REALM: Recursive Relevance Modeling for LLM-based Document Re-Ranking

Anonymous ACL submission

Abstract

Large Language Models (LLMs) have shown strong capabilities in document re-ranking, a key component in modern Information Retrieval (IR) systems. However, existing LLM-based approaches face notable limitations, including ranking uncertainty, unstable top-k recovery, and high token cost due to token-intensive prompting. To effectively address these limitations, we propose REALM, an uncertainty-aware re-ranking framework that models LLM-derived relevance as Gaussian distributions and refines them through recursive Bayesian updates. By explicitly capturing uncertainty and minimizing redundant queries, REALM achieves better rankings more efficiently. Experimental results demonstrate that our REALM surpasses state-of-the-art re-rankers while significantly reducing token usage and latency, promoting it as the next generation re-ranker for modern IR systems.

1 Introduction

Document re-ranking is a key component in modern IR systems (Zhu et al., 2024). Given a user query, retrieval systems typically begin with a fast but coarse retrieval stage that returns a broad set of potentially relevant documents. However, these initial results are often noisy or only loosely related to the query. Re-ranking addresses this issue by applying a more accurate, context-aware scoring model to refine the order of the candidates and place the most relevant documents at the top (Nogueira and Cho, 2020). For example, in academic paper searching, an initial retrieval step may return a large number of documents that match surface-level keywords, while missing more in-depth relevance analysis. Without re-ranking, users could suffer from missing important references, thus struggling to construct a complete picture of the research landscape. Recent work (Fukuda and Tomiura, 2025) shows that integrating multiple retrieval methods, particularly re-ranking the combined results, significantly improves recall, helping users retrieve more comprehensive and relevant literature within a manageable number of results. In a nutshell, re-ranking is crucial to ensure that high-quality, contextually appropriate documents are selected as input for subsequent applications (Lewis et al., 2020).

LLMs are redefining document re-ranking by enabling deep semantic and contextual understanding that traditional lexical-based methods fundamentally lack. Traditional re-rankers, such as those based on BM25 scores (Robertson et al., 2009) or learning-to-rank models like LambdaMART (Burges, 2010), rely heavily on sparse features-term overlap, document frequency, or hand-crafted heuristics-which often fail in capturing nuanced relevance, especially in complex or ambiguous queries. In contrast, LLM-based re-rankers treat the query and candidate documents as joint inputs, allowing for fine-grained relevance estimation grounded in deep semantic and contextual comprehension. Recent studies have shown that LLMs, when applied as cross-encoders or guided with task-specific prompting, consistently outperform classical re-rankers across benchmarks (Sun et al., 2023). These advancements suggest that LLMs are not just an incremental improvement but a paradigm shift toward unifying retrieval and comprehension within a single, adaptable framework.

However, LLM-based document re-ranking faces a three-pronged challenge: (i) **ranking uncertainty**, stemming from the inherent stochastic nature of LLMs (see Section 2.2); (ii) **unstable top-k recovery**, where minor input variations can substantially disrupt document rankings (see Figure 2); and (iii) **high token costs**, due to the need of complex prompting strategy (see Table 1).

Recent research endeavors have fallen short in effectively addressing all three challenges: **Pointwise methods** (Nogueira et al., 2020) are efficient and parallelizable, as they assess each document in-

dependently. Some variants (Zhuang et al., 2023b) further leverage generation likelihood as a relevance score. However, pointwise approaches fail to model interactions among candidates, making them less effective at resolving uncertainty or producing globally consistent top-k rankings. **Listwise methods** (Sun et al., 2023) enable joint evaluation of multiple candidates in a single query, which helps mitigate ranking inconsistency. Approaches like TourRank (Chen et al., 2025) adopt tournament-style aggregation to extend listwise scoring. Despite these benefits, listwise methods still suffer from context length constraints and positional bias (Liu et al., 2024b), especially for long candidate sets. **Pairwise methods** (Qin et al., 2024) improve local comparison quality by directly modeling relative preferences between document pairs. Advanced systems like PRP-Graph (Luo et al., 2024) further exploit graph structures to aggregate pairwise signals. Nevertheless, the repeated comparisons, however, lead to high token usage and substantial inference latency. **Setwise methods** (Zhuang et al., 2024; Podolak et al., 2025) improve efficiency by evaluating small subsets at a time, but discard fine-grained preference information, such as full relevance logits—thereby underutilizing the model's capacity.

To effectively address the three-pronged challenge, this paper proposes REALM, an uncertainty-aware re-ranking framework that combines relevance estimation with a recursive refinement process. REALM explicitly models uncertainty, improves top-k stability, and reduces inference costs. Our contributions are as follows:

- **Uncertainty-Aware Relevance Modeling.** We model each document's relevance as a Gaussian distribution, capturing both the estimated score and uncertainty to support robust re-ranking under the inherent stochastic nature of contemporary LLMs.
- **Recursive Refinement Framework.** We introduce a recursive framework that compares pivot documents with subsets and refines relevance distributions through Bayesian updates, enhancing ranking stability.
- **Pivot-Centric Optimizations.** We optimize efficiency by selecting high-confidence pivots, aggregating updates via uncertainty-aware averaging, and applying pivot adjustment to ensure effective workload reduction.

Experiments show that REALM outperforms state-of-the-art re-ranking methods while substantially reducing token usage and improving stability, making it suitable for real-world retrieval systems.

2 Related Work & Preliminary

2.1 Related Work

Zero-shot document re-ranking with LLMs is typically grounded in four fundamental prompting strategies: pointwise (Nogueira et al., 2020), pairwise (Qin et al., 2024), listwise (Sun et al., 2023), and setwise (Zhuang et al., 2024). In this section, we compare REALM with these papers.

Pointwise methods. Pointwise methods prompt LLMs to assess the relevance of each document independently with respect to a given query, typically by generating a relevance score or extracting the score from the output logits (Nogueira et al., 2020). Several variants exist. For example, Query Generation (Zhuang et al., 2023b) estimates query-document compatibility by computing the likelihood of the query given a passage. While these approaches are token efficient and scalable, they struggle to capture comparative relevance across candidates, which is well preserved in REALM.

Listwise methods. With the continued expansion of LLM capacity and input window size, listwise ranking—where the model receives a group of candidate documents and directly outputs their relative ordering—has become increasingly feasible. By supporting joint reasoning over multiple candidates within a single inference, this paradigm has motivated a series of methods aimed at better leveraging LLM capabilities for document re-ranking.

RankGPT (Sun et al., 2023) and LRL (Ma et al., 2023) adopt a sliding-window listwise re-ranking strategy, comparing a subset of candidates at each step, retaining the most relevant ones, and forwarding the rest for subsequent comparisons. TourRank (Chen et al., 2025) draws inspiration from sports tournaments, treating each subset as a group match and aggregating results through a point-based system. ListT5 (Yoon et al., 2024) follows a similar tournament-style design, effectively implementing an m-ary heap traversal over listwise scoring primitives to recover the top-k results.

Despite their effectiveness, these methods face inherent limitations. Current LLMs are still restricted by finite context lengths and remain sensitive to positional biases (Liu et al., 2024b), which hinders their ability to process long candidate lists

holistically and maintain consistency across multiple comparisons. In contrast, REALM avoids full-list comparisons by decomposing the ranking process into a sequence of setwise updates. This approach enables consistent top-k selection without suffering from context-length limitations or positional bias, while still leveraging the LLM's capacity to reason over small candidate sets.

Pairwise methods. Pairwise prompting (PRP) (Qin et al., 2024) was introduced to overcome the limitations of pointwise and listwise ranking by prompting the model to compare two candidates at a time and choose the more relevant one. To extend this into a full ranking, the authors implemented a multi-round bubble sort using overlapping comparisons to extract the top-k candidates. PRP-Graph(Luo et al., 2024) further generalizes this idea by constructing a weighted comparison graph and applying a PageRank-style aggregation to derive a global ranking.

While pairwise prompting yields accurate comparisons, it incurs high token costs due to repeated queries. In contrast, our method reduces the number of LLM calls by performing aggregation over setwise comparison, achieving greater efficiency without sacrificing ranking quality.

Setwise methods. Setwise prompting (Zhuang et al., 2024) was introduced as a refined variant of listwise prompting, leveraging model output logits to select the top-k documents within a group. This strategy was extended by integrating it with classic sorting algorithms such as bubble sort and heap sort. Setwise Insertion (Podolak et al., 2025) further advanced this line of work by incorporating the initial document ranking as prior knowledge, thereby improving ranking efficiency.

While drawing inspiration from setwise prompting, REALM preserves the full comparative information encoded in logits and performs uncertainty-aware updates through probabilistic aggregation, enabling more robust relevance estimation.

Other directions in LLM for re-ranking. *(i)* Training strategies for LLM-based re-rankers. RankT5 (Zhuang et al., 2023a) adopts pairwise and listwise training objectives for T5, while ChainRank-DPO (Liu et al., 2024a) enhances ranking consistency using CoT-style supervision with DPO. Rank-R1 (Zhuang et al., 2025) introduces reinforcement learning with limited supervision to promote reasoning over queries and documents. RankGPT (Sun et al., 2023) and RankVicuna (Pradeep et al., 2023) distill ChatGPT/GPT-3.5 into smaller models via pairwise and listwise losses. ListT5 (Yoon et al., 2024) uses Fusion-in-Decoder for listwise inference, and TSARankLLM (Zhang et al., 2024) adopts a two-stage pre-training and fine-tuning strategy.

(ii) Hybrid architectures. Hybrid methods restructure the inference process by combining ranking components or decomposing tasks, e.g., EcoRank (Rashid et al., 2024) and RankFlow (Jin et al., 2025). In parallel, Permutation Self-Consistency (Tang et al., 2024) aggregates multiple permutations to reduce positional bias, and LLM-RankFusion (Zeng et al., 2024) improves robustness via calibration and fusion-based aggregation.

2.2 LLM Uncertainty & Bayesian Rating System

LLM uncertainty. LLMs exhibit two forms of uncertainty: aleatoric, which stems from inherent data noise, and epistemic, which arises due to limited training coverage (Kendall and Gal, 2017). While aleatoric uncertainty is largely irreducible, epistemic uncertainty can be mitigated by introducing additional informative signals during inference.

In this context, studies have shown that in multiple-choice settings, LLM output logits are often well-calibrated—i.e., their relative magnitudes reliably reflect the model's confidence (Kadavath et al., 2022; Si et al., 2023). This calibration enables softmax-normalized logits to serve as meaningful probability estimates, supporting downstream applications such as uncertainty estimation and probabilistic relevance modeling.

Bayesian rating systems offer principled probabilistic frameworks for estimating latent skill levels or quality scores based on observed outcomes from comparisons or matches. Grounded in Bayesian inference, these systems iteratively update skill estimates by combining prior beliefs with new evidence. Key advantages include explicit uncertainty modeling, incremental update capabilities, and robustness to noisy or incomplete data. Two widely adopted instances are the Elo rating system (ELO, 1978) and its more expressive successor, TrueSkill (Herbrich et al., 2006), which extend the rating process to handle more complex scenarios, which we detail in the Appendix A.

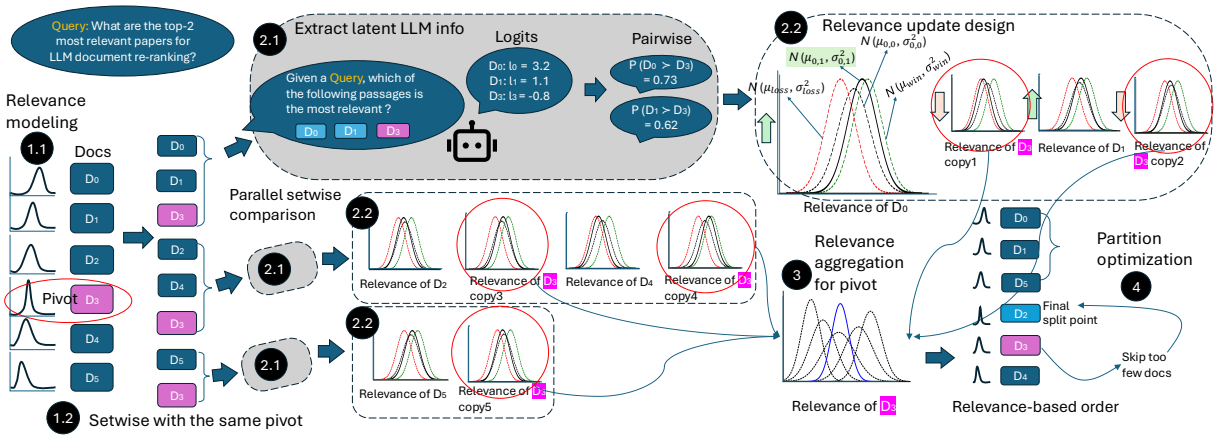

Figure 1: Workflow of our recursive relevance modeling framework for LLM-based document re-ranking.

3 Methodology

3.1 REALM Framework

Figure 1 illustrates the workflow of our relevance modeling framework for LLM-based document re-ranking. As an illustrative example, the task is to retrieve the top-2 most relevant documents on the topic of LLM-based re-ranking. Given a set of documents, we model each document's relevance as a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ (step 1.1).

We begin by selecting a pivot D_3 to put the documents into multiple subsets. For each subset, we conduct a setwise comparison involving the same pivot, as shown in step 2.1 in Figure 1. Then, as depicted in step 2.2, we adopt a Bayesian update to refine the relevance distributions. Further details are presented in § 3.2. Further, as shown in step 3, we introduce a mechanism to update document D_3's relevance model.

A naive subsequent design of REALM would directly rely on the relevance distributions of the first iteration to order all the documents. Subsequently, we can select the top-k most relevant documents as the final result. Particularly, we can derive the relevance score of each document using a distribution-based rule $\mu - k\sigma$, where k is a constant controlling conservativeness, with higher values penalizing uncertainty more heavily. However, this design could potentially rely on relevance distributions that are very unstable, as a single round of comparison might fail to effectively curb the uncertainty of the relevance distributions (see Table 2).

Consequently, we introduce a recursive design, that is, we compare each document against the pivot document. If a document is closer to the query than the pivot, we keep it for the next round of calculation. Otherwise, we filter out that document. Moreover, directly relying on the pivot D_3 to filter out unpromising documents would yield unstable workload reduction. We thus design an effective workload reduction mechanism to cope with this concern, which could derive D_4 as the final split point (4). Of note, we also design strategy to select the document with the highest confidence as the pivot (step 1.2).

3.2 REALM's Relevance Modeling Scheme

Modeling relevance as a normal distribution. In REALM, relevance judgment derived from LLMs is inherently noisy due to their contextual sensitivity, response stochasticity, and inherent biases (Dai et al., 2024). To capture this uncertainty, following TrueSkill (Herbrich et al., 2006), a framework originally designed for competitive player rating, we model the relevance of each document to a given query as a Gaussian distribution, which is denoted as $\mathcal{N}(\mu, \sigma^2)$, where the mean μ represents the estimated relevance score and the variance σ^2 quantifies the uncertainty of this estimate.

As illustrated in step 1.1 of Figure 1, each document is initially associated with an *initial relevance distribution* (i.e., the dark blue squiggle lines beside each document D_i). We model the relevance as a Gaussian distribution with fixed standard deviation σ_0, reflecting a uniform level of uncertainty across all unobserved documents. The initial mean μ_0 is set based on the document's retrieval score if available (e.g., from a pre-LLM retrieval pipeline such as embedding search); otherwise, a shared default value is assigned.

Extracting latent LLM information for relevance model update. To refine these initial relevance distributions, REALM extracts latent information from LLMs during setwise comparison.

In setwise comparison, LLM is prompted to select the most relevant document from a small group (of size m) of candidate documents. Our example query is "Given a query, which of the following passages is the most relevant?". Internally, LLM assigns scalar logits to each option; we extract these as a score vector $\{\ell_i \colon D_i \in \text{set}\}$, where each ℓ_i corresponds to the document D_i. For instance, in step 2.1 of Figure 1, the logits assigned to (D_0, D_1, D_3) are 3.2, 1.1, and -0.8, respectively. Of note, for closed-source LLMs, we can prompt them to output the confidence values and use that as the scores (Xia et al., 2025).

Rather than relying on these scores for direct selection or ranking (Zhuang et al., 2024), REALM interprets their differences as pairwise preference probabilities. Specifically, the probability that document D_i is preferred over D_j is computed as:

$$P(D_i \succ D_j) = \sigma\left(\frac{\ell_i - \ell_j}{T}\right),$$

where σ is the sigmoid function and T is a temperature parameter. This formulation transforms a single multi-document comparison into a set of $\binom{m}{2}$ pairwise probabilities updates.

To ensure consistency in aggregation, REALM adopts a pivot-based strategy. One document in the prompt is designated as the pivot, and preference probabilities are computed between the pivot and the others. As illustrated in step 2.1 of Figure 1, when D_3 is used as the pivot alongside D_0 and D_1, we extract the probabilities $P(D_0 \succ D_3)$ and $P(D_1 \succ D_3)$. These preference probabilities are then used to update the relevance distributions of both the pivot and its comparators as follows.

Relevance update design. The extracted pairwise preference probabilities, such as $P(D_0 \succ D_3)$ and $P(D_1 \succ D_3)$ from step 2.1, are then used to update each document's relevance distribution.

Rather than treating each update as a deterministic outcome (win, loss, or draw) as in the original TrueSkill, we leverage the preference probability to capture richer information in relevance updates. Specifically, we interpolate the influence of the win and loss outcomes based on the predicted probability, i.e., $P(D_i \succ D_j)$.

As illustrated in step 2.2, the model predicts $P(D_0 \succ D_3) = 0.73$. In this case, the relevance distribution of D_0, i.e., $\mathcal{N}(\mu_{0,0}, \sigma^2_{0,0})$, is updated to $\mathcal{N}(\mu_{0,1}, \sigma^2_{0,1})$ by interpolating between the two TrueSkill-updated distributions—corresponding to a win $\mathcal{N}(\mu_{win}, \sigma^2_{win})$ (green dashed curve) and a loss $\mathcal{N}(\mu_{loss}, \sigma^2_{loss})$ (red dashed curve) against D_3—weighted by the predicted probability.

The resulting distribution for D_0 thus shifts toward the win-specific distribution while incorporating uncertainty, effectively reflecting both the model's directional preference and its confidence.

Formally, let the distribution of document D_i be $\mathcal{N}(\mu_{i,0}, \sigma^2_{i,0})$, with natural parameters defined as the precision $\lambda_{i,0} = 1/\sigma^2_{i,0}$ and the precision-adjusted mean $\tau_{i,0} = \mu_{i,0}/\sigma^2_{i,0}$. We compute two updated distributions for D_i by applying the standard TrueSkill update rules for a 1v1 match against $D_j \sim \mathcal{N}(\mu_{j,0}, \sigma^2_{j,0})$, as follows:

- $\mathcal{N}\left(\mu_{\text{win}}, \sigma^2_{\text{win}}\right)$, assuming D_i wins over D_j,
- $\mathcal{N}\left(\mu_{\text{loss}}, \sigma^2_{\text{loss}}\right)$, assuming D_i loses to D_j.

Let $\tau_{\text{win}} = \mu_{\text{win}}/\sigma^2_{\text{win}}$ and $\lambda_{\text{win}} = 1/\sigma^2_{\text{win}}$, and similarly for the loss outcome. Given a win probability $p = P(D_i \succ D_j)$, we apply a fractional update (Minka, 2004) in the natural parameter space by combining the additive changes from the win/loss outcomes:

$$\lambda_{i,1} = \lambda_{i,0} + p \cdot (\lambda_{\text{win}} - \lambda_{i,0}) + (1-p) \cdot (\lambda_{\text{loss}} - \lambda_{i,0}),$$

$$\tau_{i,1} = \tau_{i,0} + p \cdot (\tau_{\text{win}} - \tau_{i,0}) + (1-p) \cdot (\tau_{\text{loss}} - \tau_{i,0}),$$

The resulting distribution of D_i is again a Gaussian, given by:

$$\mathcal{N}\left(\mu_{i,1}, \sigma^2_{i,1}\right), \text{ where } \mu_{i,1} = \frac{\tau_{i,1}}{\lambda_{i,1}},\ \sigma^2_{i,1} = \frac{1}{\lambda_{i,1}}.$$

This relevance update enables our model to integrate both the direction and confidence of each comparison. Further details are provided in Appendix A.3. As shown in step 2.2 of Figure 1, *the updated distribution becomes narrower, indicating increased certainty and shifts toward the more likely outcome*, resulting in a more accurate and confident relevance distribution.

3.3 Pivot-Centric Optimizations

Pivot aggregation. The previous pivot-based strategy enables the extraction of useful pairwise comparisons centered on each pivot. To integrate global relevance signals for a given pivot, we introduce *pivot aggregation* that consolidates its comparison outcomes across all subsets.

After the current round, we combine the relevance models of various copies for the pivot via an

uncertainty-aware averaging:

$$\tau = \sum_{i=1}^{c} \sigma_i^{-2},$$
$$\mu_{\text{agg}} = \frac{\sum_{i=1}^{c} \mu_i \, \sigma_i^{-2}}{\tau},$$
$$\sigma_{\text{agg}} = \left(\frac{\tau}{n}\right)^{-1/2}.$$

Here, τ denotes the total precision accumulated from c shadow comparisons, and the averaging yields an aggregated distribution that favors more confident estimation while reducing overall uncertainty. As shown in step ③ of Figure 1, the pivot D_3 is replicated into five copies and updated independently through soft comparisons. The final distribution of D_3 is then obtained by aggregating these updates, resulting in a more stable and unbiased estimation.

Pivot selection. To determine the global pivot at each recursive step, we select the document with the lowest estimated standard deviation σ from the current candidate pool. Intuitively, a lower σ indicates higher confidence in the document's relevance estimate. Using such a document as the pivot improves the stability of the partitioning process. As illustrated in step 1.2 of Figure 1, document D_3 is selected as the global pivot due to its lowest uncertainty. The remaining documents are then grouped into prompts: $\{D_0, D_1\}$, $\{D_2, D_4\}$, and $\{D_5\}$, ensuring that each non-pivot document is compared once against D_3. This structure enables us to construct a globally consistent relevance preference centered on a high-confidence document.

Pivot adjustment for effective reduction. Considering that pivot might not be able to effectively reduce the documents, we interpolate the pivot with the interval midpoint:

$$i^* = \lambda \, r_p + (1 - \lambda) \, \frac{l + r}{2}, \quad \lambda \in [0, 1],$$

where l and r are the current interval bounds. Using i^* as the split index helps avoid unbalanced partitions. Setting $\lambda < 1$ guarantees recursion depth remains bounded by $O(\log n)$ and total LLM comparisons scale linearly with n. While the pivot still guides comparisons, the softened partition is used only to improve efficiency and does not change the underlying ranking based on the pivot. As illustrated in step ④ of Figure 1, only documents ranked above the split point (e.g., D_3) are retained for the next iteration of refinement. This iteration continues until the desired top-k set is extracted.

4 Experiments

4.1 Experimental Setup

We conduct evaluations on Flan-T5 models (Longpre et al., 2023) of three sizes—Flan-T5-Large (770M parameters), Flan-T5-XL (3B), and Flan-T5-XXL (11B)—following recent work on LLM-based re-ranking (Qin et al., 2024; Luo et al., 2024; Zhuang et al., 2024; Podolak et al., 2025). To assess the generality of REALM, we additionally evaluate Flan-UL2 (20B) (Tay et al., 2023) and LLaMA3 (8B and 70B) (Grattafiori et al., 2024). Notably, LLaMA3 follows a decoder-only architecture and differs from Flan-T5 models in both model structure and pretraining objectives.

All experiments are conducted on a server with 512 GB RAM, two Intel Xeon Silver 4309Y CPUs (16 cores), and four A100 GPUs (80 GB each). All models are evaluated on a single GPU, except for LLaMA3-70B, which uses all four GPUs.

Datasets and metrics. Experiments are conducted on two widely used benchmarks: TREC Deep Learning 2019 (Craswell et al., 2020) and 2020 (Craswell et al., 2021). All LLM-based methods re-rank the top 100 documents retrieved by a BM25 first-stage retriever. Adopting prior work's evaluation strategy (Zhuang et al., 2024; Podolak et al., 2025), we formulate re-ranking as a top-k task, with $k = 10$ as the default setting. Effectiveness is measured using the NDCG@10 metric for all datasets. For clarity, all NDCG@10 results are presented as percentages

Baselines. We compare our method with four recent LLM-based re-ranking approaches: TourRank (Chen et al., 2025), PRP-Graph (Luo et al., 2024), Setwise-Heapsort (Zhuang et al., 2024), and Setwise-Insertion (Podolak et al., 2025). Implementation details are provided in Appendix B.

TourRank is a state-of-the-art listwise re-ranking method inspired by sports tournaments. It treats each subset of documents as a "group match" and aggregates the results using a point-based system. PRP-Graph constructs a global ranking by aggregating local pairwise preferences through a graph-based approach. Setwise-Heapsort and Setwise-Insertion utilize setwise prompting to compare multiple candidates jointly in a token-efficient way. The former focuses on computational efficiency using a heap-based sorting strategy, while the latter improves ranking accuracy through a more refined insertion-based sorting mechanism.

LLM	Method	TREC DL 2019					TREC DL 2020				
		N@10	#Inf.	P. tks.	G. tks.	Lat.(s)	N@10	#Inf.	P. tks.	G. tks.	Lat.(s)
NA	BM25	50.6	-	-	-	-	48.0	-	-	-	-
Flan-T5-Large	TourRank	48.2	130.0	95271.4	1507.2	56.9	40.7	130.0	95341.8	1524.5	57.1
	PRP-Graph	65.8	492.7	221781.9	-	43.3	61.8	492.5	224605.5	-	42.4
	Setwise-Heapsort	66.9	125.3	40449.6	626.5	8.8	61.8	124.2	40357.4	621.0	8.7
	Setwise-Insertion	66.9	92.5	29913.1	93.4	4.5	62.5	91.3	29757.7	93.8	4.4
	REALM	**67.0**	**79.0**	**25165.7**	-	**3.9**	**63.0**	**74.6**	**23584.9**	-	**3.6**
Flan-T5-XL	TourRank	64.3	130.0	95257.0	2719.2	96.8	59.7	130.0	95277.6	2791.0	100.6
	PRP-Graph	67.6	492.6	212884.5	-	43.0	66.1	492.5	216071.3	-	43.1
	Setwise-Heapsort	69.2	129.5	41665.7	647.7	10.1	67.8	127.8	41569.1	639.1	9.6
	Setwise-Insertion	69.0	106.0	34732.7	100.7	5.3	67.0	105.3	34400.7	99.5	5.1
	REALM	**70.5**	**80.4**	**25823.0**	-	**4.0**	**68.4**	**75.6**	**24033.6**	-	**3.7**
Flan-T5-XXL	TourRank	61.9	130.0	95269.3	1610.4	133.6	62.7	130.0	95273.6	1615.6	133.2
	PRP-Graph	66.6	492.6	213536.2	-	73.3	66.1	492.6	216332.4	-	74.4
	Setwise-Heapsort	70.6	130.1	42078.6	650.5	15.9	68.8	128.2	41633.7	640.8	15.7
	Setwise-Insertion	68.4	104.9	34284.2	99.2	10.6	67.1	100.7	33036.9	100.0	10.2
	REALM	**71.2**	**76.5**	**24659.8**	-	**7.5**	**69.1**	**74.1**	**23759.6**	-	**7.3**

Table 1: Evaluation on TREC DL 2019 and TREC DCL 2020 datasets: REALM vs TourRank (Chen et al., 2025), PRP-Graph (Luo et al., 2024), Setwise-Heapsort (Zhuang et al., 2024), and Setwise-Insertion (Podolak et al., 2025).

4.2 Overall Evaluation

Table 1 presents a comprehensive comparison of our projects on both the TREC-DL 2019 and 2020 benchmarks. We report NDCG@10 (N@10), Inf. (inference counts, a.k.a., the # of LLM calls), P. tks. (#tokens in prompt), G. tks. (# of generated tokens), and Lat. (latency in seconds).

Compared to TourRank, PRP-Graph, Setwise-Heapsort, and Setwise-Insertion, REALM achieves consistent improvements in both ranking quality and inference efficiency across all evaluated benchmarks. On average, REALM outperforms all baselines, improving NDCG@10 by $0.7 - 11.9$, while simultaneously reducing the number of LLM inferences by $23.4 - 84.4\%$ and cutting inference latency by $25.0 - 88.7\%$. In terms of prompt token usage (P. tks.), REALM reduces the cost by $25.2 - 94.8\%$. Furthermore, since it only leverages the logits of the first generated token, the generation token cost (G. tks.) is effectively eliminated.

On the model size dimension, we observe that Flan-T5-XL outperforms Flan-T5-Large, and Flan-T5-XXL further surpasses Flan-T5-XL, aligning with the general trend that larger instruction-tuned models exhibit stronger ranking capabilities. Among all methods, TourRank demonstrates the highest sensitivity to model capacity, as its listwise comparison approach relies heavily on both the model's input context length and its vulnerability to positional biases.

While PRP-Graph and TourRank consume the most tokens, this is mainly due to their reliance on multiple iterative rounds of pairwise or listwise comparisons to accumulate sufficient preference information, resulting in significantly higher total query costs. Setwise-Heapsort and Setwise-Insertion offer a more favorable efficiency-performance trade-off by utilizing structured comparisons with fewer rounds. However, they still underutilize the rich preference information embedded in the LLM's output, such as the model's confidence in its ranking decisions. This leaves room for further enhancement of REALM by incorporating more principled information aggregation strategies and refined comparison scheduling to fully leverage the LLM's capacity.

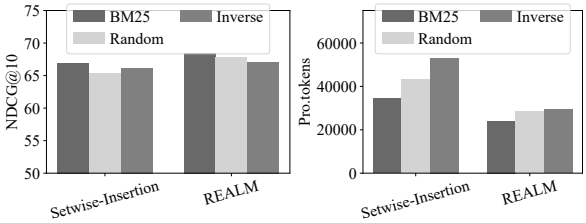

Figure 2: Sensitivity to initial ranking order on TREC DL 2020 using Flan-T5-XL.

REALM vs Setwise-Insertion on the initial ranking. Among existing baselines, Setwise-Insertion offers a reasonable balance between performance and efficiency, utilizing structured comparisons to reduce the number of LLM calls while maintaining competitive ranking quality. However, its effectiveness can still be affected by the quality of the initial document ordering.

Figure 2 compares Setwise-Insertion and

LLM	Method	Avg. Performance on TREC DL			
		N@10	#Inf.	P. tks	Lat.(s)
NA	BM25	49.3	-	-	-
Flan-T5-Large	w/o modeling	62.0	109.2	35064.2	5.4
	w/o recursive	63.0	50.0	16089.0	2.6
	w/o opt.	64.2	112.5	36335.6	5.6
	REALM	**65.0**	76.8	24375.3	3.8
Flan-T5-XL	w/o modeling	66.2	114.9	37003.4	5.8
	w/o recursive	68.3	50.0	16089.0	2.6
	w/o opt.	68.9	118.6	38409.6	6.0
	REALM	**69.5**	78.0	24928.3	3.9
Flan-T5-XXL	w/o modeling	68.7	113.7	36696.1	11.5
	w/o recursive	68.6	50.0	16089.0	5.0
	w/o opt.	68.9	119.2	38846.4	12.1
	REALM	**70.2**	75.3	24209.7	7.4

Table 2: Ablation study.

REALM on the TREC DL 2020 dataset under three initial document orderings: BM25, Inverse, and Random. The left plot reports NDCG@10, while the right plot shows the corresponding prompt token usage. Here, Inverse refers to reversing the original BM25 ranking (i.e., least relevant documents placed first), while Random denotes a random permutation of the BM25-ranked list.

In terms of NDCG, the two methods perform comparably: Setwise-Insertion's best and worst scores differ by 1.6 points, while REALM shows a smaller gap of 1.3 points, indicating slightly better stability. However, the difference becomes more pronounced when comparing token efficiency. Because Setwise-Insertion relies more heavily on the assumptions of the initial ranking, it requires significantly more insertion operations when the initial order is suboptimal (e.g., under Inverse). This leads to substantially higher prompt token usage, whereas REALM is less affected by the quality of the initial ranking and maintains consistently low token consumption across all input orders.

4.3 Analysis

Ablation study. Table 2 presents the results of ablation study, comparing the full REALM system with three reduced variants by disabling key components: (1) W/O MODELING, which removes uncertainty modeling and uses QuickSelect (Hoare, 1961) to retrieve top-k; (2) W/O RECURSIVE, which disables recursive refinement; and (3) W/O OPTIMIZATION, omitting pivot optimization.

Removing any of these components leads to a consistent drop in performance. Disabling uncertainty modeling (W/O MODELING) results in a $1.5-3.3$ decrease in NDCG@10 across all models, highlighting the value of Gaussian-based relevance modeling. The absence of recursive reasoning (W/O RECURSIVE) also causes noticeable degradation, underscoring the benefit of multi-round refinement. Lastly, disabling pivot-centric optimization (W/O OPTIMIZATION) nearly doubles latency—for example, from 7.4s to 12.1s with Flan-T5-XXL—confirming that our pivot selection and partitioning strategy substantially improves efficiency without compromising effectiveness.

LLM (Size)	TREC DL 2019		TREC DL 2020	
	N@10	lat.(s)	N@10	lat.(s)
NA	50.6	-	48.0	-
Flan-T5-Large (770M)	67.0	3.9	63.0	3.6
Flan-T5-XL (3B)	70.5	4.0	68.4	3.7
LLaMA3 (8B)	49.6	30.9	43.6	27.3
Flan-T5-XXL (11B)	71.2	7.5	69.1	7.3
Flan-UL2 (20B)	**72.2**	13.2	**71.4**	12.8
LLaMA3 (70B)	72.0	81.4	68.4	80.2

Table 3: Performance across different LLMs.

Performance across different LLMs. As shown in Table 3, our method benefits from stronger LLMs, achieving higher re-ranking performance and reduced prompt usage. For instance, Flan-UL2 achieves the best results on TREC DL datasets with an average NDCG@10 of 71.8, while Flan-T5-XXL and Flan-T5-XL reach 70.2 and 69.5, respectively, surpassing the no-LLM baseline (48.0) and LLaMA models by a large margin.

In contrast, decoder-only architectures such as LLaMA-8B yield substantially lower performance (e.g., 49.6 on TREC DL 2019), despite incurring 30.9s per query. This discrepancy is partly due to their limited formatting capabilities and strong positional bias. In particular, we observe that LLaMA-8B selects the first option in approximately 85.6% of cases on our balanced binary-choice tasks constructed from the TREC DL 2019 dataset (see Appendix C.1), indicating a strong positional bias that undermines its effectiveness for re-ranking.

5 Conclusion

We present REALM, an uncertainty-aware re-ranking framework. By modeling document relevance as Gaussian distributions and refining them through recursive comparisons and Bayesian aggregation, REALM achieves both high effectiveness and efficiency. Experiments across multiple LLMs and TREC benchmarks demonstrate that REALM consistently outperforms existing re-ranking methods across various configurations.

Limitations

Our current design choices are partially constrained by resource and space limitations. While our relevance modeling is, in principle, compatible with a broad range of re-ranking methods, we center our evaluation on the specific framework we designed that achieved the strongest empirical performance. A more comprehensive comparison with alternative designs is left to future work. Secondly, we conduct experiments using open-source models, including the Flan-T5 and LLaMA 3 series. We did not include closed-source models such as GPT-3.5 or GPT-4 accessed via API. Future extensions may consider such models to provide a more complete empirical picture of REALM.

References

Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11(23-581):81.

Yiqun Chen, Qi Liu, Yi Zhang, Weiwei Sun, Xinyu Ma, Wei Yang, Daiting Shi, Jiaxin Mao, and Dawei Yin. 2025. Tourrank: Utilizing large language models for documents ranking with a tournament-inspired strategy. In *Proceedings of the ACM on Web Conference 2025*, pages 1638–1652.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2021. Overview of the trec 2020 deep learning track. *arXiv e-prints*, pages arXiv–2102.

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*.

Sunhao Dai, Chen Xu, Shicheng Xu, Liang Pang, Zhenhua Dong, and Jun Xu. 2024. Bias and unfairness in information retrieval systems: New challenges in the llm era. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, page 6437–6447, New York, NY, USA. Association for Computing Machinery.

ARPAD E. ELO. 1978. *The rating of chessplayers, past and present*. Published under grant from the American Chess Foundation, New York.

Satoshi Fukuda and Yoichi Tomiura. 2025. Ranking method for an academic paper search with an emphasis on comprehensiveness. *Discover Computing*, 28(1):40.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Ralf Herbrich, Tom Minka, and Thore Graepel. 2006. Trueskill™: a bayesian skill rating system. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'06, page 569–576, Cambridge, MA, USA. MIT Press.

Charles AR Hoare. 1961. Algorithm 65: find. *Communications of the ACM*, 4(7):321–322.

Can Jin, Hongwu Peng, Anxiang Zhang, Nuo Chen, Jiahui Zhao, Xi Xie, Kuangzheng Li, Shuya Feng, Kai Zhong, Caiwen Ding, and Dimitris N. Metaxas. 2025. Rankflow: A multi-role collaborative reranking workflow utilizing large language models. *Preprint*, arXiv:2502.00709.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, and 17 others. 2022. Language models (mostly) know what they know. *Preprint*, arXiv:2207.05221.

Alex Kendall and Yarin Gal. 2017. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Patrick 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. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

Haowei Liu, Xuyang Wu, Guohao Sun, Zhiqiang Tao, and Yi Fang. 2024a. Chainrank-dpo: Chain rank direct preference optimization for llm rankers. *Preprint*, arXiv:2412.14405.

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

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, and Adam Roberts. 2023. The flan collection: Designing data and methods for effective instruction tuning. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 22631–22648. PMLR.

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.

Xueguang Ma, Xinyu Zhang, Ronak Pradeep, and Jimmy Lin. 2023. Zero-shot listwise document reranking with a large language model. *Preprint*, arXiv:2305.02156.

Thomas Minka. 2004. Power ep. Technical report, Technical report, Microsoft Research, Cambridge.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human-generated machine reading comprehension dataset.

Rodrigo Nogueira and Kyunghyun Cho. 2020. Passage re-ranking with bert. *Preprint*, arXiv:1901.04085.

Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document ranking with a pre-trained sequence-to-sequence model. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 708–718, Online. Association for Computational Linguistics.

Jakub Podolak, Leon Peric, Mina Janicijevic, and Roxana Petcu. 2025. Beyond reproducibility: Advancing zero-shot llm reranking efficiency with setwise insertion. *Preprint*, arXiv:2504.10509.

Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023. Rankvicuna: Zero-shot listwise document reranking with open-source large language models. *Preprint*, arXiv:2309.15088.

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.

Muhammad Rashid, Jannat Meem, Yue Dong, and Vagelis Hristidis. 2024. EcoRank: Budget-constrained text re-ranking using large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 13049–13063, Bangkok, Thailand. Association for Computational Linguistics.

Stephen Robertson, Hugo Zaragoza, and 1 others. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Chenglei Si, Dan Friedman, Nitish Joshi, Shi Feng, Danqi Chen, and He He. 2023. Measuring inductive biases of in-context learning with underspecified demonstrations. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11289–11310, Toronto, Canada. Association for Computational Linguistics.

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.

Raphael Tang, Crystina Zhang, Xueguang Ma, Jimmy Lin, and Ferhan Ture. 2024. Found in the middle: Permutation self-consistency improves listwise ranking in large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2327–2340, Mexico City, Mexico. Association for Computational Linguistics.

Yi Tay, Mostafa Dehghani, Vinh Q. Tran, Xavier Garcia, Jason Wei, Xuezhi Wang, Hyung Won Chung, Siamak Shakeri, Dara Bahri, Tal Schuster, Huaixiu Steven Zheng, Denny Zhou, Neil Houlsby, and Donald Metzler. 2023. Ul2: Unifying language learning paradigms. *Preprint*, arXiv:2205.05131.

Zhiqiu Xia, Jinxuan Xu, Yuqian Zhang, and Hang Liu. 2025. A survey of uncertainty estimation methods on large language models. *Preprint*, arXiv:2503.00172.

Soyoung Yoon, Eunbi Choi, Jiyeon Kim, Hyeongu Yun, Yireun Kim, and Seung-won Hwang. 2024. ListT5: Listwise reranking with fusion-in-decoder improves zero-shot retrieval. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2287–2308, Bangkok, Thailand. Association for Computational Linguistics.

Yifan Zeng, Ojas Tendolkar, Raymond Baartmans, Qingyun Wu, Lizhong Chen, and Huazheng Wang. 2024. Llm-rankfusion: Mitigating intrinsic inconsistency in llm-based ranking. *Preprint*, arXiv:2406.00231.

Longhui Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, Meishan Zhang, and Min Zhang. 2024. A two-stage adaptation of large language models for text ranking. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 11880–11891, Bangkok, Thailand. Association for Computational Linguistics.

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.

Honglei Zhuang, Zhen Qin, Rolf Jagerman, Kai Hui, Ji Ma, Jing Lu, Jianmo Ni, Xuanhui Wang, and Michael Bendersky. 2023a. 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*, SIGIR '23, page 2308–2313, New York, NY, USA. Association for Computing Machinery.

Shengyao Zhuang, Bing Liu, Bevan Koopman, and Guido Zuccon. 2023b. Open-source large language models are strong zero-shot query likelihood models for document ranking. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 8807–8817, Singapore. Association for Computational Linguistics.

Shengyao Zhuang, Xueguang Ma, Bevan Koopman, Jimmy Lin, and Guido Zuccon. 2025. Rank-r1: Enhancing reasoning in llm-based document rerankers via reinforcement learning. *Preprint*, arXiv:2503.06034.

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, page 38–47, New York, NY, USA. Association for Computing Machinery.

A Bayesian Rating Systems

Bayesian rating systems provide probabilistic frameworks to estimate latent skill levels or quality scores of entities based on observed outcomes of comparisons or matches. Such systems leverage Bayesian inference principles, combining prior knowledge with observed data to update skill estimations dynamically. The general characteristics include modeling uncertainty explicitly, supporting incremental updating, and providing robustness to noise and incomplete data. Two widely-adopted Bayesian rating systems are Elo (ELO, 1978) and its more advanced successor, TrueSkill (Herbrich et al., 2006), which progressively extend rating complexity and flexibility.

A.1 Elo Rating System

The Elo rating system is a foundational Bayesian rating method originally designed to quantify the relative skill levels of chess players. In this system, each player's ability is represented by a single numerical rating. When two players compete, the ratings are updated based on the observed outcome compared to the expected outcome calculated from current ratings. A player's rating increases after wins against higher-rated opponents and decreases upon losses or unexpected outcomes. The Elo system's simplicity and adaptability make it particularly effective in scenarios involving sequential pairwise competitions.

A.2 TrueSkill Rating System

TrueSkill, introduced by Microsoft, generalizes the Elo rating system by explicitly modeling player skills using probability distributions rather than single scalar values. Specifically, TrueSkill represents each player's skill as a Gaussian distribution characterized by two parameters: a mean μ reflecting the estimated skill level, and a standard deviation σ capturing the uncertainty of this estimate. Following each match, TrueSkill applies approximate Bayesian inference to update these parameters according to the observed results, factoring in the certainty of each player's current rating. This mechanism enables TrueSkill to naturally handle multiplayer and team-based matches, uncertain outcomes, and noisy comparisons.

A.3 Bayesian Update Details

We adopt a Gaussian-based update rule derived from the 1v1 setting in TrueSkill (Herbrich et al., 2006), adapted for document ranking.

Given two documents D_i and D_j with current relevance estimates $\mu_{i,0}, \sigma_{i,0}^2$ and $\mu_{j,0}, \sigma_{j,0}^2$, we define the following intermediate quantities:

$$\delta = \mu_{i,0} - \mu_{j,0}, \quad c^2 = \sigma_{i,0}^2 + \sigma_{j,0}^2 + 2\beta^2,$$

$$t = \frac{\delta}{\sqrt{c^2}}, \quad v(t) = \frac{\phi(t)}{\Phi(t)}, \quad w(t) = v(t)(v(t)+t),$$

where $\phi(t)$ and $\Phi(t)$ denote the probability density function and cumulative distribution function of the standard normal distribution, respectively. The parameter β is a fixed constant that controls comparison noise; we follow the TrueSkill default and set $\beta = \mu_0/3$.

We then define the Bayesian updates to the relevance distribution of D_i, under two possible outcomes:

If D_i wins over D_j:

$$\mathcal{N}(\mu_{\text{win}}, \sigma_{\text{win}}^2)$$

$$\Delta\lambda^+ = \frac{\sigma_{i,0}^4}{c^2} \cdot w(t),$$
$$\Delta\tau^+ = \frac{\sigma_{i,0}^2}{\sqrt{c^2}} \cdot v(t) + \mu_{i,0} \cdot \Delta\lambda^+,$$
$$\lambda_{\text{win}} = \lambda_{i,0} + \Delta\lambda^+,$$
$$\tau_{\text{win}} = \tau_{i,0} + \Delta\tau^+,$$
$$\mu_{\text{win}} = \frac{\tau_{\text{win}}}{\lambda_{\text{win}}}, \quad \sigma_{\text{win}}^2 = \frac{1}{\lambda_{\text{win}}}.$$

If D_i loses to D_j:

$$\mathcal{N}(\mu_{\text{loss}}, \sigma_{\text{loss}}^2)$$

$$\Delta\lambda^- = \frac{\sigma_{i,0}^4}{c^2} \cdot w(-t),$$
$$\Delta\tau^- = -\frac{\sigma_{i,0}^2}{\sqrt{c^2}} \cdot v(-t) + \mu_{i,0} \cdot \Delta\lambda^-,$$
$$\lambda_{\text{loss}} = \lambda_{i,0} + \Delta\lambda^-,$$
$$\tau_{\text{loss}} = \tau_{i,0} + \Delta\tau^-,$$
$$\mu_{\text{loss}} = \frac{\tau_{\text{loss}}}{\lambda_{\text{loss}}}, \quad \sigma_{\text{loss}}^2 = \frac{1}{\lambda_{\text{loss}}}.$$

B Implementation Details

B.1 Detailed Explanation of Datasets

The TREC Deep Learning (DL) 2019 (Craswell et al., 2020) and 2020 (Craswell et al., 2021) datasets are benchmark collections designed to evaluate document ranking systems in complex information retrieval tasks. Both datasets are based on queries derived from real-world search logs and are built on top of the MS MARCO (Nguyen et al., 2016) passage and document corpora. The TREC DL 2019 dataset includes 43 queries with graded relevance judgments, while the 2020 version expands the test set to 54 queries. Documents are written in English and come from a large web-scale corpus, with each query typically associated with hundreds to thousands of candidate documents. These datasets emphasize fine-grained relevance estimation and are widely used for evaluating neural re-ranking models and LLM-based retrieval methods. The reported NDCG@10 scores are averaged over the entire dataset with a single run, ensuring stable and reliable evaluation results.

B.2 Parameter Settings

For a fair comparison, we set the number of comparison rounds for TourRank and PRP-Graph to 10. Since the original implementation of TourRank only supports the OpenAI API, we re-implemented it with a T5-based interface. For Setwise-Insertion, we adopt the best-performing variant, *Setwise Insertion Sort Compare Prior*, as reported in their paper (Podolak et al., 2025). All other baselines are used with their default hyperparameters. In our method, we set $\lambda = 2/3$ (see Section 3.3) to balance effectiveness and efficiency.

B.3 Prompts

For TourRank, we adopt their default listwise prompt:

> **System Prompt**
> You are an intelligent assistant that can compare multiple documents based on their relevancy to the given query.

> **User Prompt**
> 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}`.
> 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.
> Do not provide any explanation or commentary. Output format:
> `Document 3, ..., Document 1`

For PRP-Graph, we also adopt their default pairwise prompt:

> Given a query `{query}`, which of the following two passages is more relevant to the query?
>
> Passage A: `{document_1}`
>
> Passage B: `{document_2}`
>
> Output Passage A or Passage B:

For the remaining methods—REALM, Setwise-Heapsort, and Setwise-Insertion—we use a consistent setwise prompt of the following form:

> Given a query {query}, which of the following passages is the most relevant to the query?
> {passages}
> Output only the passage label of the most relevant passage:

For Setwise-Insertion, we additionally append the following sentence to the prompt:

> If their relevance is similar, or none of them is relevant, output A.

This modification follows their original paper, which claims this change as a key contribution.

C Supplementary Evaluations

C.1 Model Capability Analysis

We evaluate the pairwise comparison capability of different LLMs by randomly sampling 500 document pairs per query from the TREC DL 2019 dataset. Each pair is presented to the model for binary relevance judgment. As shown in Table 4, we observe that LLaMA 3 8B exhibits a strong position bias, often favoring the document appearing in a particular position regardless of content. In contrast, Flan-T5 models demonstrate more reliable behavior and stronger alignment with ground-truth preferences in pairwise comparisons.

Choice	Correct	Wrong	Accuracy
LLaMA3 8B			
A (85.6%)	10258	8145	55.7%
B (14.4%)	2669	428	86.2%
Total	12927	8573	60.1%
LLaMA3 70B			
A (54.9%)	9979	1815	84.6%
B (45.1%)	8794	912	90.6%
Total	18773	2727	87.3%
FLAN-T5-XXL 11B			
A (43.9%)	8709	722	92.3%
B (56.1%)	10047	2022	83.2%
Total	18756	2744	87.2%
FLAN-UL2 20B			
A (50.3%)	9684	1137	89.5%
B (49.7%)	9542	1137	89.4%
Total	19226	2274	89.4%

Table 4: Statistical summary of different models' choices in pair-wise comparison on TREC DL 19.