

Contextual Relevance and Adaptive Sampling for LLM-Based Document Reranking

Jerry Huang^{1*}, Siddarth Madala¹, Cheng Niu²,
Julia Hockenmaier¹, Tong Zhang¹

¹University of Illinois at Urbana-Champaign, ²NewsBreak
{jerry8, smadala2, juliahmr, tozhang}@illinois.edu
cheng.niu@newsbreak.com

Abstract

Reranking algorithms have made progress in improving document retrieval quality by efficiently aggregating relevance judgments generated by large language models (LLMs). However, identifying relevant documents for queries that require in-depth reasoning remains a major challenge. Reasoning-intensive queries often exhibit multifaceted information needs and nuanced interpretations, rendering document relevance inherently context dependent and often noisy. To address this, we propose *contextual relevance*, which we define as the probability that a document is relevant to a given query, marginalized over the distribution of different reranking contexts it may appear in (i.e., the set of candidate documents it is ranked alongside and the order in which the documents are presented to a reranking model). While prior works have studied methods to mitigate the positional bias LLMs exhibit by accounting for the ordering of documents, we empirically show that batch composition also materially affects relevance judgments. To efficiently estimate contextual relevance, we propose **TS-SetRank**, a sampling-based, uncertainty-aware reranking algorithm. Empirically, TS-SetRank improves nDCG@10 over retrieval and reranking baselines by 15–25% on BRIGHT and 6–21% on BEIR, highlighting the importance of modeling relevance as context-dependent.

1 Introduction

Large language models (LLMs) have shown strong performance in zero-shot document reranking (Chen et al., 2025; Ma et al., 2024), with the Setwise prompting approach (Zhuang et al., 2024) offering a favorable trade-off between the number of LLM inference calls required and final reranking quality. As illustrated in Figure 1, in this approach, smaller subsets or batches of documents are drawn

from the initial set of retrieved documents and presented to an LLM trained for reranking, which in turn generates per-document binary judgments of relevance to a given query. Final rankings are then constructed by aggregating the model’s relevance judgments across batches using aggregation or well-known sorting algorithms such as bubble sort or heap sort.

Final reranking quality is thus determined by the ability of a reranking model to identify all relevant documents in each batch, while constrained by a fixed inference budget. While past work has investigated the positional bias LLMs exhibit when making relevance judgments (Tang et al., 2024; Vardasbi et al., 2025; Li et al., 2024), we find that the context in which documents are presented to LLMs also impacts their ability to identify all relevant documents. This phenomenon is particularly evident in queries that require deeper levels of reasoning, have multifaceted information needs, and/or contain nuanced interpretations (Zeng et al., 2024; Gienapp et al., 2022). As highlighted by BRIGHT (Su et al., 2025), a benchmark for reasoning-intensive retrieval, both first-stage retrievers and reranking models struggle to retrieve the set of all relevant documents in such cases. In such settings, treating relevance as a stable, deterministic property can be brittle, especially when downstream ranking decisions are based on a small number of noisy Setwise judgments.

As a descriptive example, consider the following prompt:

Judge the relevance of the following passages to the query, “What is the primary ecological role of sea otters in kelp forest ecosystems?”

- (1) *Sea otters are considered a keystone species because they control sea urchin populations, which in turn helps maintain healthy kelp forests.*

*This work was done while Jerry was a Research Intern at NewsBreak.

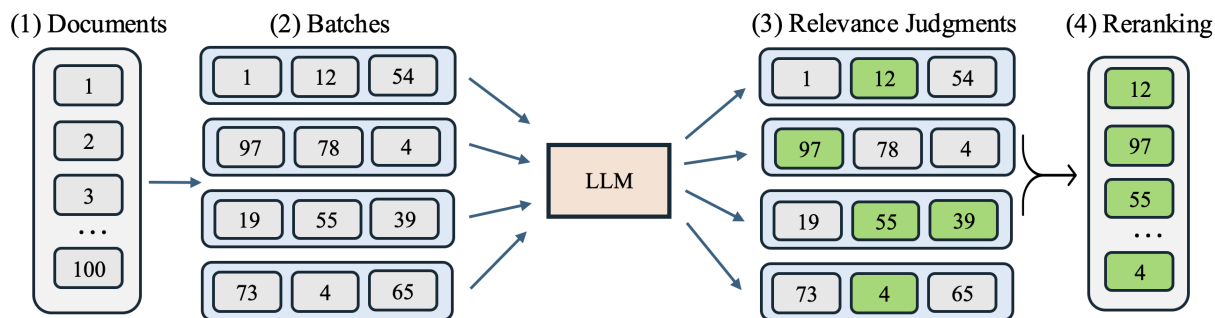


Figure 1: An overview of the Setwise prompting approach for document reranking. Here, we draw batches of size three from the initial list of retrieved documents and prompt an LLM to identify all relevant documents in each batch (highlighted in green). The final reranking list is formed by aggregating all relevance judgments across batches.

- (2) *Unchecked sea urchin populations can devastate kelp forests, making predator control essential for ecosystem stability.*
- (3) *Marine mammals such as seals, whales, and dolphins play critical roles in ocean nutrient cycling.*

Passage (1) is consistently judged relevant, as it directly connects sea otters to the query. Passage (2) is also relevant but its relevance is dependent on the context of (1). Passage (3) is irrelevant. Thus, a reranking model might incorrectly judge all the documents in a batch formed by documents (2) and (3) as irrelevant. One interpretation of this effect is that surrounding documents provide a form of pseudo-relevance feedback, clarifying the aspect of the query that makes another document relevant (Wang et al., 2021).

To capture these effects, we define *contextual relevance* as the probability that a document is judged relevant given a query marginalized over all batches in which it may appear, where each batch is defined by both its contents and the ordering of its documents. Although this formulation assumes independence across documents, for tractability, marginalizing over all possible batches allows us to recover each document’s expected relevance across diverse contexts. This effectively captures contextual dependencies in expectation even when individual batch judgments are interdependent.

A natural approach to rerank a list of documents is to enumerate all possible batches in which each document may appear. Instead, to more efficiently estimate contextual relevance, we propose Thompson Sampling for Setwise Reranking (TS-SetRank), a two-phase Bayesian reranking algorithm that first samples document batches uni-

formly to collect unbiased relevance feedback, and then adaptively constructs batches using Thompson sampling (Agrawal and Goyal, 2012). We evaluate TS-SetRank on BRIGHT (Su et al., 2025) and BEIR (Thakur et al., 2021), two benchmarks for evaluating retrieval capabilities, and compare its Normalized Discounted Cumulative Gain (nDCG@k; Järvelin and Kekäläinen, 2002) performance against deterministic reranking and statistical retrieval baselines.

Our contributions are as follows:

- We propose the notion of *contextual relevance* as a target of estimation, which takes into account both the contents and ordering of retrieved documents within each batch for Setwise reranking.
- We show that variance in LLM-based Setwise relevance judgments arises from LLM sampling variability and contextual dependencies, and empirically quantify the share of each.
- We propose TS-SetRank and show that it outperforms deterministic reranking algorithms in nDCG@10 under comparable inference budgets on BRIGHT and BEIR.

2 Estimating Contextual Relevance

2.1 Problem Formalization

Objective. Let \mathcal{Q} denote the space of user queries and \mathcal{D} the document corpus. For each query $q \in \mathcal{Q}$, a first-stage retriever returns an ordered list of candidate documents:

$$D = (d_{(1)}, d_{(2)}, \dots, d_{(N)}) \subseteq \mathcal{D},$$

where $d_{(1)}$ is the highest-scoring document and $N \ll |\mathcal{D}|$. The reranking objective is to select

an ordered subset of D that maximizes the final reranking quality as measured by a specified retrieval metric such as nDCG@k.

Reranking Model. A Setwise reranking model is an LLM that takes a query q and a small batch of documents $S = (d_1, \dots, d_b)$ where $b \ll N$, and makes a binary relevance judgment for each document.

Contextual Relevance. We define the *contextual relevance* of a document d_i with respect to a query q as its probability of being judged relevant, marginalized over all batches in which it may appear:

$$\theta_{i,q} = \mathbb{E}_{S \sim D_b(d_i)} \left[\Pr(d_i \text{ is judged relevant} \mid q, S) \right]$$

where $D_b(d_i)$ denotes the distribution over size- b batches that include d_i , and $D_b(D)$ denotes the distribution over all size- b batches drawn from D . We use θ_i interchangeably with $\theta_{i,q}$, as the query q is fixed within each reranking task.

Modeling Assumptions. This formulation assumes conditional independence across documents: each document’s judgments are modeled as independent draws following a Bernoulli distribution with mean $\theta_{i,q}$. While this simplification is reasonable when documents do not exhibit strong dependencies, more complex models (e.g., structured bandits or Plackett–Luce) would be required in domains such as multi-hop question answering (Mavi et al., 2024) or citation retrieval (Qian et al., 2024), where relevance depends on prerequisite evidence. Benchmarks like BRIGHT and BEIR do not exhibit such dependencies, making this assumption appropriate in our setting.

2.2 Bayesian Reranking

We model inference-time reranking as a combinatorial semi-bandit problem with a fixed budget of T rounds. At each round $t = 1, \dots, T$, the model selects a batch $S_t \sim D_b(D)$ and queries the reranking model for binary relevance feedback. We maintain independent Beta-Bernoulli posteriors for each document $d_i \in D$, initialized with $\alpha_i = \beta_i = 1$, so that $\theta_i \sim \text{Beta}(\alpha_i, \beta_i)$ represents our uncertainty over d_i ’s contextual relevance.

At each round, we perform:

1. **Batch Selection.** We sample a batch $S_t \sim D_b(D)$ according to our policy (e.g. uniform exploration or Thompson sampling, defined in Section 2.3).

2. **Setwise Feedback.** For each batch S_t , the reranker makes a binary relevance decision for each document. $R(q, S_t)$ denotes the subset of documents judged relevant, and $r_t(d_i) = 1$ if $d_i \in R(q, S_t)$ and 0 otherwise.

3. **Posterior Update.** We update the Beta posterior for each $d_i \in S_t$ following:

$$\alpha_i \leftarrow \alpha_i + r_t(d_i), \quad \beta_i \leftarrow \beta_i + (1 - r_t(d_i)),$$

where the posterior mean

$$\hat{\theta}_i = \frac{\alpha_i}{\alpha_i + \beta_i} \quad \forall d_i \in D$$

serves as our current estimate of d_i ’s contextual relevance.

The final ranking is computed by sorting the candidate set D in descending order by the posterior means $\hat{\theta}_i$.

2.3 TS-SetRank

We now introduce Thompson Sampling for Setwise Reranking (TS-SetRank) as outlined in Algorithm 1, a two-phase algorithm that combines uniform sampling with Thompson sampling to guide batch selection. We maintain independent Beta-Bernoulli posteriors for each document d_i ’s relevance and adaptively allocate queries using posterior sampling, aiming to improve ranking quality under a fixed inference budget T .

TS-SetRank proceeds in two phases: T_f rounds of uniform sampling followed by $T - T_f$ rounds of adaptive sampling.

- **Phase I: Uniform Sampling.** For rounds $t = 1, \dots, T_f$, we sample batches S_t uniformly at random from $D_b(D)$. This provides sufficient initial observations for posterior inference. While one could initialize from a first-stage retriever ranking, we opt for uniform sampling given the relatively weak performance of first-stage retrievers on our chosen benchmarks.
- **Phase II: Adaptive Sampling.** For $t = T_f + 1, \dots, T$, we draw samples $\tilde{\theta}_i \sim \text{Beta}(\alpha_i, \beta_i)$ from each document’s posterior and form the batch S_t by selecting the b documents with the highest sampled values. Thompson sampling balances exploration and exploitation by sampling from the posterior: documents with higher estimated relevance are

Algorithm 1: TS-SetRank

Input: First-stage retrieval results
 $D = \{d_1, \dots, d_N\}$, query q , budget T , batch size b , exploration rounds T_f , reranking model M

Output: Final reranked list of documents

foreach $d_i \in D$ **do**
 \perp Initialize posterior: $\alpha_i \leftarrow 1, \beta_i \leftarrow 1$

for $t = 1$ to T **do**
 if $t \leq T_f$ **then**
 Sample batch
 $S_t \sim \text{Uniform}(D_b(D))$
 else
 foreach $d_i \in D$ **do**
 \perp Draw $\tilde{\theta}_i \sim \text{Beta}(\alpha_i, \beta_i)$
 Let $S_t \leftarrow \{d_i \in D : \tilde{\theta}_i \text{ is among the top-}b\}$
 Obtain feedback: $R_t := R(q, S_t)$ from reranking model M
 foreach $d_i \in S_t$ **do**
 Observe $r_t(d_i) \in \{0, 1\}$
 Update posterior: $\alpha_i \leftarrow \alpha_i + r_t(d_i)$,
 $\beta_i \leftarrow \beta_i + (1 - r_t(d_i))$
 return documents sorted by posterior means $\hat{\theta}_i = \alpha_i / (\alpha_i + \beta_i)$ in descending order.

more likely to be chosen, while uncertain documents still have a chance of being selected to discover overlooked relevant items (Agrawal and Goyal, 2012).

2.4 Theoretical Guarantees

We summarize the theoretical properties of TS-SetRank based on established analyses of Thompson sampling in stochastic semi-bandit settings.

Sublinear Regret. Reranking quality in our experiments is measured by nDCG@10. For analytical tractability, we instead consider a linear *surrogate objective* in which each document d_i is treated as an independent Bernoulli arm with mean contextual relevance θ_i . At each round, the model selects a batch of b documents from the pool of N retrieved documents and observes binary relevance feedback. The surrogate reward is defined as the sum of these feedback signals, representing the number of relevant documents retrieved in the batch.

We note, however, that maximizing this surrogate encourages selection of documents with high expected relevance but does not directly optimize

the position-weighted nDCG@10 objective. Yet, in practice, improving expected relevance tends to increase nDCG@10 empirically.

Under standard assumptions of independent arms, bounded rewards, and sufficient exploration, Thompson sampling in stochastic semi-bandit settings achieves sublinear cumulative regret (Agrawal and Goyal, 2012; Chen et al., 2013; Wang and Chen, 2018):

$$\mathbb{E}[\mathcal{R}(T)] = \tilde{O}(\sqrt{bNT}),$$

where b is the batch size, N the number of retrieved documents, T the number of rounds, and $\mathcal{R}(T)$ denotes the cumulative surrogate regret. Since TS-SetRank follows the same structure, it inherits this sublinear regret bound for the surrogate estimation task.

Posterior Consistency. For each document d_i , feedback observations are Bernoulli-distributed with mean θ_i . Under the standard Beta–Bernoulli update, the posterior mean $\hat{\theta}_i = \alpha_i / (\alpha_i + \beta_i)$ converges almost surely to the true mean θ_i whenever the document is sampled infinitely often, by the strong law of large numbers and Beta–Bernoulli conjugacy.

Uniform Exploration. A non-adaptive policy such as uniform sampling does not exploit accumulated feedback and therefore incurs linear regret, $\mathcal{R}(T) = \Theta(T)$ (Auer et al., 2002). This contrast highlights the benefit of adaptive exploration in reducing long-run cumulative error.

3 Experiments

To evaluate the effectiveness of our framework for modeling contextual relevance and the performance of TS-SetRank, we organize our experiments around the following research questions:

- RQ1.** To what extent is variability in LLM-based relevance judgments attributable to LLM sampling variability versus changes in the surrounding document context?
- RQ2.** How does TS-SetRank compare with uniform sampling, deterministic reranking methods, and standard retrieval baselines (e.g., BM25) in terms of nDCG@10 under a fixed inference budget?
- RQ3.** How does uniform sampling improve reranking performance as the inference budget grows, and when does it reach convergence?

3.1 Experimental Setup

Reranking Model. We train the Setwise reranking model used for all our experiments by fine-tuning the Qwen2.5-7B-Instruct base model (Qwen et al., 2025) on 25,000 training samples from the MS MARCO v2.1 dataset (Nguyen et al., 2016). Post-training is performed using reinforcement learning with verifiable rewards (RLVR) via Group Relative Policy Optimization (GRPO; Shao et al., 2024). Following prior work on reasoning-enhanced rerankers (Zhuang et al., 2025; Weller et al., 2025), our model is trained to leverage test-time compute by first producing an explicit reasoning trace enclosed in predefined reasoning tags before emitting the set of passages it deems relevant.

Reward Function. We define two reward components: a formatting reward and a correctness-based reward. The formatting reward ensures that the model produces outputs enclosed within valid tags (e.g. <answer>...</answer>), which are required for evaluation. Specifically, the model receives a reward of 0.5 if both the opening and closing tags are present and correctly formatted, and 0 otherwise.

The correctness reward measures agreement between the predicted and gold document sets using the F_β metric, computed only when valid <answer> tags are detected; otherwise, it is assigned 0. We use $\beta = 2$ to emphasize recall:

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}}.$$

The total reward is the sum of the formatting and correctness components.

Training Details. All training samples contain a query and batches of size 10, each containing between 1 and 7 relevant documents with the positions of the relevant documents randomly determined for each training sample. All fine-tuning is conducted on H100 GPUs. Additional implementation details, including the specific prompt templates and hyperparameters used are provided in Appendix A.

Datasets & Retrieval. We benchmark our model and reranking algorithms on BRIGHT (Su et al., 2025) and BEIR (Thakur et al., 2021), two widely used benchmarks for evaluating retrieval and reranking quality. Following prior work, we use BM25 (Robertson et al., 1994; Lù, 2024) as the first-stage retriever and use the top-100 retrieved

documents for reranking. We set the batch size to 10 for sampling-based reranking algorithms and set the total inference budget to $T = 100$ as is common in prior work. For BRIGHT, we use the benchmark provided GPT-4o rewritten queries to improve the quality of our first-stage retrieval results. For BEIR, we benchmark on the datasets containing fewer than 2,000 queries and sample 100 queries per dataset for evaluation.

Baselines & Variants

- **BM25:** A statistical retrieval method based on term frequency and inverse document frequency, widely used as a first-stage retriever. BM25 serves as a strong non-neural baseline.
- **Heapify:** A deterministic Setwise reranking algorithm that performs a sequence of binary heap comparisons over document triplets, requiring a total of $O(N \log N)$ LLM inference calls. Zhuang et al. (2024) show that Heapify achieves strong performance under a fixed budget.
- **TS-SetRank (Uniform):** A variant of TS-SetRank that performs 100 steps of uniform sampling with no adaptive Thompson sampling phase. This corresponds to the case TS(100/0), and serves as a non-adaptive baseline for comparison.
- **TS-SetRank ($X/100-X$):** Our full TS-SetRank algorithm, parameterized by X steps of uniform sampling followed by $100-X$ steps of Thompson sampling. Varying X allows us to study the exploration-exploitation trade-off and assess how much adaptive sampling improves performance.

Metrics We report nDCG@10 after $T = 100$ reranking steps, along with intermediate performance snapshots to visualize the progression of reranking quality. All results are reported as mean \pm one standard deviation across three random seeds.

3.2 RQ1: Reranking Model Variability

We first empirically quantify how much of the variability in LLM-based relevance judgments can be attributed to intrinsic model stochasticity (token sampling) versus contextual factors such as batch composition and document order.

Regime	Batch size $b = 2$			Batch size $b = 10$		
	Intrinsic	Positional	Total	Intrinsic	Positional	Total
Accuracy	0.27	0.26	0.26	0.28	0.28	0.27
Variance	0.063	0.076	0.083	0.062	0.103	0.113

Table 1: Mean accuracy and per-query variance of LLM judgments on BRIGHT for batch sizes 2 and 10 under three regimes: *Intrinsic* (fixed batch documents, fixed order), *Positional* (fixed batch documents, shuffled order), and *Total* (resampled documents, shuffled order). Variance is computed per query across repeated trials and then averaged.

Experimental Setup. For each query in BRIGHT, we randomly select one relevant document d^+ from the ground-truth set and construct 30 batches of size b . Each batch contains d^+ alongside $b-1$ other documents sampled from the top-100 BM25 candidates. For each batch, accuracy is recorded as 1 if d^+ is judged relevant and 0 otherwise. This setup abstracts away from ranking metrics such as nDCG and focuses exclusively on per-document judgment variability at the batch level.

We evaluate three regimes:

1. **Intrinsic.** All 30 batches we construct are identical in both composition and order, so variability reflects only randomness introduced by token sampling.
2. **Positional.** Batch composition is fixed, but document order is shuffled. Variability here includes both intrinsic randomness and any additional sensitivity to order.
3. **Total.** Distractors are resampled each time and order randomized, so variability includes intrinsic, positional, and compositional effects.

Batch Size Effects. Table 1 reports mean accuracy and per-query variance under each regime for batch sizes 2 and 10. We observe that accuracy improves modestly with larger batch sizes, but overall variability also increases. The intrinsic baseline remains stable across batch sizes, suggesting that the additional variability arises from contextual factors, particularly to document order when b is larger.

Because the regimes build on one another, we can quantify the contributions of each by computing their differences. This shows how much additional variability is attributable to ordering effects beyond intrinsic variability, and how much more variability is added when we also consider compositional effects. From Table 1, we find that contextual factors explain about 25% of the total variability

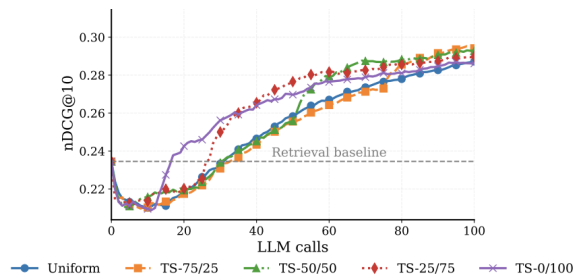


Figure 2: TS-SetRank variants achieve faster gains under smaller inference budgets, particularly when using fewer exploration rounds. The initial dip in reranking quality reflects noisy posterior estimates early in sampling, but performance quickly surpasses the retrieval baseline as more rounds are completed.

for size-2 batches and 45% for size-10 batches. Most of the increase comes from sensitivity to order (16% and 36%, respectively), while compositional effects remain fairly steady at around 9%. These results serve only as an *empirical heuristic* as the sources of variance are not completely independent.

We further study the effects of batch size under the uniform sampling setting by varying the size of each batch from 2 to 10 while using a fixed inference budget. As shown in Table 2, larger batch sizes correlate with better performance as measured by nDCG@10, suggesting that larger batches improve the efficiency of contextual averaging under a fixed number of LLM calls. Additionally, as the number of sampling steps increases, similar to other experiments, reranking quality typically drops at the beginning, reflecting noisy relevance estimates in the early stages of sampling, and then gradually recovers as additional judgments are accumulated.

3.3 RQ2: Comparison of Reranking Algorithms

We evaluate TS-SetRank against deterministic and non-adaptive reranking baselines under a fixed inference budget of $T=100$. Table 3 reports

Method	Step 0	Step 10	Step 20	Step 30	Step 40	Step 50	Step 60	Step 70	Step 80	Step 90	Step 100
uniform-batch-2	0.235	0.220	0.216	0.211	0.205	0.202	0.203	0.202	0.202	0.195	0.193
uniform-batch-4	0.235	0.220	0.216	0.214	0.211	0.207	0.211	0.219	0.222	0.229	0.229
uniform-batch-6	0.235	0.222	0.214	0.213	0.217	0.224	0.228	0.237	0.246	0.252	0.257
uniform-batch-8	0.235	0.209	0.212	0.215	0.228	0.233	0.243	0.249	0.257	0.264	0.266
uniform-batch-10	0.235	0.215	0.209	0.224	0.234	0.249	0.259	0.263	0.271	0.273	0.276

Table 2: Effect of batch size on nDCG@10 for uniform sampling over 100 reranking steps. Larger batch sizes yield better final performance under a fixed inference budget.

Benchmark	Snapshot t	BM25s	Heapify	Uniform	TS-75/25	TS-50/50	TS-25/75	TS-0/100
BRIGHT	100	0.235	0.2560	0.287 ± 0.003	0.294 ± 0.002	0.293 ± 0.003	0.290 ± 0.004	0.286 ± 0.002
	50	—	—	0.258 ± 0.003	0.255 ± 0.004 †	0.255 ± 0.003 †	0.276 ± 0.006	0.270 ± 0.003
BEIR	100	0.357	0.4080	0.421 ± 0.003	0.429 ± 0.002	0.424 ± 0.004	0.431 ± 0.010	0.428 ± 0.005
	50	—	—	0.393 ± 0.000	0.401 ± 0.009 †	0.395 ± 0.002 †	0.417 ± 0.001	0.417 ± 0.005

Table 3: nDCG@10 at snapshots t at 50 and 100 on BRIGHT and BEIR. TS- $\alpha/(100 - \alpha)$ denotes TS-SetRank with α uniform exploration steps and $100 - \alpha$ Thompson sampling steps. Uniform corresponds to TS-SetRank with 100 rounds exploration. † indicates snapshots before Thompson sampling begins. Full topic-level results for both benchmarks are provided in Appendix B.1 and B.2.

nDCG@10 at two checkpoints ($t=50$ and 100) on BRIGHT and BEIR.

At $t=100$, Uniform achieves strong performance, reflecting the benefit of averaging judgments across reranking contexts. TS-SetRank attains comparable or slightly better final performance across different exploration–exploitation splits, with dataset-specific variation: on BRIGHT, TS-75/25 achieves the best score (0.294), while on BEIR, TS-25/75 slightly outperforms other variants. These results suggest that adaptive sampling is at least as effective as uniform sampling in the long run.

The advantage of TS-SetRank is most evident under smaller budgets. At the halfway point ($t=50$), illustrated in Figure 2 and reported in Table 3, TS-SetRank variants with more exploitation (e.g., TS-25/75 and TS-0/100) outperform Uniform. On BRIGHT, TS-25/75 improves upon Uniform by 1.8 points, and on BEIR, TS-25/75 and TS-0/100 reach 0.417 nDCG@10 compared to 0.393 for Uniform. This indicates that TS-SetRank can more effectively allocate its limited inference budget by focusing on promising candidates earlier.

Heapify underperforms across both datasets due to its reliance on potentially noisy pairwise comparisons. In such regimes, relevant documents often fail to consistently accumulate wins to rise to the top of the heap. This limitation aligns with noisy sorting theory, which shows that reliable rankings under stochastic comparisons require repeated re-sampling and aggregation (Braverman and Mossel,

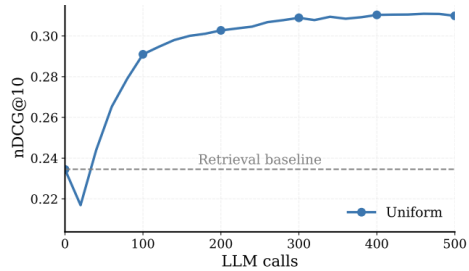


Figure 3: nDCG@10 vs. LLM calls on BRIGHT. Uniform sampling converges after ≈ 300 inference calls, showing diminishing returns.

2008).

3.4 RQ3: Convergence of Uniform Sampling

To investigate how uniform sampling behaves as the inference budget increases, we examine its convergence characteristics under an unconstrained evaluation setting. Unlike the adaptive TS-SetRank variants, uniform sampling performs all reranking steps by drawing document batches uniformly at random, without leveraging feedback from previous rounds. This setup allows us to isolate the effect of repeated contextual averaging on final ranking quality.

As shown in Figure 3, nDCG@10 improves rapidly during the initial 100–200 inference calls, followed by progressively smaller gains beyond 300 calls. Empirically, convergence occurs after approximately 300 inference calls, at which point additional sampling yields negligible improvements in nDCG@10.

3.5 Inference Costs & Parallelism

Both TS-SetRank and Uniform sampling operate under a fixed inference budget of 50 or 100 inference calls per query, each evaluating batches of 10 documents. In contrast, Heapify follows a binary comparison tree: it performs 50 initial comparisons (one per internal node for $n=100$) and up to 7 additional comparisons per extraction during its top-10 selection phase, yielding an average of 110 LLM inference calls per query in our experiments. Fully reranking all 100 documents requires up to 743 comparisons in the worst case. While individual TS-SetRank calls process more tokens due to larger batch sizes, amortizing computation across multiple documents substantially improves throughput. Empirically, our results in Table 3 show that at the snapshot of $t=50$, TS-SetRank achieves stronger performance while remaining more efficient when compared to Heapify.

Uniform sampling trivially parallelizes across queries and document batches, achieving linear scaling with the number of available workers. By contrast, Heapify exposes structured but depth-dependent parallelism: its binary comparison tree allows concurrent evaluation of all comparisons within a level once their descendants have been resolved, yielding $O(n)$ total work and an ideal span of $O(\log n)$. This design enables high concurrency near the leaves but diminishing efficiency toward the root, where the active frontier narrows and synchronization costs dominate.

Extensions to Delayed Feedback. Phase II of TS-SetRank is not parallelizable due to the reliance of feedback from previous rounds. To increase the parallelism of TS-SetRank, we propose a throughput-optimized variant, TS-SetRank-T (see Appendix C), which extends TS-SetRank by deferring parameter updates within each phase while preserving Thompson sampling semantics. Under mild regularity assumptions, such delayed-update schemes maintain sublinear regret in the combinatorial semi-bandit setting with delayed feedback (Joulani et al., 2013; Karbasi et al., 2021), enabling more efficient contextual relevance estimation when more workers are available.

Tradeoffs. More broadly, the computational and quality tradeoffs among these reranking methods suggest that the preferred reranking strategy depends heavily on deployment setting. Uniform sampling is a strong choice when throughput and

end-to-end latency are the primary concerns, particularly when sufficient parallel resources are available. Heapify, by contrast, has a deterministic inference cost for a given number of documents and top- k target, rather than a tunable budget. TS-SetRank is most suitable for maximizing reranking quality under a fixed number of LLM calls with less emphasis on minimizing end-to-end latency. TS-SetRank-T provides a practical middle ground, making it preferable in environments where some loss in quality is acceptable in exchange for improved throughput and lower latency.

4 Related Work

4.1 Reranking Paradigms

The four dominant prompting approaches for using LLMs in zero-shot document reranking include Pointwise (Sachan et al., 2023), Pairwise (Qin et al., 2024), Listwise (Ma et al., 2023; Pradeep et al., 2023a,b), and Setwise (Zhuang et al., 2024) prompting. Respectively, each of these prompting approaches employs an LLM to generate: (1) a binary judgment of relevance for a single document, (2) a preference judgment for a pair of documents, (3) an ordered list of relevance scores given a list of documents, and (4) a subset of relevant documents from a candidate set. In this work, we focus on improving Setwise reranking models, which have been shown to balance efficiency and accuracy by allowing for multiple pairwise comparisons to be batched within a single Setwise prompt. Moreover, because most retrieval datasets are designed for Pointwise or Setwise evaluation, training effective listwise reranking models remains challenging (Zhang et al., 2025). Other works have also studied the performance of training reasoning language models (RLMs) specifically for the task of document reranking and have observed improved performance on reasoning-based document retrieval tasks (Zhuang et al., 2025; Weller et al., 2025).

4.2 Non-Transitivity of LLM Judgments

The transitivity of pairwise judgments has been well-studied in the field of information retrieval, with empirical studies showing that this property holds reliably only when the judgment model is well-trained (Hui and Berberich, 2017; Xu et al., 2025). In LLM-based reranking, studies have found that using smaller models or benchmarking on complex or subjective tasks is correlated with

higher intransitivity (Liu et al., 2025). Reasoning-intensive queries also exhibit intransitive judgments as they often involve nuanced forms of relevance (i.e., multiple valid ways a document can relate to a query). To mitigate intransitivity and improve reranking quality, prior work has proposed methods such as aggregating judgments from multiple reranking models (Zeng et al., 2024), subsampling a larger set of pairwise comparisons (Gienapp et al., 2022), using permutation-based self-consistency to marginalize positional bias (Tang et al., 2024), and employing sorting-inspired methods such as Heapify (Zhuang et al., 2024) and Pairwise Ranking Prompting (Qin et al., 2024).

4.3 Adaptive Ranking Methods

Adaptive inference methods have a long history in information retrieval, often framing document selection as a sequential decision-making problem. Early work applied multi-armed bandit algorithms to optimize ranking under click-through feedback (Li et al., 2010), later extending to combinatorial and semi-bandit settings that model item interactions (Chen et al., 2013). Thompson sampling (Agrawal and Goyal, 2012) offers a Bayesian framework for balancing exploration and exploitation, with strong regret bounds in the i.i.d. setting.

5 Discussion

This work introduces *contextual relevance*, a probabilistic framework for modeling document relevance as a function not only of the query but also of the surrounding batch context. Our formulation challenges standard assumptions in reranking, namely, that relevance judgments are deterministic and context-independent, and instead treats them as stochastic outcomes conditioned on batch composition and ordering. Our experiments and ablations quantify the extent of contextual and intrinsic variance in LLM-based judgments and demonstrate that reranking strategies which explicitly marginalize over this variability can substantially outperform deterministic baselines under fixed inference budgets.

Limitations

A key modeling simplification in our approach is the assumption of conditional independence across documents. While this abstraction enables tractable and effective Bayesian inference, it does not capture settings in which document-level relevance

depends on interdependencies such as those seen in multi-hop question answering or citation retrieval.

Extensions of our work include modeling structured dependencies between documents (e.g., via graph-based reranking or complex latent variable models) and training reranking models that produce continuous rather than binary relevance signals. These enhancements could enable more sample-efficient reranking under tight budgets and may generalize better to complex information-seeking tasks. Additionally, integrating contextual relevance into end-to-end RAG pipelines, where reranking quality directly impacts answer generation, remains a compelling area for future exploration.

References

- Shipra Agrawal and Navin Goyal. 2012. [Analysis of thompson sampling for the multi-armed bandit problem](#). In *Proceedings of the 25th Annual Conference on Learning Theory*, volume 23 of *Proceedings of Machine Learning Research*, pages 39.1–39.26, Edinburgh, Scotland. PMLR.
- Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. 2002. [Finite-time analysis of the multiarmed bandit problem](#). *Machine Learning*, 47(2-3):235–256.
- Mark Braverman and Elchanan Mossel. 2008. Noisy sorting without resampling. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '08, page 268–276, USA. Society for Industrial and Applied Mathematics.
- Shijie Chen, Bernal Jimenez Gutierrez, and Yu Su. 2025. [Attention in large language models yields efficient zero-shot re-rankers](#). In *The Thirteenth International Conference on Learning Representations*.
- Wei Chen, Yajun Wang, and Yang Yuan. 2013. Combinatorial multi-armed bandit: general framework, results and applications. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML '13, page I–151–I–159. JMLR.org.
- Lukas Gienapp, Maik Fröbe, Matthias Hagen, and Martin Potthast. 2022. [Sparse pairwise re-ranking with pre-trained transformers](#). In *Proceedings of the 2022 ACM SIGIR International Conference on Theory of Information Retrieval*, ICTIR '22, page 72–80, New York, NY, USA. Association for Computing Machinery.
- Kai Hui and Klaus Berberich. 2017. [Transitivity, time consumption, and quality of preference judgments in crowdsourcing](#). In *Advances in Information Retrieval - 39th European Conference on IR Research, ECIR 2017, Aberdeen, UK, April 8-13, 2017, Proceedings*, volume 10193 of *Lecture Notes in Computer Science*, pages 239–251.
- Kalervo Järvelin and Jaana Kekäläinen. 2002. [Cumulated gain-based evaluation of ir techniques](#). *ACM Trans. Inf. Syst.*, 20:422–446.
- Pooria Joulani, András György, and Csaba Szepesvári. 2013. [Online learning under delayed feedback](#). In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 1453–1461. JMLR.org.
- Amin Karbasi, Vahab Mirrokni, and Mohammad Shadravan. 2021. Parallelizing thompson sampling. In *Proceedings of the 35th International Conference on Neural Information Processing Systems, NIPS '21*, Red Hook, NY, USA. Curran Associates Inc.
- Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. 2010. [A contextual-bandit approach to personalized news article recommendation](#). In *Proceedings of the 19th international conference on World wide web*, WWW '10, page 661–670. ACM.
- Zongjie Li, Chaozheng Wang, Pingchuan Ma, Daoyuan Wu, Shuai Wang, Cuiyun Gao, and Yang Liu. 2024. [Split and merge: Aligning position biases in LLM-based evaluators](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11084–11108, Miami, Florida, USA. Association for Computational Linguistics.
- Yinhong Liu, Han Zhou, Zhijiang Guo, Ehsan Shareghi, Ivan Vulić, Anna Korhonen, and Nigel Collier. 2025. [Aligning with human judgement: The role of pairwise preference in large language model evaluators](#). *arXiv preprint arXiv:2403.16950*.
- Xing Han Lù. 2024. [Bm25s: Orders of magnitude faster lexical search via eager sparse scoring](#). *arXiv preprint arXiv:2407.03618*.
- Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. 2024. [Fine-tuning llama for multi-stage text retrieval](#). In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, page 2421–2425, New York, NY, USA. Association for Computing Machinery.
- Xueguang Ma, Xinyu Zhang, Ronak Pradeep, and Jimmy Lin. 2023. [Zero-shot listwise document reranking with a large language model](#). *arXiv preprint arXiv:2305.02156*.
- Vaibhav Mavi, Anubhav Jangra, and Adam Jatowt. 2024. [Multi-hop question answering](#). *Found. Trends Inf. Retr.*, 17(5):457–586.
- 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](#). In *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016)*, Barcelona, Spain, December 9, 2016, volume 1773 of *CEUR Workshop Proceedings*. CEUR-WS.org.
- Ronak Pradeep, Sahel Sharifmoghammad, and Jimmy Lin. 2023a. [Rankvicuna: Zero-shot listwise document reranking with open-source large language models](#). *arXiv preprint arXiv:2309.15088*.
- Ronak Pradeep, Sahel Sharifmoghammad, and Jimmy Lin. 2023b. [Rankzephyr: Effective and robust zero-shot listwise reranking is a breeze!](#) *arXiv preprint arXiv:2312.02724*.
- Haosheng Qian, Yixing Fan, Ruqing Zhang, and Jiafeng Guo. 2024. [On the capacity of citation generation by large language models](#). *ArXiv*, abs/2410.11217.

- 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.
- Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, and 1 others. 2025. [Qwen2.5 technical report](#). *Preprint*, arXiv:2412.15115.
- Stephen Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. 1994. Okapi at trec-3. pages 0–.
- Devendra Singh Sachan, Mike Lewis, Mandar Joshi, Armen Aghajanyan, Wen tau Yih, Joelle Pineau, and Luke Zettlemoyer. 2023. Improving passage retrieval with zero-shot question generation. *arXiv preprint arXiv:2204.07496*.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.
- Hongjin Su, Howard Yen, Mengzhou Xia, Weijia Shi, Niklas Muennighoff, Han yu Wang, Liu Haisu, Quan Shi, Zachary S Siegel, Michael Tang, Ruoxi Sun, Jinsung Yoon, Sercan O Arik, Danqi Chen, and Tao Yu. 2025. [BRIGHT: A realistic and challenging benchmark for reasoning-intensive retrieval](#). In *The Thirtieth International Conference on Learning Representations*.
- 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.
- 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 *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.
- Ali Vardasbi, Gustavo Penha, Claudia Hauff, and Hugues Bouchard. 2025. [Adaptive repetition for mitigating position bias in llm-based ranking](#). *Preprint*, arXiv:2507.17788.
- Siwei Wang and Wei Chen. 2018. [Thompson sampling for combinatorial semi-bandits](#). In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5114–5122. PMLR.
- Xiao Wang, Craig Macdonald, Nicola Tonellotto, and Iadh Ounis. 2021. Pseudo-relevance feedback for multiple representation dense retrieval. In *Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval*, pages 297–306.
- Orion Weller, Kathryn Ricci, Eugene Yang, Andrew Yates, Dawn Lawrie, and Benjamin Van Durme. 2025. Rank1: Test-time compute for reranking in information retrieval. *arXiv preprint arXiv:2502.18418*.
- Yi Xu, Laura Ruis, Tim Rocktäschel, and Robert Kirk. 2025. Investigating non-transitivity in llm-as-a-judge. *arXiv preprint arXiv:2502.14074*.
- Yifan Zeng, Ojas Tendolkar, Raymond Baartmans, Qingyun Wu, Lihong Chen, and Huazheng Wang. 2024. Llm-rankfusion: Mitigating intrinsic inconsistency in llm-based ranking. *arXiv preprint arXiv:2406.00231*.
- Crystina Zhang, Sebastian Hofstätter, Patrick Lewis, Raphael Tang, and Jimmy Lin. 2025. Rank-without-gpt: Building gpt-independent listwise rerankers on open-source large language models. In *European Conference on Information Retrieval*, pages 233–247. Springer.
- 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 2024, page 38–47. ACM.

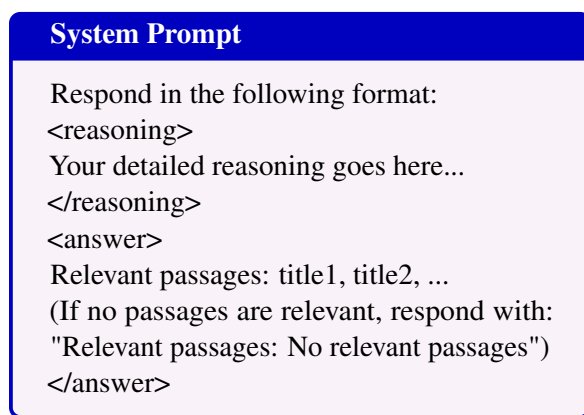
A Experimental Setup

A.1 Post-Training Setup

We fine-tune our reranking model, Qwen2.5-7B-Instruct, over one epoch with Axolotl¹ at a constant learning rate of 1.0e-6, global batch size 294, KL coefficient 0.01, and 7 rollouts across 42 problems per batch. For inference, we set the temperature to 0.6 for all experiments.

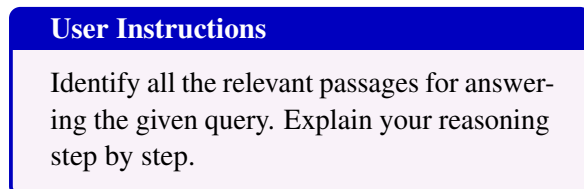
A.2 Prompts

We use the following system prompt and instructions during training and inference:

A blue-bordered box with a dark blue header containing the text: Respond in the following format: <reasoning> Your detailed reasoning goes here... </reasoning> <answer> Relevant passages: title1, title2, ... (If no passages are relevant, respond with: "Relevant passages: No relevant passages") </answer>

```
System Prompt
Respond in the following format:
<reasoning>
Your detailed reasoning goes here...
</reasoning>
<answer>
Relevant passages: title1, title2, ...
(If no passages are relevant, respond with:
"Relevant passages: No relevant passages")
</answer>
```

Figure 4: System prompt used for all experiments.

A blue-bordered box with a dark blue header containing the text: Identify all the relevant passages for answering the given query. Explain your reasoning step by step.

```
User Instructions
Identify all the relevant passages for answering
the given query. Explain your reasoning
step by step.
```

Figure 5: Instructions provided to the LLM during training and inference.

B More Experimental Results

B.1 BRIGHT Results.

Table 4 reports nDCG@10 results on BRIGHT, split by topic with an inference budget of $T = 100$.

B.2 BEIR Results.

Table 5 reports nDCG@10 results on BEIR datasets with fewer than 2,000 queries. We sample 100 queries per dataset.

¹<https://github.com/axolotl-ai-cloud/axolotl>

C TS-SetRank-T: Throughput-Optimized Variant

While the TS-SetRank algorithm (Algorithm 1) in the main paper outlines our general two-phase Bayesian inference strategy, we also introduce TS-SetRank-T, a throughput-optimized variant designed for environments with batched or delayed LLM calls to increase the parallelism in Phase II.

Unlike TS-SetRank, which performs posterior updates immediately after each batch, TS-SetRank-T (Algorithm 2) aggregates binary feedback across multiple rounds before updating. The update interval τ controls how frequently the posterior is updated, effectively governing the number of queries that can be executed concurrently in Phase II. In Phase I, all inference rounds are trivially parallelizable due to the uniform sampling strategy.

Topic	BM25s	Heapify	Uniform	TS-75/25	TS-50/50	TS-25/75	TS-0/100
AoPS	0.01582	0.0158	0.017 ±0.003	0.013 ±0.001	0.016 ±0.003	0.014 ±0.004	0.011 ±0.001
Biology	0.44682	0.4800	0.480 ±0.007	0.497 ±0.006	0.493 ±0.015	0.480 ±0.006	0.489 ±0.012
Earth Science	0.52155	0.5140	0.459 ±0.013	0.462 ±0.015	0.458 ±0.007	0.466 ±0.017	0.453 ±0.004
Economics	0.27620	0.3030	0.276 ±0.005	0.300 ±0.006	0.289 ±0.014	0.284 ±0.009	0.294 ±0.008
Psychology	0.36296	0.4050	0.439 ±0.008	0.468 ±0.004	0.464 ±0.005	0.476 ±0.006	0.463 ±0.003
Robotics	0.16064	0.2040	0.281 ±0.013	0.291 ±0.007	0.290 ±0.009	0.292 ±0.003	0.286 ±0.010
Stack Overflow	0.27077	0.2760	0.275 ±0.006	0.292 ±0.009	0.294 ±0.012	0.299 ±0.013	0.297 ±0.003
Sustainable Living	0.23136	0.2820	0.373 ±0.012	0.384 ±0.010	0.390 ±0.010	0.383 ±0.004	0.374 ±0.002
Leetcode	0.12262	0.1160	0.164 ±0.008	0.144 ±0.007	0.138 ±0.014	0.129 ±0.003	0.119 ±0.005
Pony	0.08560	0.0879	0.189 ±0.002	0.201 ±0.002	0.199 ±0.004	0.200 ±0.003	0.199 ±0.003
TheoremQA-Q	0.11721	0.1150	0.127 ±0.002	0.117 ±0.004	0.113 ±0.003	0.104 ±0.003	0.086 ±0.004
TheoremQA-T	0.20301	0.2720	0.368 ±0.014	0.360 ±0.011	0.369 ±0.004	0.353 ±0.004	0.363 ±0.010
All topics	0.23455	0.2560	0.287 ±0.003	0.294 ±0.002	0.293 ±0.003	0.290 ±0.004	0.286 ±0.002

Table 4: nDCG@10 performance under $T = 100$ on BRIGHT split by topic.

Dataset	BM25s	Heapify	Uniform	TS-75/25	TS-50/50	TS-25/75	TS-0/100
ArguAna	0.2850	0.4700	0.4850 ±0.0190	0.4640 ±0.0020	0.4530 ±0.0230	0.4620 ±0.0080	0.4710 ±0.0150
Climate-FEVER	0.1410	0.1440	0.1840 ±0.0040	0.1990 ±0.0030	0.2010 ±0.0090	0.2030 ±0.0070	0.2090 ±0.0140
DBPedia	0.3130	0.3300	0.3580 ±0.0030	0.3680 ±0.0030	0.3710 ±0.0080	0.3820 ±0.0060	0.3810 ±0.0030
FiQA-2018	0.2320	0.3090	0.3730 ±0.0080	0.4030 ±0.0010	0.4020 ±0.0080	0.4030 ±0.0050	0.4030 ±0.0100
NFCorpus	0.3560	0.3820	0.4040 ±0.0030	0.4010 ±0.0040	0.3980 ±0.0040	0.3990 ±0.0040	0.3950 ±0.0010
SCIDOCS	0.1320	0.1380	0.1570 ±0.0050	0.1590 ±0.0010	0.1550 ±0.0020	0.1630 ±0.0020	0.1480 ±0.0110
SciFact	0.7200	0.7490	0.7330 ±0.0210	0.7550 ±0.0130	0.7460 ±0.0140	0.7540 ±0.0500	0.7430 ±0.0200
TREC-COVID	0.5930	0.6950	0.8060 ±0.0060	0.8210 ±0.0090	0.8110 ±0.0060	0.8200 ±0.0100	0.8110 ±0.0120
Touche-2020	0.4410	0.4570	0.2910 ±0.0250	0.2940 ±0.0190	0.2820 ±0.0110	0.2950 ±0.0250	0.2940 ±0.0090
Average	0.3570	0.4080	0.421 ±0.003	0.429 ±0.002	0.424 ±0.004	0.431 ±0.010	0.428 ±0.005

Table 5: nDCG@10 performance under $T = 100$ on BEIR split by dataset.

Algorithm 2: TS-SetRank-T (Throughput-Optimized)

Input: First-stage retrieval results

$D = \{d_1, \dots, d_N\}$, query q , budget
 T , batch size b , exploration rounds
 T_f , update interval τ , reranking
model M

Output: Final reranked list of documents

foreach $d_i \in D$ **do**

┌ Initialize posterior: $\alpha_i \leftarrow 1, \beta_i \leftarrow 1$
└ Initialize counters: $S_i \leftarrow 0, F_i \leftarrow 0$

for $t = 1$ **to** T **do**

┌ **if** $t \leq T_f$ **then**

└ Sample batch
└ $S_t \sim \text{Uniform}(D_b(D))$

else

┌ **foreach** $d_i \in D$ **do**

└ Draw $\tilde{\theta}_i \sim \text{Beta}(\alpha_i, \beta_i)$

└ Let $S_t \leftarrow \{d_i \in D :$
└ $\tilde{\theta}_i \text{ is among the top-}b\}$

└ Obtain feedback: $R_t := R(q, S_t)$ from
└ reranking model M

└ **foreach** $d_i \in S_t$ **do**

└ **if** $d_i \in R_t$ **then**
└ $S_i \leftarrow S_i + 1$

└ **else**
└ $F_i \leftarrow F_i + 1$

└ **if** $t > T_f$ **and** $(t - T_f) \bmod \tau = 0$ **then**

└ **foreach** $d_i \in D$ **do**

└ $\alpha_i \leftarrow \alpha_i + S_i, \beta_i \leftarrow \beta_i + F_i$
└ $S_i \leftarrow 0, F_i \leftarrow 0$

foreach $d_i \in D$ **do**

┌ Final update: $\alpha_i \leftarrow \alpha_i + S_i,$
└ $\beta_i \leftarrow \beta_i + F_i$

return documents sorted by posterior means

$\hat{\theta}_i = \alpha_i / (\alpha_i + \beta_i)$ in descending order.
