Budget-Constrained Document Re-ranking with Bayesian LM-Based Pairwise Comparisons

Anonymous ACL submission

Abstract

We propose a cost-efficient, Bayesian-inspired approach for re-ranking documents with large language models (LLMs) with pair-wise comparisons under strict inference budgets. Our method incorporates BM25 priors and TrueSkill-based uncertainty sampling to select the most informative pairs for LLM comparison. It surpasses classical sorting and binary baselines in achieving higher nDCG@10 with fewer comparisons.

1 Introduction

Large language models (LLMs) have transformed language understanding and enabled zero-shot re-ranking capabilities that enhance information retrieval (IR) pipelines (Brown et al., 2020; Zhu et al., 2024). However, because each pairwise document comparison relies on a costly LLM inference, it is essential to enforce a strict **budget** on the number of feasible queries. This constraint is especially critical in both cloud-based and on-premises deployments of Retrieval-Augmented Generation (RAG), where re-ranking forms a performance-critical component that must avoid excessive latency and expense.

The high cost of LLM inferences makes traditional sorting algorithms(Qin et al., 2024; Zhuang et al., 2024)—involving $\mathcal{O}(n \log n)$ or even $\mathcal{O}(n^2)$ comparisons—very costly. Relying on such algorithms for Pairwise Ranking Prompting (Qin et al., 2024) is therefore unrealistic in practical settings. Instead, there is a need for novel strategies that selectively prioritize comparisons to achieve robust re-ranking accuracy with minimal LLM usage.

To address these challenges, we focus on a family of *partial ranking algorithms* that generate strong partial rankings *on the fly*. This approach ensures that even if the query budget is exhausted before all comparisons are completed, a high-quality ordering of the documents is still returned. Our specific contributions are as follows:

- We propose a Bayesian-inspired, partial sorting algorithm based on Thurstone's Case V model (Thurstone, 1927) for pairwise document comparisons. Dynamically choosing the most informative pairs to execute the costly LLM comparison on based on Herbrich et al. (2006).

- We demonstrate that our approach achieves improved re-ranking performance even under budget constraints. Our method leverages BM25 scores as priors and integrates TrueSkill-based uncertainty sampling.

- We systematically analyze the trade-offs between *batch size* and *total budget*, showing that simple batching strategies and efficient partial orderings can significantly reduce inference costs while maintaining accuracy.

2 Related Work

LLMs have enabled zero-shot re-ranking via the Pairwise Ranking Process (PRP) (Qin et al., 2024), which compares document pairs through zero-shot prompting. However, these pairwise comparisons are expensive, motivating cost-aware strategies. Sorting-based PRP methods (Qin et al., 2024; Zhuang et al., 2024) reduce comparisons, yet LLM inference costs remain high, and traditional algorithms (e.g., HeapSort) can be suboptimal under noisy conditions. Luo et al. (2024) introduced a round-based framework that limits inferences per round to the number of documents, offering controlled costs but lacking the flexibility to target only the most uncertain pairs. Cost-awareness is similarly critical in retrieval-augmented generation (RAG) pipelines (Lewis et al., 2020; Zhu et al., 2024).

Active sampling and Bayesian ranking methods (e.g., ASAP (Mikhailiuk et al., 2020)) have shown promise for efficient pairwise inference. However,

these methods have not yet been adapted to LLMs, where each comparison is costly yet reflects a consistent underlying preference despite uncertainty or model error. We bridge this gap with a Bayesian-inspired framework that selectively queries pairs, integrates prior knowledge, and permits any natural number of comparisons. A full discussion is provided in Appendix A.

3 Algorithms

Pairwise comparisons for ranking documents resemble tournaments for skill estimation, where statistical methods model uncertainty (Nikolenko and Sirotkin, 2011). We adapt these methods by updating posterior distributions after each comparison to robustly re-rank documents.

ASAP (Mikhailiuk et al., 2020) models each item with a latent Gaussian score and employs Approximate Message Passing (AMP) for joint posterior updates. Although computationally expensive relative to LLM calls, it selects informative pairs by computing Expected Information Gain (EIG) for all candidates and constructing a minimum spanning tree (MST) to prioritize uncertain pairs while maintaining broad coverage.

Our Bayesian-Inspired Reranking Strategy extends Thurstone's Case V model (Thurstone, 1927) and TrueSkill-based uncertainty sampling (Herbrich et al., 2006) to learn pairwise preferences under tight LLM query budgets. Each document is assigned a latent score, seeded by a prior (e.g., BM25). We identify uncertain pairs (i, j) with preference probabilities $P_{ij} \approx 0.5$ and query the LLM using randomized order to mitigate bias. The LLM outputs are converted to z-scores $\Phi^{-1}(p_{ij})$ and used in a regularized least-squares update of the latent scores. Pre-processing BM25 scores before using them as priors and post-processing final rankings with BM25 can further boost performance.

To manage query budgets, we batch pairs into groups of size B, with each batch producing $K \times B$ comparisons. Batching amortizes overhead, reuses model states, and balances update frequency. We further address potential positional bias by querying both (i, j) and (j, i) or by randomizing document order.

The LLM-derived probabilities, once converted to z-scores, are fed into a least-squares solver for all document scores, preserving interpretability by explicitly modeling priors, likelihoods, and posteriors. Our approach is competitive with PRP-Graph (Luo et al., 2024), outperforming it on some datasets while underperforming on others, yet it flexibly handles any number of pairwise comparisons. Full details and pseudocode are provided in Appendix H.

4 Experimental Setup

We focus on re-ranking n BM25-retrieved documents $\{d_1, \ldots, d_n\}$ for a given query to identify the top-10 relevant ones. Each inference call to a Large Language Model (LLM) can handle B pairwise comparisons (producing preference probabilities p_{ij}), and we measure total cost by counting these calls. Following we treat each call as a single cost unit, ignoring token usage or monetary factors. We evaluate Flan-T5-L and Flan-T5-XL (Chung et al., 2022) under different batch sizes ($B \in \{1, 2, 5\}$) to explore the trade-off between update frequency and efficiency. We compare sorting methods from Qin et al. (2024); Zhuang et al. (2024), ASAP (Mikhailiuk et al., 2020), and our proposed active ranking approach, analyzing both binary preferences vs. LLM logits, uniform vs. BM25 priors, and varying batch sizes under different cost constraints. Experiments are conducted on multiple BEIR (Thakur et al., 2021) subsets (Webis-Touche2020, NFCorpus, Large-Scifact, TREC-COVID, FiQA, DBpedia-Entity), re-ranking the top 100 BM25-retrieved documents per query. Full methodological details, including our BM25 pre-/post-processing strategies and ablation settings, are in Appendix B.

5 Results

5.1 Comparison to Sorting-Based Methods

We evaluated our partial sorting Thurnstone score algorithm against a binary version of the same algorithm and also traditional sorting-based methods—specifically, Quicksort and Bubblesort, with the adaptation of early stopping when budget is exceeded. In this experiment, we fixed the batch size to 1 to ensure that all methods performed an identical number of pairwise comparisons. As shown in Figure 1, our Thurnstone score algorithm consistently achieves higher *nDCG@10* scores than both sorting-based baselines and binary adapatations of Thurnstone. These findings indicate that our approach is able to extract more informative pairwise comparisons under strict LLM query budgets, leading to superior re-ranking performance.

	Method	Inferences *	Covid	Touche	SciFact	DBPedia	NFCorpus	FiQA
	BM25	–	59.5	44.2	67.9	31.8	32.2	23.6
Flan-T5-L	HeapSort	1328	76.1	44.9	70.0	41.3	33.9	31.4
	BubbleSort	9900	71.4	45.4	69.0	41.6	34.1	29.7
	Allpair	9900	77.0	44.0	70.4	43.8	34.1	33.2
	PRP-Graph-10	1000	76.3	48.6	70.9	44.9	34.7	33.2
	ThurstoneMostellerSortScore	1000	**77.6**	33.6	69.5	43.7	34.5	31.6
	PRP-Graph-20	2000	78.1	49.1	70.4	45.6	35.3	33.7
	ThurstoneMostellerSortScore	2000	77.8	34.0	**71.6**	44.8	34.2	32.7
Flan-T5-XL	HeapSort	1328	77.9	43.9	73.7	41.7	35.2	38.3
	BubbleSort	9900	76.3	44.0	73.4	43.2	35.9	38.3
	Allpair	9900	75.9	44.7	72.2	43.0	35.9	38.2
	PRP-Graph-10	1000	77.0	45.2	74.2	42.8	35.6	36.9
	ThurstoneMostellerSortScore	1000	**77.1**	31.0	70.7	41.1	35.3	36.0

Table 1: Performance comparison of various methods across multiple datasets, with added comparisons column.

* Inferences are measured for sorting 100 documents per query.

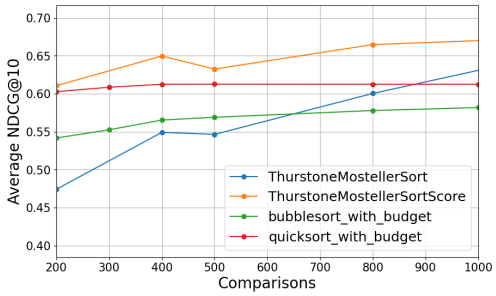

Figure 1: Comparison of the Thurnstone score algorithm with it's binary adaptation and sorting-based methods (Quicksort and Bubblesort) using the same number of comparisons and a batch size of 1.

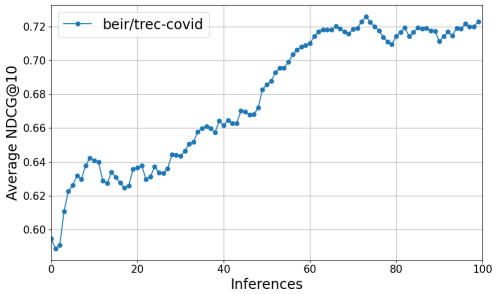

Figure 2: TREC-COVID dataset: nDCG@10 gains for the first 100 LLM comparisons. Our method shows consistent improvements with far fewer than 100 comparisons, whereas PRP-Graph must use 100 comparisons per round.

We also highlight the trend observed in the TREC-COVID dataset (from BEIR) with the first 100 comparisons: there is a nearly linear gain in nDCG@10. In contrast, round-based methods (e.g., PRP-Graph) jump in increments of 100 inferences—missing out on the opportunity to stop earlier while already achieving strong performance. Figure 2 illustrates this effect.

5.2 Batch Size Ablation

To further clarify the impact of batch size B on our re-ranking performance, we conducted two complementary experiments. These experiments elucidate the trade-off between update frequency and the effective number of comparisons, a balance already discussed in our method description.

Fixed Total Comparisons. When the total comparison budget is fixed (e.g., 400 comparisons), increasing the batch size B reduces the number of sequential update iterations (e.g., a batch size of 10 yields only 40 iterations, versus 400 iterations for a batch size of 1). As a result, smaller batches—allowing for more frequent updates—lead to higher *nDCG@10* scores. This indicates that a more sequential approach performs better. Detailed results are provided in Appendix F.

Fixed LLM Inferences. In a separate experiment, we fixed the number of LLM inferences, so that the total number of comparisons scales as $K \times B$ (with K being the number of inferences). Under this condition, larger batch sizes yield a higher effective comparison budget, which translates into improved re-ranking performance. Figure 3 illustrates that, for the Thurnstone-with-score variant, increasing the batch size leads to higher *nDCG@10* when the number of inferences is the limiting factor. (See Appendix D for corresponding results on the Thurnstone binary variant.)

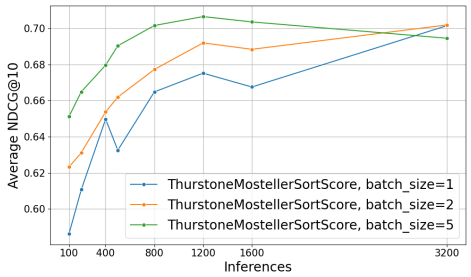

Figure 3: Re-ranking performance (*nDCG@10*) as a function of the number of LLM inferences for different batch sizes for the Thurnstone-with-score variant.

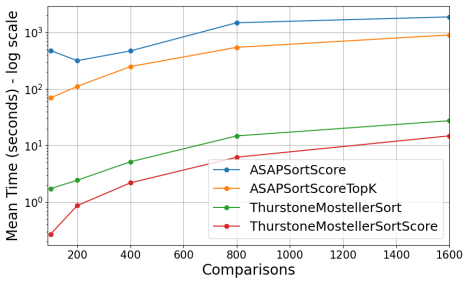

Figure 4: Runtime (on a logarithmic scale) versus the number of comparisons for different algorithms. The ASAP methods (shown in blue and orange) require around 100 times longer than our proposed approach for any given number of comparisons.

Prior Regularization Tuning We performed a sweep over the regularization parameter that balances the BM25 prior with LM-based comparisons. For TREC DL 2019 and TREC DL 2020, our model achieves peak performance at a moderate regularization value (see Appendix G for the full hyperparameter-sweep plot).

Priors relevance and pre/post-processing We evaluated the impact of BM25 priors and our pre-/post-processing techniques under various comparison budgets. Our findings indicate that:

- Using BM25 priors notably improves performance in budget-constrained settings (see Appendix C for detailed comparisons).
- The *RankTransformPrior*, which assigns each document a score in the set $\{1 - k/N \mid k \in \{0, \ldots, N-1\}\}$, outperforms the alternative pre-processing variants.
- A simple weighted interpolation with $\alpha = 0.9$ yields the best nDCG@10 scores.

Detailed tables and figures, including a full ablation study (Tables 2 and 3) and comparison plots (Figure 6), are provided in Appendix C.

5.3 Comparison to ASAP

ASAP and other iterative message-passing approaches aim to minimize the number of pairwise comparisons by adaptively selecting the most informative ones. However, as shown in Figure 4, these techniques require roughly 10–100 times more computation than our proposed approach. This significant runtime increase renders them impractical when LLM query budgets are strict or computational resources are limited. In contrast, our method delivers competitive re-ranking performance while operating at a fraction of the computational cost.

Figure 5 further compares the ranking performance of the ASAP methods with our approach across various comparison budgets. Although the ASAP variants—both the general method and the version optimized for the top-10 items—typically deliver higher ranking accuracy when ample comparisons are available, their high computational demands limit their practical applicability.

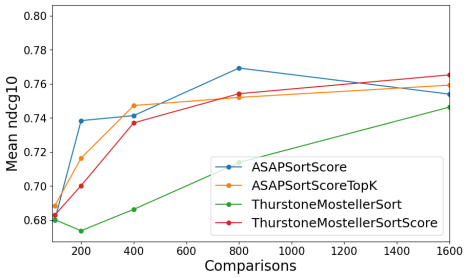

Figure 5: Ranking performance versus the number of comparisons. Both ASAP variants (the general method and the top-10 focused version) tend to achieve higher ranking performance than the Thurstone-based approaches across a range of comparison budgets.

6 Conclusion

This work examined document re-ranking under LLM inference budgets, demonstrating that partial ranking algorithms can significantly outperform traditional sorting methods when comparisons are costly. Our experiments revealed three key insights: first, that leveraging BM25 priors and uncertainty-based pair selection leads to better nDCG@10 scores than deterministic sorting approaches; second, that while ASAP provides superior theoretical guarantees, its computational overhead makes it impractical for real-world applications; and third, that batching strategies offer crucial trade-offs between update frequency and total comparison count.

7 Limitations

While our methods offer practical advantages over classical sorting-based or message-passing-based algorithms, they do carry some limitations that open interesting directions for future work. First, our approach does not provide the strict theoretical guarantees (e.g., exact ordering) afforded by classical sorting or advanced active ranking frameworks, instead offering flexible use of any budget or partial ordering. Moreover, our experiments were conducted with a limited set of batch sizes, and although careful batching strategies were employed, the results may not fully generalize to scenarios requiring extremely large or dynamic batching. An expanded investigation of more complex batching regimes would be beneficial.

In our experiments, we observed that using the logit of the LLM is far more informative than a binary order classification; however, our method does not incorporate a more sophisticated mechanism to exploit this advantage. Specifically, we do not refine pair selection to focus on items most likely to influence top-k results, nor do we directly optimize for nDCG@10 to prioritize costly inferences for the most impactful comparisons. Moreover, the absence of a probability calibration stage for the LLM logit may cause suboptimal performance, as uncalibrated logits might either over- or underestimate the true confidence of comparisons. These factors limit the current approach's ability to fully leverage the LLM's predictive power and highlight the need for more sophisticated selection, optimization, and calibration strategies.

We also acknowledge that our testing was limited to a particular set of language models and benchmark datasets. While these choices are representative, performance in other domains or with differently calibrated LLMs remains an open question. Finally, like most LLM-based approaches, our method may inherit potential biases from the underlying model prompts and responses. Addressing these biases, through systematic prompt engineering or post-hoc adjustments, is an important direction for future work.

References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:1877–1901.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models. *arXiv preprint*.

Marco Dinarelli and Sophie Rosset. 2011. Models cascade for tree-structured named entity detection. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 1269–1278, Chiang Mai, Thailand. Asian Federation of Natural Language Processing.

Ralf Herbrich, Tom Minka, and Thore Graepel. 2006. Trueskill™: A bayesian skill rating system. In *Advances in Neural Information Processing Systems*, volume 19. MIT Press.

Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. *CoRR*, abs/2005.11401.

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 (ACL)*, volume 1: Long Papers, pages 5766–5776, Bangkok, Thailand. Association for Computational Linguistics.

Irina Matveeva, Chris Burges, Timo Burkard, Andy Laucius, and Leon Wong. 2006. High accuracy retrieval with multiple nested ranker. pages 437–444.

Aliaksei Mikhailiuk, Clifford Wilmot, Maria Perez-Ortiz, Dingcheng Yue, and Rafal Mantiuk. 2020. Active sampling for pairwise comparisons via approximate message passing and information gain maximization. *Preprint*, arXiv:2004.05691.

Sergey Nikolenko and Alexander Sirotkin. 2011. A new bayesian rating system for team competitions. pages 601–608.

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.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A heterogenous benchmark for zero-shot evaluation of information retrieval models. *CoRR*, abs/2104.08663.

Louis Leon Thurstone. 1927. A law of comparative judgement. *Psychological Review*, 34:278–286.

Yutao Zhu, Huaying Yuan, Shuting Wang, Jiongnan Liu, Wenhan Liu, Chenlong Deng, Haonan Chen, Zheng Liu, Zhicheng Dou, and Ji-Rong Wen. 2024. Large language models for information retrieval: A survey. *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 2024, page 38–47. ACM.

A Extended Related Work

Traditional IR & Cross-Domain Generalization. Traditional information retrieval (IR) systems often rely on extensive labeled data and struggle with cross-domain generalization (Matveeva et al., 2006; Dinarelli and Rosset, 2011). In contrast, large language models (LLMs) have enabled zero-shot re-ranking capabilities, reducing dependence on domain-specific labeled data.

Pairwise Ranking Process (PRP). The advent of LLMs has led to Pairwise Ranking Process (PRP) (Qin et al., 2024), which compares document pairs through simple prompting, removing the need for fine-tuning or specialized model access. This approach can achieve strong re-ranking performance but incurs a high computational cost because each pairwise comparison requires a separate LLM inference, often scaling quadratically with the number of documents.

Efficient PRP and Cost Models for Re-Ranking. To address PRP's high computational overhead, recent works have explored sorting-based approaches (Qin et al., 2024; Zhuang et al., 2024). Although such methods reduce the number of comparisons in principle LLM inference—rather than the raw number of comparisons—dominates actual runtime costs. Moreover, classical sorting algorithms (e.g., HeapSort, BubbleSort) assume noise-free or minimal-noise comparisons. This assumption is often violated in LLM-based ranking, where outputs can be inconsistent or biased.

Round-Based Re-Ranking Approaches. Luo et al. (2024) propose a method that divides the re-ranking process into a fixed number of rounds, each requiring two inference per each pair of documents. This controlled structure simplifies cost estimation per round and can reduce excessive all-pairs comparisons. However, it does not allow selectively querying only the most uncertain pairs, they rely on matching documents with similar scores, not limiting efficiency when some comparisons are already well-resolved. In contrast, our Bayesian approach enables adaptive decisions on both the number of queries and which pairs to compare, leveraging interpretable priors, likelihoods, and posteriors to guide ranking under noisy conditions.

Retrieval-Augmented Generation (RAG) and LLM-Based Re-Ranking. LLM-based re-ranking is especially valuable in retrieval-

augmented generation (RAG) pipelines (Lewis et al., 2020), in which even small improvements in selecting relevant documents can yield substantial performance gains. While current efforts emphasize effective retrieval (Zhu et al., 2024), computational efficiency remains underexplored. Our cost-aware approach addresses this gap by explicitly considering LLM querying budgets.

Active Sampling and Bayesian Ranking. Bayesian models have been used to handle scenarios where the number of possible comparisons is large, but the system is restricted to only a subset of them. ASAP (Mikhailiuk et al., 2020) is a prime example: it selects pairs via expected information gain, prioritizing those that reduce global uncertainty the most. Although ASAP offers clear statistical advantages, we discuss practical limitations—such as elevated local computation overhead—in LLM-based contexts.

Beyond Sorting: Cost-Aware Statistical Ranking. Our work advances this line of research by proposing a Bayesian-inspired ranking framework that explicitly factors in LLM inference costs. Unlike sorting-based solutions, which rigidly follow comparison rules that can lead to many unnecessary queries, our method adaptively chooses which pairs to compare, integrates BM25 priors, and employs TrueSkill-based uncertainty sampling to maximize each LLM inference. This approach capitalizes on partial knowledge, yielding more robust rankings under strict computational constraints.

B Detailed Experimental Setup

B.1 BM25 Pre-processing Variants

Following Luo et al. (2024), re-ranking methods can benefit from converting raw BM25 outputs into a different numerical scale before using them as priors:

1. **StandardizePrior**: Subtract the mean and divide by the standard deviation (mean 0, std. 1).
2. **MinMaxScalePrior**: Scale scores to the range $[0, 1]$.
3. **RankTransformPrior**: Rank documents by BM25 score and assign values uniformly spaced between 1.0 and 0, ignoring raw score magnitudes.

B.2 Post-processing Score Interpolation

To blend re-ranking outputs S_{final} with BM25 scores S_{bm25}, we use:

$$S_{\text{interp}} = \alpha \cdot S_{\text{final}} + (1 - \alpha) \cdot S_{\text{bm25}},$$

where $\alpha \in [0, 1]$ controls the relative influence of the learned scores versus BM25.

B.3 Experimental Factors

We consider:

- **Binary vs. Logit Preferences**: Direct binary preferences from the LLM versus extracting logits to represent preference strengths.
- **BM25 vs. Uniform Priors**: Initializing the Thurstone model with either BM25-based scores or uniform ones.
- **Batch Size with Fixed Budget**: Varying B under a constant total number of LLM calls (i.e., K calls each handling B comparisons).
- **Batch Size with Fixed Iterations**: Varying B under a fixed K, allowing total comparisons to grow with batch size.

B.4 Datasets and Evaluation

We use the top 100 BM25-retrieved documents from six BEIR datasets (Thakur et al., 2021): Webis-Touche2020, NFCorpus, Large-Scifact, TREC-COVID, FiQA, and DBpedia-Entity. Our goal is to re-rank these to identify the top-10 most relevant ones. All metrics reported are averaged across datasets and LLMs, with standard deviations shown. Individual dataset and LLM breakdowns are provided in the extended results.

C Detailed Results

C.1 Effect of BM25 Priors Across Budgets

Figure 6 compares the standard Thurnstone re-ranker (with BM25 priors) against a variant using non-informative priors. For smaller budgets (around 100 comparisons), the lack of informative priors degrades performance, while the performance gap narrows as the budget increases.

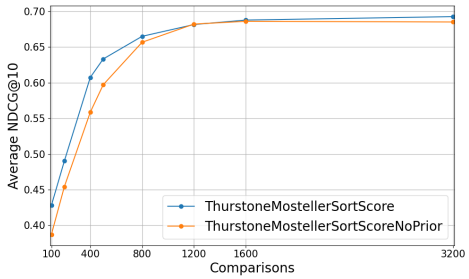

Figure 6: Thurnstone re-ranking with BM25 priors vs. non-informative priors (score logits version).

C.2 Ablation Study: BM25 Pre-processing

Table 2 presents the nDCG@10 results across four datasets (*TREC DL 2019, TREC DL 2020, TREC-COVID, DBpedia*) using Flan-T5-L under 1000 comparisons. The *RankTransformPrior* shows the highest average nDCG@10.

Pre-processing Strategy	nDCG@10 (Mean)
MinMaxScalePrior	60.9
StandardizePrior	60.6
RankTransformPrior	**61.7**

Table 2: BM25 pre-processing strategies comparison over four datasets.

C.3 Ablation Study: Post-processing Interpolation

Table 3 summarizes the performance of various interpolation methods on TREC-COVID with Flan-T5-L. The best performance (77.5 nDCG@10) is achieved by the simple weighted post-processor with $\alpha = 0.9$.

Post-processor	nDCG@10
No Post-processing (Ours)	75.5
HarmonicMeanPostProcessor	68.6
MultiplicativePostProcessor	68.2
RankSumPostProcessor	73.5
SimpleWeightedPostProcessor$_{\alpha=0.5}$	72.8
SimpleWeightedPostProcessor$_{\alpha=0.7}$	76.3
SimpleWeightedPostProcessor$_{\alpha=0.9}$	**77.5**
SumScorePostProcessor	72.3
WeightedMixPostProcessor$_{\alpha=0.7}$	76.2
WeightedMixPostProcessor$_{\alpha=0.9}$	75.7

Table 3: Comparison of interpolation post-processors on TREC-COVID.

D Thurnstone Binary Variant Results

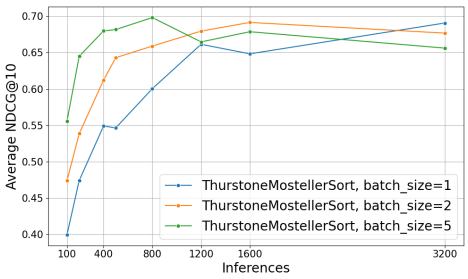

Figure 7: Re-ranking performance (*nDCG@10*) as a function of the number of LLM inferences for different batch sizes for the Thurnstone binary variant.

E Additional Figures for BM25 Priors Comparison (Binary Variant)

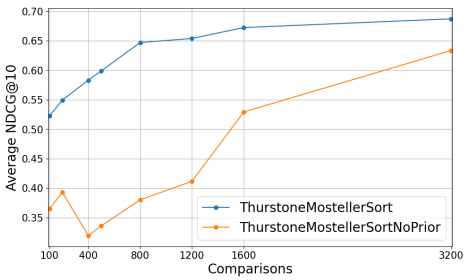

Figure 8: Comparison of Thurnstone-based re-ranking with BM25 priors versus a no-prior variant for the binary version.

F Detailed Results for Fixed Total Comparisons in Batch inference

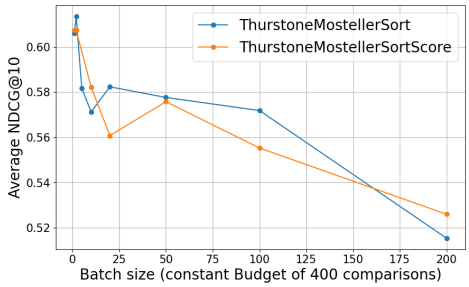

Figure 9: Performance versus batch size when the total comparison budget is held constant. Larger batches reduce the number of update iterations, generally resulting in lower *nDCG@10*.

G Hyperparameter Tuning

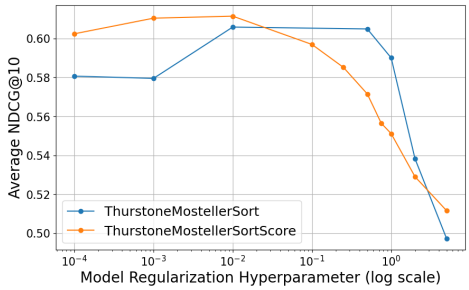

Figure 10: *nDCG@10* performance across different values of the prior regularization parameter. Both Thurnstone variants show an inverted-U relationship, indicating an optimal balance between BM25 priors and LM-based comparisons. We see that the performance of TREC DL 2019 and 2020 peaks at 10^{-2}.

H Supplementary Algorithm Details

For completeness, we note that our method leverages Thurstone's probabilistic framework combined with TrueSkill's uncertainty sampling to specifically target pairs with maximal uncertainty. Unlike ASAP—which uses AMP to jointly update all item posteriors via an MST built on Expected Information Gain—our algorithm processes LLM responses by converting probabilities to z-scores and performing regularized least-squares updates, thereby decoupling the high cost of LLM queries from local computations. Additionally, our query design includes bidirectional (or randomized) comparisons to mitigate positional bias. Although ASAP achieves slightly higher ranking accuracy, its computational overhead is approximately 100 times greater than that of our approach. The full pseudocode and additional experimental implementation details are provided elsewhere in this appendix.

Algorithm 1 Thurstone-Mosteller Sorting Algorithm

Require: Data set D, initial scores S_0, comparator function f, LLM inference budget B, regularization parameter λ

Ensure: Sorted data set D^*

Set $S \leftarrow S_0$

Initialize win counts W and comparison counts C

$comparisons\_made \leftarrow 0$

while $comparisons\_made < B$ **do**

  Compute preliminary probabilities: $P_{ij} \leftarrow \Phi(S_i - S_j)$ for all pairs (i, j)

  Evaluate pairwise uncertainty: $U_{ij} \leftarrow P_{ij}(1 - P_{ij})$

  Select the pairs with highest U_{ij} to request from LLM in a batch

  Execute the batch comparison via $f(q, d_i, d_j)$ on the selected pairs

  Update W and C with the new wins and total comparisons

  Estimate empirical probabilities from updated counts: $p_{ij} = W_{ij}/C_{ij}$

  Compute $z_{ij} \leftarrow \Phi^{-1}(p_{ij})$

  Formulate the regularized least-squares system:

$$(A^T A + \lambda I)\, x = A^T\, b,$$

  where b encodes the z_{ij} values and A encodes pairwise relationships

  Solve for x and update S accordingly

  $comparisons\_made \leftarrow$ updated count of total comparisons

end while

return D^* sorted according to the final scores S