler in our experiments is the algorithm of Mohajer et al. (2017), which we call **Mohajer**: it adaptively selects which pairs to query, concentrating comparisons near the top-K boundary. **Q1**: On TREC DL2019/2020 with Flan-T5-XL, Mohajer outperforms the best sorting baseline by +9.7 NDCG@10 at B=300 calls (66.1 vs. 56.4), under the same bidirectional oracle, with the advantage holding across the entire call-constrained regime (B=200–450). **Q2**: Randomized-direction prompting improves both strategies, but in different ways. For PRP rerankers, it raises quality at fixed budget: BubbleSort gains +5.5 NDCG@10 at B=300 (56.4→62.0) simply by halving the call cost per pair and covering more comparisons. For active rankers, the effect is more

[1] Code available at https://anonymous.4open.science/r/prp-active-reranking-4B77

pronounced: comparing Mohajer under both oracles, the randomized-direction oracle raises the quality ceiling from 66.96 to 68.0 while reducing the calls needed to reach it from B=450 to B=250, a 44% reduction. Across BEIR-style tasks, active rankers reach NDCG@10 comparable to QuickSort (Avg. 56.8 for Flan-T5-XL) with up to 7× fewer calls.

2 Related Work

Pairwise LLM reranking. PRP elicits pairwise preferences from an LLM and aggregates them into a ranking (Sun et al., 2023; Qin et al., 2024), typically via sorting algorithms that assume transitivity and target an unbudgeted complete order.

Order effects. LLM comparisons are direction-sensitive (Shi et al., 2024; Yin et al., 2025; Jeong et al., 2025), so PRP often queries both prompt orders, doubling cost (Qin et al., 2024; Wu et al., 2025). Our randomized-direction oracle makes one call per pair, producing aggregate outcomes robust to direction bias.

PRP beyond sorting. PRP-Graph uses adaptive pairings (Luo et al., 2024) and tournament designs structure comparisons (Chen et al., 2024). We reframe reranking as active learning from noisy feedback and evaluate active top-K identifiers as drop-in replacements for sorting (Heckel et al., 2016; Shah and Wainwright, 2018; Ren et al., 2020; Mohajer et al., 2017), using adaptive pairings in the spirit of PRP-Graph but grounded in noise-tolerant active ranking theory.

Complementary paradigms. Setwise and listwise methods (Zhuang et al., 2024; Huang et al., 2025; Wang et al., 2025) reduce cost by processing multiple documents per call, changing the prompting primitive itself; pairwise and listwise calls differ in token cost, context length, and bias, making raw call counts incommensurable across paradigms. Our goal is to improve scheduling *within* pairwise PRP, which remains widely deployed for its fine-grained signal and constrained-output reliability (Qin et al., 2024); the two directions are complementary.

3 Reranking from Noisy Comparisons

Given a query q, a first-stage retriever returns N candidates $\mathcal{D}(q) = \{d_1, \ldots, d_N\}$ ($N \geq K$). The reranker outputs an ordered top-K list $\mathcal{R}_K(q) = (r_1, \ldots, r_K)$ with $r_\ell \in \mathcal{D}(q)$.

Pairwise oracle interface. Algorithms interact with candidates only via noisy pairwise outcomes: for each unordered pair $\{i, j\}$, a call returns $X_{ij}(q) \in \{0, 1\}$, where $X_{ij}(q) = 1$ means d_i is preferred (judged more relevant to q) over d_j, i.e. $d_i \succ d_j$, with win probability $p_{ij}(q) := \Pr[X_{ij}(q) = 1]$. We assume only *pair-consistency*, $p_{ij}(q) = 1 - p_{ji}(q)$ for $i \neq j$ (this is enforced via oracle design).

Call-centric cost. We count LLM inference calls: bidirectional uses two per pair, randomized-direction uses one. Since calls dominate PRP cost (Wisznia et al., 2025), this changes which routines are optimal.

Oracles Let $\mathrm{LLM}(d_a, d_b) \in \{1, 0\}$ denote the outcome of one call, where 1 means the first document is preferred.

Bidirectional (two calls). The standard PRP oracle (Qin et al., 2024): $V_{ij} = 1$ iff $\mathrm{LLM}(d_i, d_j) = 1 \wedge \mathrm{LLM}(d_j, d_i) = 0$, else $V_{ij} = 0$.

Randomized-direction (one call). We randomize input order: $V_{ij} = \mathrm{LLM}(d_i, d_j)$ with probability $1/2$, else $V_{ij} = 1 - \mathrm{LLM}(d_j, d_i)$. This ensures reciprocity in expectation, i.e. $\Pr[V_{ij}{=}1] = 1{-}\Pr[V_{ji}{=}1]$: each individual call may be position-biased, but averaging over the random direction converts systematic bias into zero-mean noise, preserving pair-consistency (proof in Appendix E).

4 Selecting Active Rankers for Call-Budgeted Top-K Reranking

Sorting treats every comparison as equally informative. Under a budget, this uniformity is wasteful. Active rankers concentrate comparisons on candidates whose relative order remains uncertain. This is the key mechanism behind our gains: a better schedule for the same comparator, requiring only lightweight bookkeeping with no model training or forward passes. The dominant cost remains the LLM call itself.

Our goal is high-quality top-K prefixes under a strict call budget B via the pairwise-oracle interface of §3. We select algorithms with three criteria: **(C1)** *Top-K objective*: targets best-K/prefix identification; **(C2)** *Noise tolerance*: well-defined under pair-consistency without assuming a global order; **(C3)** *Anytime behavior*: outputs a competitive top-K prefix as comparisons accrue.

We focus on comparison scheduling gains and benchmark two complementary active rankers: tournament-based vs. anchor-based. Methods as-

Oracle	Ranker	Number of LLM calls (budget)								
		100	150	200	250	300	350	400	450	500
Bidirectional	BubbleSort	49.27	49.27	56.43	56.43	56.42	56.98	60.25	60.30	60.51
	HeapSort	6.81	6.13	6.13	9.28	23.04	41.84	54.29	62.81	**68.21**
	QuickSort	**55.93**	**55.89**	55.87	56.20	56.20	56.20	56.40	56.59	56.68
	PAC + Bubble	49.27	49.27	49.27	49.27	57.52	60.59	60.61†	60.61	60.61
	Mohajer + Bubble	30.12	30.12	**62.34**	**64.80**	**66.09**	**66.28**	**66.83**	**67.02**†	67.02
	Mohajer	30.12	30.12	**62.34**	**64.80**	**66.09**	**66.28**	66.81	66.96†	66.96
Randomized	BubbleSort	55.90±0.28	56.10±0.21	59.82±0.20	59.68±0.21	61.95±0.21	62.03±0.18	64.04±0.19	64.00±0.25	65.42±0.30
	HeapSort	6.58±0.14	16.46±0.43	50.17±0.24	65.80±0.16	**68.50±0.39**	**68.41±0.30**	**68.34±0.11**	**68.53±0.21**	**68.71±0.21**
	QuickSort	54.49±0.27	54.71±0.23	55.26±0.24	56.20±0.47	57.56±0.19	58.95±0.34	59.81±0.41	61.87±0.36	63.76±0.38
	PAC + Bubble	49.27±0.00	57.02±0.21	60.01±0.21†	60.01±0.21	60.01±0.20	60.01±0.21	60.01±0.20	60.01±0.21	60.01±0.21
	Mohajer + Bubble	**61.36±0.31**	**65.84±0.33**	**67.66±0.35**	67.59±0.34	68.14±0.26	68.25±0.27	68.08±0.12†	68.08±0.12	68.08±0.12
	Mohajer	**61.36±0.31**	**65.84±0.33**	**67.66±0.35**	**68.00±0.19**†	68.00±0.19	68.00±0.19	68.00±0.19	68.00±0.19	68.00±0.19

Table 1: Average NDCG@10 (%) on TREC DL 2019 and DL 2020 with Flan-T5-XL across LLM call budgets. Bold = best per column; underline = second-best (within each oracle block). † indicates the smallest budget at which a method completes; results at larger budgets are de-emphasized. **Randomized** rows report mean ± 95% bootstrap CI half-width over 8 oracle seeds (10k resamples); **Bidirectional** is deterministic given outcomes so CIs are omitted.

suming transitivity or targeting a full global ranking are omitted.

Tournament/heap extraction. Mohajer et al. (2017) identifies best-K via tournaments with heap extraction, focusing comparisons on likely contenders (C1–C3). We use one oracle call per match.

Anchor-based Probably Approximately Correct (PAC) best-K. Agarwal et al. (2022) identifies best-K via anchors and winner sets (C1, C3). We take anchors from a zero-cost BM25 prior and restrict comparisons to the top $K \times m$ (m=3) BM25 prefix, keeping calls low.

Ordered outputs. PAC returns an unordered best-K set, so we apply BubbleSort on the final top-K. Mohajer outputs an ordered prefix; BubbleSort polishing is optional and negligible. Any added comparisons count toward the budget.

5 Results

Setup. We rerank the top N=100 BM25 candidates into an ordered top-K list (K=10) and report NDCG@10 on BEIR-style tasks (Thakur et al., 2021; Voorhees et al., 2021; Voorhees, 2005; Bondarenko et al., 2020; Wadden et al., 2020; Hasibi et al., 2017; Maia et al., 2018) (Table 2) and TREC DL2019/2020 (Bajaj et al., 2016; Craswell et al., 2020, 2021), capping each method at $B \in \{100, 150, \ldots, 500\}$ LLM calls. The pairwise oracle uses Flan-T5-L/XL (Raffel et al., 2020; Chung et al., 2024) under (i) bidirectional and (ii) randomized-direction prompting. BubbleSort uses caching (Wisznia et al., 2025). Additional Qwen results (Yang et al., 2025) and code are in the appendix/repository.

Main findings. Table 1 reports NDCG@10 on TREC DL2019/2020 (Flan-T5-XL) vs. budget B (CIs and bootstrap tests in Appendix D). *(i)* In the call-constrained regime ($B \approx$ 200–450), Mohajer outperforms PRP rerankers under the same oracle. *(ii)* Randomized-direction compresses "time-to-quality": Mohajer reaches peak quality by B=250. *(iii)* At high budgets, sorting catches up as global refinement pays off. PAC lags because its two-phase design splits budget across objectives; Mohajer's tournament concentrates comparisons on likely top candidates. PAC benefits when the BM25 prior is strong (e.g., Touché).

Bidirectional oracle Low budgets: sorting is preferable. At $B \in \{100, 150\}$, QuickSort reaches $\approx$ 55.9 NDCG@10 while Mohajer is in warm-up (30.1). Below the warm-up threshold ($\sim$100 calls for N=100, K=10) sorting is preferable; above it, active ranking dominates.

Call-constrained regime: active reranking is better. Mohajer leads from B=200 to B=450: at B=300, 66.09 vs. 56.42 (+9.67); at B=350, 66.28 vs. 56.98 (+9.30); at B=450, Mohajer+Bubble 67.02 vs. HeapSort 62.81 (+4.21). Paired bootstrap tests (10k query resamples, p<0.05; Table A.8 in the appendix) confirm these gains are significant: under the randomized oracle, Mohajer+Bubble significantly outperforms BubbleSort at every budget; under bidirectional, from B=200 onward.

High budgets: sorting can catch up. At B=500, HeapSort (68.21) narrowly exceeds Mohajer+Bubble (67.02) as global refinement pays off.

Randomized oracle: faster "time-to-quality" for active ranking Using one call per pair, randomized-direction covers $\sim$2$\times$ as many pairs as bidirectional at the same budget. Mohajer is already strong at B=100 (61.4) and converges at 68.0 by B=250, suggesting that broader pair coverage may outweigh spending two calls per pair

	Ranker	Covid	Robust04	Touche	SciFact	DBPedia	DL19	DL20	Avg. NDCG@10	Avg. Calls/Task
	BM25	59.5	40.7	44.2	67.9	31.9	50.6	48.0	49.0	-
Flan-T5-L	BubbleSort@10 (Bidirectional)	70.9	**44.2**	**44.7**	**69.2**	**41.7**	63.4	58.6	**56.1**	679
	HeapSort (Bidirectional)	76.0	40.4	33.2	67.5	41.4	**65.0**	**62.6**	55.2	1230
	QuickSort (Bidirectional)	76.2	41.0	27.4	60.1	41.1	64.5	58.5	52.7	1954
	PAC+Bubble (Bidirectional)	69.3	44.0	41.4	68.5	39.2	61.7	57.2	54.5	323
	PAC+Bubble (Randomized)	70.2	41.0	38.2	67.0	38.1	60.0	57.3	53.1	184
	Mohajer+Bubble (Bidirectional)	76.5	37.8	26.7	54.9	40.0	63.0	56.4	50.7	423
	Mohajer+Bubble (Randomized)	**76.9**	37.8	25.7	58.8	40.0	61.8	58.8	51.4	354
	Mohajer (Bidirectional)	76.5	37.5	26.4	53.8	39.7	62.6	56.1	50.4	399
	Mohajer (Randomized)	76.2	36.2	24.4	57.5	39.1	60.6	57.2	50.2	232
Flan-T5-XL	BubbleSort@10 (Bidirectional)	74.8	**55.4**	**42.8**	**71.3**	**43.1**	68.4	67.0	**60.4**	941
	HeapSort (Bidirectional)	78.2	54.9	28.4	70.6	41.6	**70.6**	**68.9**	59.0	1409
	QuickSort (Bidirectional)	77.2	53.7	25.8	61.4	41.6	70.4	67.2	56.8	1669
	PAC+Bubble (Bidirectional)	71.3	48.9	41.1	70.4	39.1	62.6	58.6	56.0	332
	PAC+Bubble (Randomized)	71.3	48.1	38.8	68.6	38.8	61.1	58.5	55.0	184
	Mohajer+Bubble (Bidirectional)	76.0	53.7	25.7	61.9	40.9	66.6	67.5	56.0	427
	Mohajer+Bubble (Randomized)	**78.5**	54.0	27.9	63.5	41.2	69.5	66.3	57.3	345
	Mohajer (Bidirectional)	76.0	53.6	25.4	61.2	40.7	66.6	67.3	55.8	399
	Mohajer (Randomized)	77.6	53.2	27.2	62.8	40.4	68.7	67.6	56.8	232

Table 2: End-to-end BEIR-style NDCG@10 (%) and average pairwise LLM calls. Bold = best per column; underline = second-best per column (within each model block).

to reduce order effects. HeapSort surpasses Mohajer at B=300 (68.50 vs. 68.00), reaching 68.71 at B=500; sorting remains preferable once budgets are large enough for global refinement to dominate.

End-to-end efficiency in the full pipeline Table 2 reports end-to-end results (N=100, K=10). For Flan-T5-XL, PRP baselines use 941–1669 calls/task (Avg. 56.8–60.4 NDCG@10), while Mohajer and PAC use 184–345 calls (Avg. 55.0–57.3), a $\sim$3–5$\times$ reduction. Randomized-direction further cuts calls: Mohajer drops from 399 to 232 per task, making adaptive scheduling with randomized oracle a strong low-cost design. Touché is an outlier where BM25 is strong and neural reranking headroom is limited.

Order effects and comparisons. Bidirectional prompting reverses the preferred document on 20.6% of pairs, confirming substantial order effects. Mohajer+Bubble achieves competitive or better NDCG@10 than PRP-Graph with fewer comparisons as model size increases. A top-k sweep shows the crossover where global refinement becomes preferable moves earlier as k increases. Details in the appendix.

Latency Latency is estimated as a sequential upper bound: inference time per comparison times total comparisons, ignoring parallelism (see Appendix A). Figure 1 shows active ranking reaching strong quality earlier (Mohajer at 23.3s, PAC at 10.1s), with sorting overtaking only at long runtimes. Both active rankers support within-query parallelism (independent tournaments / anchor comparisons), potentially reducing wall-clock time by an order of magnitude. Qwen and H100/H200 results in the appendix.

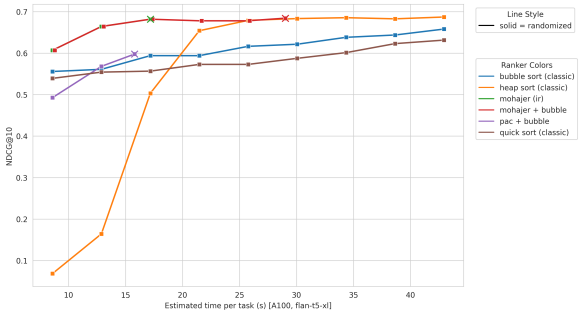

Figure 1: TREC DL 2019/2020 (Flan-T5-XL, A100): NDCG@10 vs. avg. time per task (randomized oracle). X marks the point at which a method has completed all its scheduled comparisons (convergence).

6 Conclusion

We argue that PRP reranking is better modeled as budgeted learning from noisy pairwise comparisons than deterministic sorting. Active rankers yield higher NDCG@10 at low budgets, while sorting helps mainly once large budgets make global refinement affordable. Randomized-direction prompting further improves efficiency: at the same budget, it covers roughly twice as many pairs as bidirectional. For practitioners deploying PRP in RAG pipelines, our results suggest a simple recipe: use Mohajer with the randomized-direction oracle when the call budget exceeds the warm-up threshold ($\sim K \times K$ calls), and fall back to sorting when budgets are either very small or large enough for global refinement.

Limitations

Our study focuses on settings where a reliable pairwise LLM comparator can be elicited with constrained outputs. Results may vary with prompt design, model family, and decoding settings. Our cost metric counts LLM calls but omits system-level overheads (batching, network latency); latency measurements are not fully end-to-end. Parallel execution was not implemented, though both algorithms naturally support it (Appendix A).

The NDCG@10 gains from randomized-direction oracles are empirically consistent but not theoretically explained; the hypothesis that independent single-direction samples benefit adaptive algorithms more than correlated bidirectional samples is plausible but unproven.

Our PAC best-K method (PAC+Bubble) introduces a candidate pool multiplier hyperparameter m (default $m = 3$), which controls the trade-off between comparison cost and coverage of the prior ranking. We did not perform a systematic ablation over m due to computational constraints, and the optimal value likely depends on prior quality and dataset characteristics. While $m = 3$ yields strong empirical results in our experiments, future work should investigate data-driven or adaptive selection strategies for this parameter.

Finally, active ranking theory often assumes conditional independence of oracle outputs; real LLM APIs can violate this through hidden state, caching, or nonstationarity.

References

Arpit Agarwal, Sanjeev Khanna, and Prathamesh Patil. 2022. Pac top-k identification under sst in limited rounds. In *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 6814–6839. PMLR.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. 2016. MS MARCO: A human generated MAchine reading COmprehension dataset. *arXiv preprint arXiv:1611.09268*.

Alexander Bondarenko, Maik Fröbe, Meriem Beloucif, Lukas Gienapp, Yamen Ajjour, Alexander Panchenko, Chris Biemann, Benno Stein, Henning Wachsmuth, Martin Potthast, and Matthias Hagen. 2020. Overview of Touché 2020: Argument retrieval. In *Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum*, volume 2696 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Yiqun Chen, Qi Liu, Yi Zhang, Weiwei Sun, Daiting Shi, Jiaxin Mao, and Dawei Yin. 2024. Tourrank: Utilizing large language models for documents ranking with a tournament-inspired strategy. *CoRR*, abs/2406.11678.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, and 16 others. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53.

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

Jialin Dong, Bahare Fatemi, Bryan Perozzi, Lin F. Yang, and Anton Tsitsulin. 2024. Don't forget to connect! improving rag with graph-based reranking. *arXiv preprint arXiv:2405.18414*.

Faegheh Hasibi, Fedor Nikolaev, Chenyan Xiong, Krisztian Balog, Svein Erik Bratsberg, Alexander Kotov, and Jamie Callan. 2017. DBpedia-Entity v2: A test collection for entity search. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1265–1268. ACM.

Reinhard Heckel, Nihar B. Shah, Kannan Ramchandran, and Martin J. Wainwright. 2016. Active ranking from pairwise comparisons and when parametric assumptions don't help. *arXiv preprint arXiv:1606.08842*.

Jerry Huang, Siddarth Madala, Cheng Niu, Julia Hockenmaier, and Tong Zhang. 2025. Contextual relevance and adaptive sampling for LLM-based document reranking. *CoRR*, abs/2511.01208.

Hawon Jeong, ChaeHun Park, Jimin Hong, Hojoon Lee, and Jaegul Choo. 2025. The comparative trap: Pairwise comparisons amplifies biased preferences of LLM evaluators. In *Proceedings of the 8th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 79–108, Suzhou, China. Association for Computational Linguistics.

Jian Luo, Xuanang Chen, Ben He, and Le Sun. 2024. PRP-graph: Pairwise ranking prompting to LLMs with graph aggregation for effective text re-ranking. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5766–5776, Bangkok, Thailand. Association for Computational Linguistics.

Macedo Maia, Siegfried Handschuh, André Freitas, Brian Davis, Ross McDermott, Manel Zarrouk, and Alexandra Balahur. 2018. WWW'18 open challenge: Financial opinion mining and question answering. In *Companion Proceedings of The Web Conference 2018*, pages 1941–1942. ACM.

Soheil Mohajer, Changho Suh, and Adel Elmahdy. 2017. Active learning for top-k rank aggregation from noisy comparisons. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2488–2497. PMLR.

Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Le Yan, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, Xuanhui Wang, and Michael Bendersky. 2024. Large language models are effective text rankers with pairwise ranking prompting. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 1504–1518, Mexico City, Mexico. Association for Computational Linguistics.

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.

Wenbo Ren, Jia Liu, and Ness Shroff. 2020. The sample complexity of best-k items selection from pairwise comparisons. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 8051–8072.

Nihar B. Shah and Martin J. Wainwright. 2018. Simple, robust and optimal ranking from pairwise comparisons. *Journal of Machine Learning Research*, 18(199):1–38.

Lin Shi, Chiyu Ma, Wenhua Liang, Xingjian Diao, Weicheng Ma, and Soroush Vosoughi. 2024. Judging the judges: A systematic study of position bias in llm-as-a-judge. *arXiv preprint arXiv:2406.07791*.

Jiashuo Sun, Xianrui Zhong, Sizhe Zhou, and Jiawei Han. 2025. Dynamicrag: Leveraging outputs of large language model as feedback for dynamic reranking in retrieval-augmented generation. *arXiv preprint arXiv:2505.07233*.

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

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Advances in Neural Information Processing Systems*, volume 34.

Ellen M. Voorhees. 2005. Overview of the TREC 2004 robust retrieval track. NIST Special Publication 500-261, National Institute of Standards and Technology.

Ellen M. Voorhees, Ian Soboroff, Kirk Roberts, Tasmeer Alam, Steven Bedrick, Dina Demner-Fushman, Kyle Lo, Lucy Lu Wang, and William R. Hersh. 2021. Searching for answers in a pandemic: An overview of TREC-COVID. *Journal of Biomedical Informatics*, 121:103865.

David Wadden, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. 2020. Fact or fiction: Verifying scientific claims. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 7534–7550, Online. Association for Computational Linguistics.

Pinhuan Wang, Zhiqiu Xia, Chunhua Liao, Feiyi Wang, and Hang Liu. 2025. REALM: Recursive relevance modeling for LLM-based document re-ranking. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 23875–23889, Suzhou, China. Association for Computational Linguistics.

Juan Wisznia, Cecilia Bolaños, Juan Tollo, Giovanni Franco Gabriel Marraffini, Agustín Andrés Gianolini, Noe Fabian Hsueh, and Luciano Del Corro. 2025. Are optimal algorithms still optimal? rethinking sorting in LLM-based pairwise ranking with batching and caching. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1064–1072, Vienna, Austria. Association for Computational Linguistics.

Jingyu Wu, Aditya Shrivastava, Jing Zhu, Alfy Samuel, Anoop Kumar, and Daben Liu. 2025. Llm optimization unlocks real-time pairwise reranking. *arXiv preprint arXiv:2511.07555*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Haonan Yin, Shai Vardi, and Vidyanand Choudhary. 2025. Fragile preferences: A deep dive into order effects in large language models. *arXiv preprint arXiv:2506.14092*.

Yinxin Zhou, Qin Luo, Bin Feng, and Bang Wang. 2025. Large language models for reranking: A survey. `https://doi.org/10.36227/techrxiv.176300630.01740917/v1`. TechRxiv preprint.

Yutao Zhu, Huaying Yuan, Shuting Wang, Jiongnan Liu, Wenhan Liu, Chenlong Deng, Haonan Chen, Zheng

Liu, Zhicheng Dou, and Ji-Rong Wen. 2023. Large language models for information retrieval: A survey. *arXiv preprint arXiv:2308.07107*.

Shengyao Zhuang, Honglei Zhuang, Bevan Koopman, and Guido Zuccon. 2024. A setwise approach for effective and highly efficient zero-shot ranking with large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '24)*, pages 38–47. ACM.

A Parallelization Opportunities

Both our proposed active ranking algorithms exhibit substantial parallelization potential that could significantly reduce wall-clock latency in practice.

Mohajer. The Mohajer algorithm's structure naturally lends itself to parallel execution at all levels. First, the K independent tournaments used for champion selection can be executed concurrently, as each tournament operates on a disjoint subset of candidates with no inter-group dependencies. Second, the heapify operation itself is amenable to parallel construction techniques, allowing the initial heap formation to proceed in $O(\log n)$ depth rather than sequential $O(n)$ time. Finally, within each tournament, pairwise comparisons at the same tree depth are independent and can be batched for simultaneous LLM inference.

Optimized PAC. Our optimized PAC algorithm also presents parallelization opportunities, where comparisons of candidates against the selected anchors are entirely independent. With $K/2$ anchors and a candidate pool of size $K \times$ pool_mult, these $O(K^2)$ comparisons can be executed in parallel, subject only to the LLM inference throughput. The subsequent winner-set construction and greedy accumulation require only lightweight aggregation, which remains negligible compared to comparison costs.

Latency implications. Assuming an LLM inference API with high throughput (e.g., batched GPU inference or distributed serving), parallelization could theoretically reduce Mohajer's latency from $O(n \log K)$ sequential comparisons to $O(\log Q \cdot \log K)$ parallel rounds, and PAC's latency from $O(K\sqrt{n})$ comparisons to $O(\sqrt{n})$ rounds with batch size K. In our TREC DL experiments, where Mohajer performs $\sim$350 comparisons and PAC $\sim$185 comparisons, parallelization with batch size 10 could reduce round counts to $\sim$35 and $\sim$19 respectively, yielding substantial speedups on systems where comparison latency dominates over LLM throughput.

B Supplementary Tables

This section provides supplementary results and ablations referenced in the main text.

(a) TREC DL2019

Oracle	Ranker	Number of LLM calls (budget)								
		100	150	200	250	300	350	400	450	500
Bidirectional	BubbleSort	50.58	50.58	57.13	57.13	57.13	57.78	60.64	60.69	60.95
	HeapSort	9.14	8.49	8.61	11.91	23.97	41.68	54.14	63.58	**68.92**
	QuickSort	**58.08**	**58.08**	58.30	58.57	58.57	58.57	58.85	59.08	59.27
	PAC + Bubble	50.58	50.58	50.58	50.58	60.00	62.59	62.60	62.60	62.60
	Mohajer + Bubble	32.76	32.76	**62.44**	**65.35**	**66.34**	**66.23**	**66.64**	**66.59**	66.59
	Mohajer	32.76	32.76	**62.44**	**65.35**	**66.34**	**66.23**	**66.64**	66.58	66.58
Randomized	BubbleSort	55.92	56.88	59.77	60.33	61.98	62.52	64.27	64.37	65.73
	HeapSort	9.11	16.89	49.27	65.93	68.49	69.03	**69.82**	68.76	68.84
	QuickSort	56.33	57.57	56.98	59.89	58.42	61.44	63.00	64.83	64.37
	PAC + Bubble	50.58	58.35	61.05	61.05	61.05	61.05	61.05	61.05	61.05
	Mohajer + Bubble	**61.41**	**67.79**	**68.70**	67.99	68.53	**69.17**	69.47	**69.47**	**69.47**
	Mohajer	**61.41**	**67.79**	**68.70**	**68.73**	**68.73**	68.73	68.73	68.73	68.73

(b) TREC DL2020

Oracle	Ranker	Number of LLM calls (budget)								
		100	150	200	250	300	350	400	450	500
Bidirectional	BubbleSort	47.96	47.96	55.72	55.72	55.72	56.18	59.86	59.90	60.07
	HeapSort	4.48	3.77	3.65	6.65	22.12	42.00	54.44	62.05	**67.50**
	QuickSort	**53.77**	**53.69**	53.44	53.83	53.83	53.83	53.95	54.10	54.10
	PAC + Bubble	47.96	47.96	47.96	47.96	55.04	58.59	58.62	58.62	58.62
	Mohajer + Bubble	27.48	27.48	**62.24**	**64.25**	**65.83**	**66.32**	**67.02**	**67.45**	67.45
	Mohajer	27.48	27.48	**62.24**	**64.25**	**65.83**	**66.32**	66.98	67.35	67.35
Randomized	BubbleSort	55.23	55.31	59.03	58.45	61.30	61.78	63.35	64.37	65.88
	HeapSort	4.59	15.96	51.38	64.88	67.31	**67.64**	67.27	**67.78**	**68.58**
	QuickSort	51.51	53.28	54.34	54.67	56.15	56.05	57.25	59.76	61.91
	PAC + Bubble	47.96	55.27	58.51	58.51	58.51	58.51	58.51	58.51	58.51
	Mohajer + Bubble	**59.95**	**65.06**	**67.66**	67.61	67.05	67.60	66.30	66.30	66.30
	Mohajer	**59.95**	**65.06**	**67.66**	**67.62**	**67.62**	67.62	**67.62**	67.62	67.62

Table A.1: Budgeted NDCG@10 (%) on TREC DL2019 and DL2020 using Flan-T5-XL under bidirectional vs. randomized-direction prompting. Budgets denote the number of LLM calls. Within each oracle block, bold/underline indicate best/second-best per budget column among the listed rankers.

k	Ranker	Number of LLM calls (budget)								
		100	150	200	250	300	350	400	450	500
10	BubbleSort	55.57	56.10	59.40	59.39	61.64	62.15	63.81	64.37	65.80
	Mohajer	**60.68**	**66.42**	**68.18**†	**68.18**	**68.18**	**68.18**	**68.18**	**68.18**	**68.18**
20	BubbleSort	**52.73**	53.07	55.76	55.62	57.36	57.60	59.18	59.31	60.49
	Mohajer	51.51	**58.30**	**61.82**	**62.98**†	**62.98**	**62.98**	**62.98**	**62.98**	**62.98**
30	BubbleSort	**51.62**	52.02	54.06	53.88	55.49	55.61	57.07	57.19	58.02
	Mohajer	48.61	**54.04**	**56.69**	**58.22**†	**58.22**	**58.22**	**58.22**	**58.22**	**58.22**
40	BubbleSort	**51.47**	51.78	53.77	53.58	55.17	55.44	**56.38**	**56.86**	**57.19**
	Mohajer	48.06	**52.94**	**54.49**	**56.03**†	**56.03**	**56.03**	56.03	56.03	56.03
50	BubbleSort	**51.46**	52.38	53.59	53.83	54.71	**55.31**	**55.80**	**56.17**	**56.49**
	Mohajer	48.12	**52.72**	**53.96**	**55.03**†	**55.03**	55.03	55.03	55.03	55.03

Table A.2: Top-k sweep with randomized-direction prompting (Flan-T5-XL). Values are average NDCG@k (%) over TREC DL2019 and DL2020. Budgets denote the number of LLM calls. For each (k, budget), bold marks the better of BubbleSort vs. Mohajer. † indicates the first budget where the method is treated as converged; subsequent budgets are de-emphasized.

Note. Mohajer dominates BubbleSort when k and the budget are low. As k increases, the crossover point where BubbleSort overtakes Mohajer arrives sooner in terms of number of LLM calls.

LLM	Method	DL2019		DL2020	
		NDCG@10 (%)	#Inf.	NDCG@10 (%)	#Inf.
NA	BM25	50.6	–	48.0	–
Flan-T5-L	PRP-Graph-10	**65.8**	492.7	**61.8**	492.5
	Mohajer + Bu.	61.75	354.60	58.8	354.245
	PAC + Bu.	59.9	183.34	57.29	183.99
Flan-T5-XL	PRP-Graph-10	67.6	492.6	66.1	492.5
	Mohajer + Bu.	**69.47**	338.11	**66.3**	338.74
	PAC + Bu.	61.05	184.16	58.51	183.72
Flan-T5-XXL	PRP-Graph-10	66.6	492.6	66.1	492.6
	Mohajer + Bu.	**68.54**	341.81	**68.96**	344.17
	PAC + Bu.	60.92	183.46	60.71	183.59

Table A.3: Comparison to PRP-Graph on TREC DL2019/2020. NDCG@10 is reported in percent. "#Inf." is the average number of inferred pairwise comparisons used by the method (lower is fewer comparisons). Bold/underline indicate best/second-best NDCG@10 within each (DL year, LLM) block.

Note. Our results show that with larger Flan models, Mohajer + Bubble has increasingly better results than PRP-Graph with fewer comparisons. PAC + Bubble falls behind in performance but has significantly fewer comparisons.

Method		**Dataset (NDCG@10 %)**								**Avg. NDCG@10**	**Avg. Calls/Task**
Reranker	**Oracle**	**Covid**	**Robust04**	**Touche**	**SciFact**	**DBPedia**	**DL19**	**DL20**	**FiQA**		
BubbleSort@10	Bidirectional	70.9	44.2	**44.7**	**69.2**	41.7	63.4	58.6	29.4	52.8	**653**
	Randomized	**75.7**	**46.3**	41.3	67.7	**43.6**	**67.0**	**62.5**	**32.4**	**54.6**	930
HeapSort	Bidirectional	76.0	**40.4**	**33.2**	**67.5**	**41.4**	**65.0**	**62.6**	**31.3**	**52.7**	1230
	Randomized	**76.9**	38.4	27.7	63.3	39.8	62.6	58.5	27.7	49.0	**860**
QuickSort	Bidirectional	**76.2**	**41.0**	**27.4**	**60.1**	**41.1**	**64.5**	**58.5**	**26.8**	**49.5**	1954
	Randomized	75.8	35.4	25.1	58.7	39.1	61.8	58.1	24.6	47.3	**556**
Mohajer	Bidirectional	**76.5**	**37.5**	**26.4**	53.8	**39.7**	**62.6**	56.1	**25.9**	**47.3**	399
	Randomized	76.2	36.2	24.4	**57.5**	39.1	60.6	**57.2**	24.9	47.0	**232**
Mohajer+Bubble	Bidirectional	76.5	37.5	**26.4**	53.9	39.7	**62.6**	56.1	25.9	47.3	423
	Randomized	**76.9**	**37.8**	25.7	**58.8**	**40.0**	61.8	**58.8**	**26.2**	**48.2**	**354**
Pac+Bubble	Bidirectional	69.3	**44.0**	**41.4**	**68.5**	**39.2**	**61.7**	57.2	**29.2**	**51.3**	323
	Randomized	**70.2**	41.0	38.2	67.0	38.1	60.0	**57.3**	27.2	49.9	**184**

Table A.4: End-to-end NDCG@10 (%) and average pairwise-comparison calls per task. For each reranker and column, **bold** indicates the better oracle variant (higher NDCG; lower calls). If the two oracle variants tie after rounding, both are bolded.

Note. Active-learning rankers are more robust to noisy comparisons and profit most from randomized-direction sampling. BubbleSort requires many more comparisons as cycles emerge, but said cycles allow the latent transitive ordering to surface more than with a deterministic oracle, yielding higher NDCG@10.

Stratified by dataset

Dataset	Flip-rate	#Pairs	#Flips
All	20.62%	1,202,850	248,021
DL19	21.40%	212,850	45,542
DL20	20.45%	990,000	202,479

Stratified by BM25 rank distance $|r(i) - r(j)|$

$\lvert r(i) - r(j)\rvert$	Flip-rate	#Pairs	#Flips
1–5	22.56%	92,639	20,903
6–10	20.73%	86,645	17,965
11–20	20.85%	160,762	33,524
> 20	20.36%	862,804	175,629

Table A.5: Bidirectional flip-rate sanity check, Flan-T5-XL (order effects). For each candidate pair $\{d_i, d_j\}$, we query the judge twice (once as (d_i, d_j) and once as (d_j, d_i)) and report the fraction of pairs whose winner flips. Results are shown overall and stratified by dataset and BM25 rank distance.

Oracle	Ranker	Number of LLM calls (budget)								
		100	150	200	250	300	350	400	450	500
Bidirectional	BubbleSort	50.58	50.58	57.04	57.04	57.18	58.73	60.98	61.30	61.90
	HeapSort	8.64	8.16	7.99	10.48	25.65	42.42	55.85	64.24	69.14
	QuickSort	**58.28**	**58.11**	58.04	58.63	58.71	58.71	58.71	59.09	59.67
	PAC + Bubble	50.58	50.58	50.58	50.58	60.74	64.25†	64.25	64.25	64.25
	Mohajer + Bubble	32.76	32.76	**62.97**	**66.69**	**68.60**	**69.39**	**70.08**	**70.49**†	**70.49**
	Mohajer	32.76	32.76	**62.97**	**66.69**	**68.60**	**69.39**	**70.08**	70.43†	70.43
Randomized	BubbleSort	57.49	57.79	61.11	61.45	64.32	64.24	66.44	66.27	67.12
	HeapSort	8.20	16.13	53.18	68.15	**71.52**	69.61	70.73	**71.14**	70.70
	QuickSort	54.94	56.56	58.25	57.03	59.96	60.92	62.13	61.92	64.81
	PAC + Bubble	50.58	57.00	63.02†	63.02	63.02	63.02	63.02	63.02	63.02
	Mohajer + Bubble	**63.37**	**66.49**	**69.37**	68.82	70.33	70.44	70.26†	70.26	70.26
	Mohajer	**63.37**	**66.49**	**69.37**	**70.98**†	70.98	**70.98**	**70.98**	70.98	**70.98**

Table A.6: Average NDCG@10 (%) on TREC DL 2019 with Qwen3-4B-Instruct across comparison budgets. Bold = best per column; underline = second-best per column (within each oracle block). † indicates the smallest budget at which a method completes; results at larger budgets are visually de-emphasized.

Note. Though not always the best in performance, active ranking algorithms reach competitive NDCG@10 relative to the classic algorithms, with drastic improvements in terms of average calls per task.

	Method	**Dataset (NDCG@10 %)**		**Avg. NDCG@10**	**Avg. Calls/Task**
	Ranker (Oracle)	**DL19**	**DL20**		
	BM25	50.60	48.00	49.30	–
Qwen3-4B-Instruct	BubbleSort (Bidirectional)	67.44	**70.45**	68.95	1062.69
	HeapSort (Bidirectional)	70.26	69.46	**69.86**	1383.02
	QuickSort (Bidirectional)	70.70	68.83	69.77	1567.02
	PAC + Bubble (Bidirectional)	64.25	67.09	65.67	332.55
	PAC + Bubble (Randomized)	63.02	66.05	64.54	**183.42**
	Mohajer + Bubble (Bidirectional)	70.49	66.92	68.70	425.67
	Mohajer + Bubble (Randomized)	70.26	67.58	68.92	346.54
	Mohajer (Bidirectional)	70.43	66.92	68.67	398.26
	Mohajer (Randomized)	**70.98**	65.43	68.20	232.01

Table A.7: TREC DL2019/2020 end-to-end NDCG@10 (%) and average pairwise-comparison calls per task using **Qwen3-4B-Instruct**. Baselines are reported with a bidirectional oracle; our methods (PAC+Bubble and Mohajer variants) include both bidirectional and randomized comparisons. Within the Qwen block, best is bold and second-best is underlined per column (higher NDCG; lower calls).

Note. Though not always the best in performance, active ranking algorithms reach competitive NDCG@10 relative to the classic algorithms, with drastic improvements in terms of average calls per task.

C Supplementary Graphs

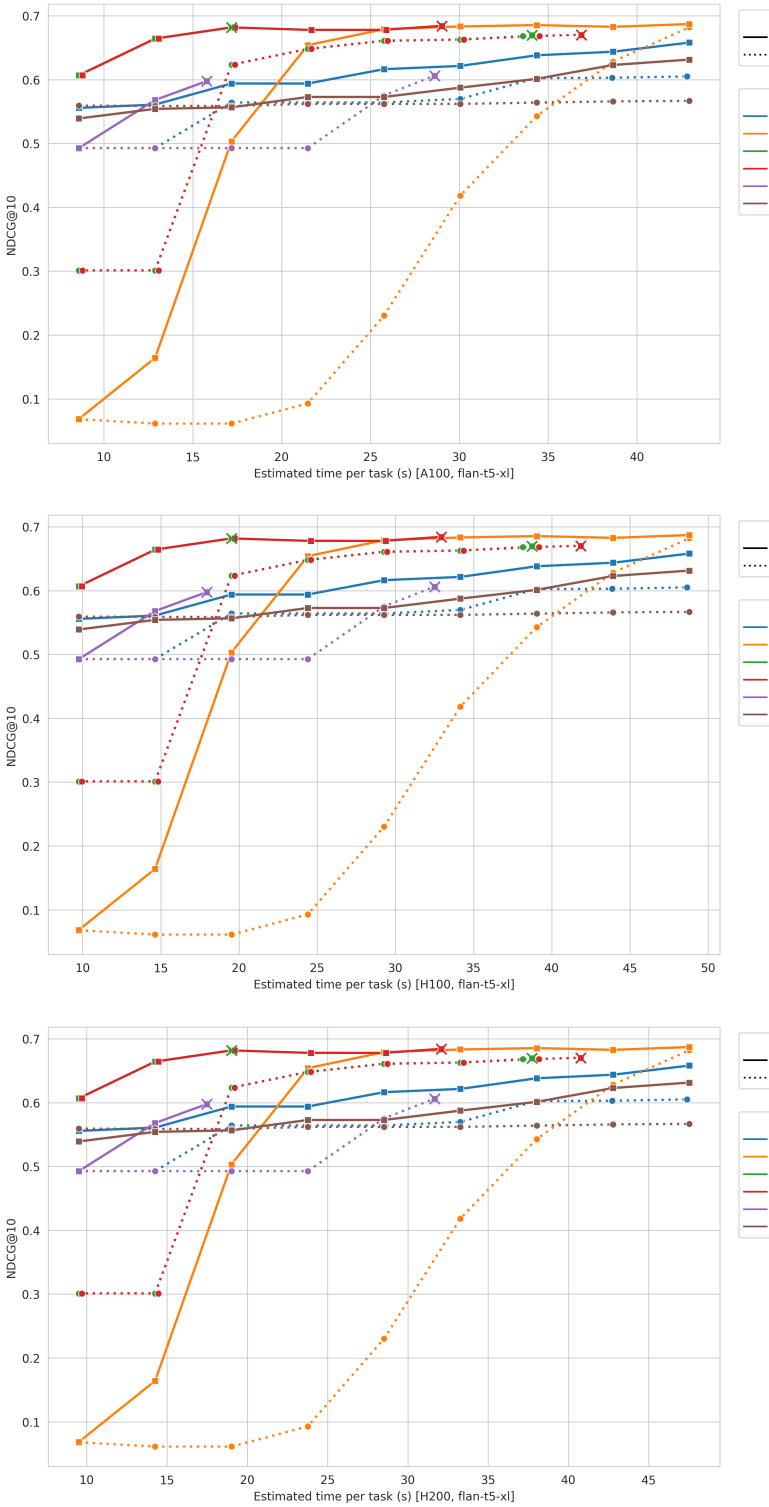

Figure 2: TREC DL 2019 and DL 2020 (Flan-T5-XL): NDCG@10 vs estimated time per task across GPUs, with both oracles shown. Colors denote rankers; solid lines are randomized and dotted lines are bidirectional oracles. X marks show when an algorithm has converged.

This section presents supplementary figures for the latency experiments referenced in the main text.

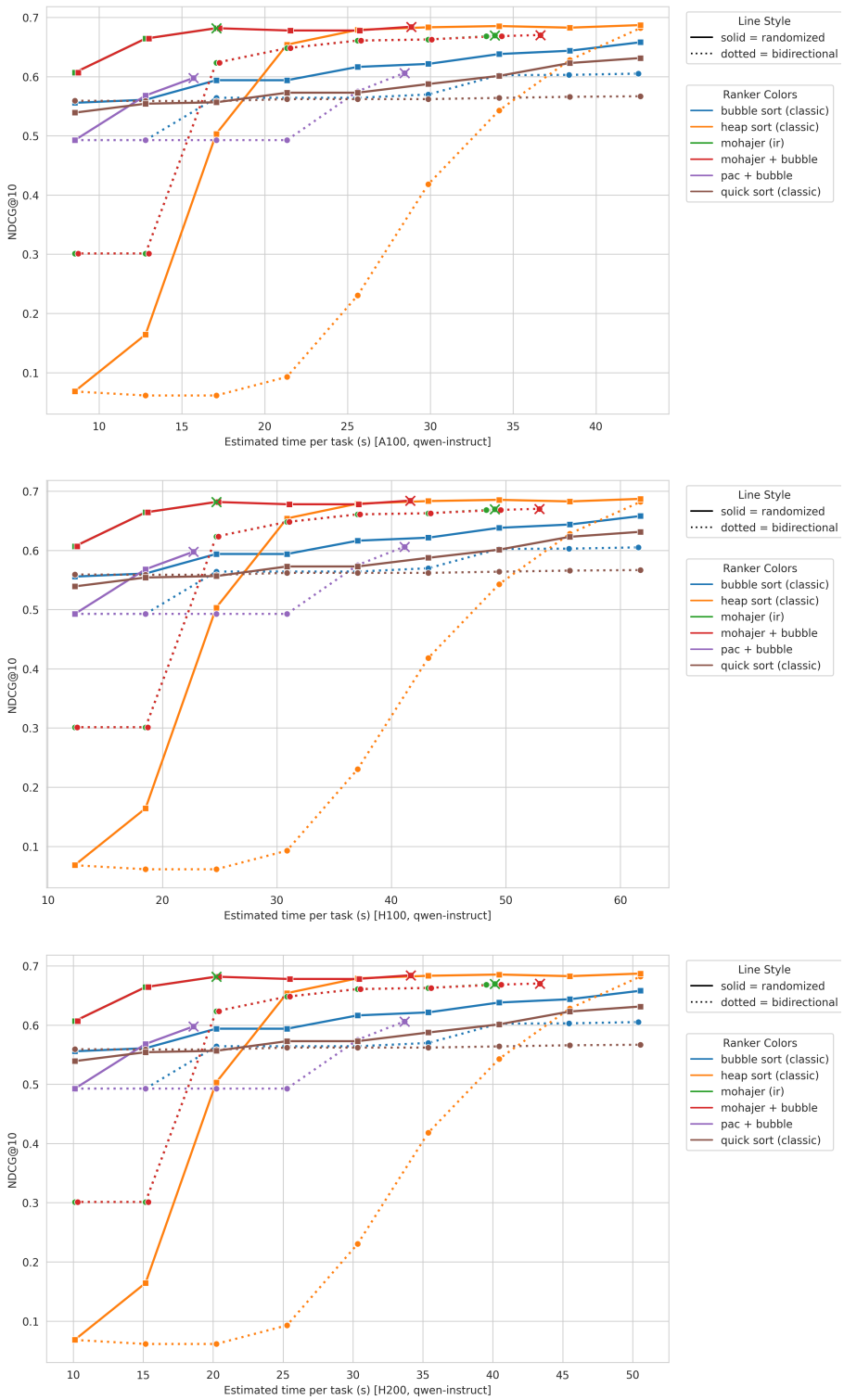

Figure 3: TREC DL 2019 and DL 2020 (Qwen3-4B-Instruct-2507): NDCG@10 vs estimated time per task across GPUs, with both oracles shown. Colors denote rankers; solid lines are randomized and dotted lines are bidirectional. X marks show when an algorithm has converged.

D Statistical Significance

To quantify the stability and significance of the reported results, we conduct two complementary non-parametric bootstrap analyses. First, we quantify seed-resampling uncertainty: we evaluate each method under 8 different oracle seeds and report the 95% confidence interval (CI) half-width of the mean NDCG@10 across seeds (10,000 resamples). These CIs are reported directly in Table 1 for the randomized-direction oracle (the bidirectional oracle is deterministic given pairwise outcomes).

Second, we report paired bootstrap significance tests over queries. For each budget, we resample queries with replacement (10,000 resamples) and measure whether the mean difference in NDCG@10 between two methods is significantly different from zero ($p < 0.05$). Table A.8 reports tests comparing Mohajer+Bubble against BubbleSort; Table A.9 compares Mohajer+Bubble against HeapSort.

		Number of LLM calls (budget)								
Oracle	Comparison (A vs B)	100	150	200	250	300	350	400	450	500
Bidirectional	Mohajer+Bubble vs BubbleSort	↓ (-19.2)	↓ (-19.2)	↑ (+5.9)	↑ (+8.4)	↑ (+9.7)	↑ (+9.3)	↑ (+6.6)	↑ (+6.7)	↑ (+6.5)
Randomized	Mohajer+Bubble vs BubbleSort	↑ (+5.1)	↑ (+10.3)	↑ (+8.8)	↑ (+8.4)	↑ (+6.1)	↑ (+6.2)	↑ (+4.1)	↑ (+3.5)	↑ (+2.1)

Table A.8: Paired bootstrap significance over queries for Mohajer+Bubble vs. BubbleSort on TREC DL19+DL20 (Flan-T5-XL). Each cell shows the direction and mean ΔNDCG@10 difference. ↑ / ↓ indicates a statistically significant difference (paired bootstrap over queries, 10,000 resamples, $p < 0.05$); = indicates not significant.

		Number of LLM calls (budget)								
Oracle	Comparison (A vs B)	100	150	200	250	300	350	400	450	500
Bidirectional	Mohajer+Bubble vs HeapSort	↑ (+23.3)	↑ (+24.0)	↑ (+56.2)	↑ (+55.5)	↑ (+43.0)	↑ (+24.4)	↑ (+12.5)	↑ (+4.2)	= (-1.2)
Randomized	Mohajer+Bubble vs HeapSort	↑ (+53.8)	↑ (+50.0)	↑ (+17.9)	↑ (+2.4)	= (-0.1)	= (+0.1)	= (-0.7)	= (-0.4)	= (-0.8)

Table A.9: Paired bootstrap significance over queries for Mohajer+Bubble vs. HeapSort on TREC DL19+DL20 (Flan-T5-XL). Each cell shows the direction and mean ΔNDCG@10 (A−B). ↑ / ↓ indicates a statistically significant difference (paired bootstrap over queries, 10,000 resamples, $p < 0.05$); = indicates not significant.

E Proof of Aggregate Unbiasedness for Randomized-Direction Oracle

Here we show that, despite the individual order bias that is present in each inference call, the randomized-direction oracle still achieves comparison results that do not depend on input order. Namely, we show that the probability of preferring document A over document B does not depend on the order in which it is fed to the comparator, and thus fulfills the property $\Pr[V_{ij} = 1] = 1 - \Pr[V_{ji} = 1]$. Let V_{ij} be the output of the randomized-direction oracle. By definition, the oracle selects the input order (d_i, d_j) or (d_j, d_i) with equal probability $p = 0.5$. The probability of the oracle preferring d_i over d_j is given by:

$$
\begin{aligned}
\Pr[V_{ij} = 1] &= \frac{1}{2}\Pr[\text{LLM}(d_i, d_j) = 1] + \frac{1}{2}\Pr[\text{LLM}(d_j, d_i) = 0] \\
&= \frac{1}{2}\left(1 - \Pr[\text{LLM}(d_i, d_j) = 0]\right) + \frac{1}{2}\left(1 - \Pr[\text{LLM}(d_j, d_i) = 1]\right) \\
&= 1 - \frac{1}{2}\left(\Pr[\text{LLM}(d_j, d_i) = 1] + \Pr[\text{LLM}(d_i, d_j) = 0]\right) \\
&= 1 - \Pr[V_{ji} = 1]
\end{aligned}
$$

This result confirms that the oracle is reciprocal in expectation. While any single LLM inference may be biased toward a specific position, the randomization of the oracle ensures that the aggregate estimator is symmetric and unbiased with respect to the document order.