Inference Scaling for Bridging Retrieval and Augmented Generation

Youngwon Lee[*] **Seung-won Hwang**[*] **Daniel Campos**
Filip Graliński **Zhewei Yao** **Yuxiong He**
Snowflake AI Research [*]Seoul National University

Abstract

Retrieval-augmented generation (RAG) has emerged as a popular approach to steering the output of a large language model (LLM) by incorporating retrieved contexts as inputs. However, existing work observed the generator bias, such that improving the retrieval results may negatively affect the outcome. In this work, we show such bias can be mitigated, from inference scaling, aggregating inference calls from the permuted order of retrieved contexts. The proposed Mixture-of-Intervention (MoI) explicitly models the debiased utility of each passage with multiple forward passes to construct a new ranking. We also show that MoI can leverage the retriever's prior knowledge to reduce the computational cost by minimizing the number of permutations considered and lowering the cost per LLM call. We showcase the effectiveness of MoI on diverse RAG tasks, improving ROUGE-L on MS MARCO and EM on HotpotQA benchmarks by $\sim$ 7 points.

1 Introduction

Retrieval-Augmented Generation (RAG) has become a widely adopted strategy to address core limitations of large language models (LLMs), such as hallucinations or restricted generalization to topics, concepts, or ideas that were not covered during training, by presenting relevant information to ground generation (Gao et al., 2023).

However, existing work observed the generator bias, such that improving the retrieval results may negatively affect the outcome. As a bridge, Figure 1 demonstrates the use of reranker: However, RankGPT (Sun et al., 2023), a widely adopted reranker based on prompting LLMs, improves the retrieval quality but negatively impacts RAG performance on the MS MARCO benchmark (Bajaj et al., 2018). Even worse, employing a stronger backbone LLM for reranking worsens the quality further. These unexpected results suggest that R's objective of maximizing relevance may not always produce optimal outputs. Meanwhile, training a dedicated bridge module for bridging such gap has been studied (Ke et al., 2024), which requires costly heuristic-based annotation to build the train set.

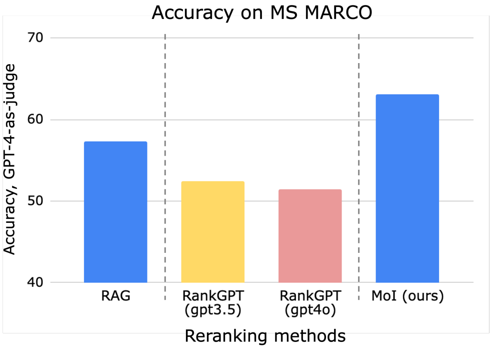

Figure 1: (Left, RAG) Top-10 passages retrieved by a complex retrieval system involving the Bing search engine are fed to the generator LLM. (Center) RankGPT, a strong reranker based on LLM, hurts the performance, even more severely with stronger backbone. (Right) MoI improves the answer quality, outperforming RAG (without reranking) by a large margin of 6 points in accuracy.

An alternative train-free approach is inference scaling, by aggregating generation from the *permutations* of the retrieved results. This strategy, known as self-consistency (Wang et al., 2023a), uses the number of permutations with consistent generation as a proxy for quality. We refer to this as a Mixture-of-Agents (Wang et al. (2024a); MoA) baseline: Figure 2A depicts MoA aggregates **black-box** outputs from parallel, independent agent calls to AG, each fed with differently permuted retrieved results, to choose the output A, which is more consistently supported.

Unlike MoA, which uses multiple calls solely for consistency voting, we leverage these calls to ob-

[*]Work done while visiting Snowflake. Correspondence to: seungwonh@snu.ac.kr.

Findings of the Association for Computational Linguistics:
NAACL 2025, pages 7339–7354
April 29 - May 4, 2025 ©2025 Association for Computational Linguistics

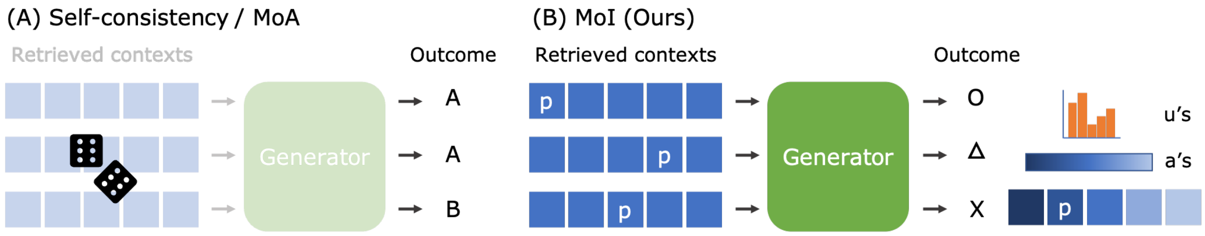

Figure 2: (A, baseline) Self-consistency (Wang et al., 2023a) and MoA (Wang et al., 2024a) treat random permutations of passages as black-box and count the consistency vote for outcomes. (B, proposed) In MoI, permutations are treated as white-box intervention of one another, such that, from the obserevations of p in varying positions, MoI estimates the effect of each passage on generation u along the impact of position bias a. Finally, the ordering based on debiased utility u is used for generation.

serve the same passage in varying positions. This allows us to directly capture **position bias**—the LLM's disproportionate weighting of input contexts based on their relative position.

Specifically, MoI distinguishes two key factors: the true utility of each passage (u) and the effect of position bias (a), enabling debiased re-ranking of retrieved contexts.

For example, Figure 2B visualizes how we predicted bias: darker colors in a represent stronger attention to passages in front positions, indicating p's contribution to the outcome is overemphasized due to its position. This aligns with the "lost-in-the-middle" bias observed in prior work (Liu et al., 2024). The final ranking, adjusted by debiased utility, moves p to second position. Empirically, this reranking leads to the generator producing higher-quality answers.

Our contributions address the following research questions through the development of MoI:

- **(RQ1)** How can the debiased utility and ranking be determined from multiple inference calls?
- **(RQ2)** Can MoI match the effectiveness of black-box MoA scaling, which requires inference calls for all permutations, while using fewer observations?
- **(RQ3)** How can we reduce the inference cost per call, for example, by using a smaller model or input?

which can be summarized as follows:

- We demonstrate that enhancing the retriever or generator alone may not improve RAG, thereby highlighting the need for the bridge.
- We propose a method to intervene in the ordering of retrieved contexts by explicitly modeling LLM position bias and aggregating diverse observations.
- We show that the ranking determined by MoI improves downstream RAG task performance, leveraging retriever prior for efficient and effective intervention.

2 Related Work

This section overviews existing work on the bias mitigation in RAG.

2.1 Mitigating Bias in RAG

Our observation of RAG bias in Figure 1 was consistently made in (Izacard et al., 2023; Lin et al., 2023; Izacard and Grave, 2021). claiming the improved retrieval may not improve RAG. A widely adopted explanation is position bias, also known as "lost-in-the-middle" (Liu et al., 2024) problem, of the generator considering the passage in the middle less significantly.

Modifying the generator For dealing with such bias, a common approach has been modifying the generator LLM, often jointly trained with the retriever as well (Izacard et al., 2023). Alternatively, positional embeddings and attention matrices have been manipulated to debias (Wang et al., 2024b; Ratner et al., 2023), often aiming for complete order invariance. However, as LLMs were never exposed to such manipulated embeddings or attention weights/masks during training, they may suffer from unexpected degradation in performance, such as multi-hop reasoning capabilities (Yang et al., 2023). Recently, Hsieh et al. (2024) also studied modifying the generator side, using the average attention weights assigned to passages to detect and account for bias.

Training bridge Among solutions, our work is most closely related to Ke et al. (2024), training a 'bridge' model between the retriever and generator, by selecting an ordered subset of retrieved passages.

Blackbox inference scaling When retraining retriever or generator, or jointly both is not feasible, a widespread approach is to rely on inference-time scaling, such as self-consistency (Wang et al., 2023a) or Minimum Bayes-Risk decoding (Kumar and Byrne, 2004) mechanism. For example, Tang et al. (2024) have generated several hypothesis rankings from different permutations of passages as inputs and then selected the one closest to other rankings in IR reranking task.

Our distinction Our method is whitebox inference scaling that can be interpreted as implementing bridge mechanism without training a separate bridge module, while leaving the retriever and generator intact. As such, our work is orthogonal to improving the retriever or generator, which can be combined with those approaches.

2.2 Mitigation by Mixture of Agents

As Figure 2A illustrates, self-consistency (Wang et al., 2023a) over permuted orders can mitigate bias by marginalizing the latent variables. Under a similar setting to ours, Tang et al. (2024) used self-consistency mechanism to account for the position bias for IR reranking task.

We build a MoA baseline (Wang et al., 2024a). where several LLM agents are called in parallel to independently generate an output given the same input, to hide inference latency of multiple calls. This corresponds to two phases: first *propose* phase generating output from permuted orders, and then *aggregate* to produce the final single reranked sequence of contexts.

Our distinction We view permutations as the intervention of one another, allowing **strategized proposal** phase, followed by **efficient aggregation**, where the cost of inference call is further reduced.

3 Method

3.1 Overview

Our proposed method, dubbed Mixture-of-Intervention (MoI), disentagles the *utility* u of each retrieved context, from the effect of *position bias* a to the given generator, shown by color gradations in Figure 2B. To better explain how MoI simultaneously computes both and why this is crucial, we first review how previous works obtain utility alone.

For instance, the Bayesian saliency score (Merth et al., 2024; Muennighoff, 2022) defines the following pointwise score:

$$u_p := P(p \mid q) \propto P(q \mid p)P(p), \qquad (1)$$

derived from probabilities given by the generator LLM. This score measures the saliency of passage passage p relative to query q. Note that dropping the second term $P(p)$ results in a variant used in question generation (QG; Sachan et al., 2022), which estimates how likely q would be answered by p.

However, this approach fails to account for how multiple passages collaborate in answer generation, for which, Eq. 1 can be generalized to a listwise score

$$\begin{aligned} u_p = P(p \mid q, p_1, \cdots, p_k) \propto \\ P(q \mid p_1; \cdots; p_k; p)P(p \mid p_1; \cdots; p_N). \end{aligned} \qquad (2)$$

where p_1 through p_k denote the k passages that have been *sequentially* selected with the passage with the highest listwise u_p score.

While this approach enables to model collaborative utility of p to other passages in the list, it has two shortcomings: The sequential nature of modeling listwise effect, requires $\mathcal{O}(N^2)$ number of evaluations of u_p which incurs $\mathcal{O}(N)$ latency even when provided with enough compute to parallelize. Another shortcoming is that it cannot observe how u_p changes when different passages were selected before, also due to the sequential dependencies.

MoI breaks dependency by observing p from diverse context, applying interventions of orders independently in parallel. These parallel observations enable to disentangle utility from positional bias, by aggregating the outcomes from different permutations of the passages, thus allowing the model to observe how varying the order of the passages influences the generation.

Formally, given a set of N retrieved passages $\{p_1, \cdots, p_N\}$ deemed relevant to a query q and M permutations $\pi_1, \cdots, \pi_M$ over $1, 2, \cdots, N$, we define and observe the outcome of a permutation

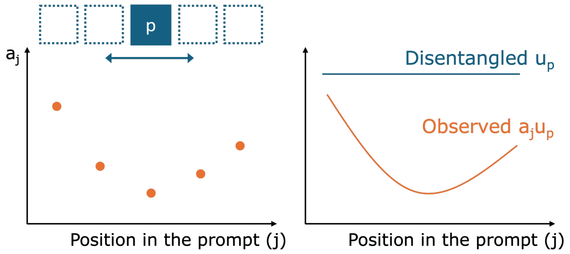

Figure 3: Ideally, wherever a passage p is placed, its contribution to generation, or utility, should be constant (blue line). However, due to position bias of LLMs, the observed orange curve varies by the position and surrounding context. MoI disentangles the effect of position bias (left figure) from observation, to determine the debiased utility u_p through multiple parallel interventions.

π_i,

$$s_i = P\left(q \mid p_{\pi_i[1]}; p_{\pi_i[2]}; \cdots; p_{\pi_i[N]}\right) \times P\left(p_{\pi_i[1]}; p_{\pi_i[2]}; \cdots; p_{\pi_i[N]}\right), \quad (3)$$

where $\pi_i[j]$ denotes the index of the passage placed at j-th position according to the i-th permutation π_i. We can see from definition that s_i, depends on (1) what other passages are in the prompt, and (2) how they are ordered in π_i.

We aim to disentangle listwise scores s_i into into two components: utility and position bias. To model this, we introduce positional bias a_j's to well predict s_i as a weighted sum for each permutation π_i:

$$\sum_{1 \leqslant j \leqslant N} a_j \cdot u_{\pi_i[j]}. \quad (4)$$

Figure 3 illustrates this idea, where the position bias of the LLM makes the contribution of a passage p in Eq. 4 vary by its relative position in the prompt. In previous works such as Liu et al. (2024), this effect was measured by moving a single gold passage to observe the outcome at different positions, while ignoring the order of other passages. MoI generalizes this idea by simultaneously determining the effects of position bias and the debiased utility u_p of each passage p, based on parallel observations from multiple passage permutations. Rather than observing $a_j u_p$ for each j and p, by moving p's relative position in the prompt, MoI aggregates the outcomes to estimate a_j's and u_p's for all j and p in Eq. 4.

In practice, we solve for u_p by minimizing the L2 loss between the predicted and observed outcomes, subject to the constraint that positional coefficients sum to 1, ensuring a valid bias distribution. Nonlinear programming solvers are used to efficiently find the optimal values for a_j and u_p:

$$\text{minimize} \sum_{1 \leqslant i \leqslant M} \left(\sum\nolimits_{1 \leqslant j \leqslant N} a_j \cdot u_{\pi_i[j]} - s_i\right)^2$$
$$\text{subject to} \sum\nolimits_j a_j = 1,\ 0 \leqslant a_j \leqslant 1.$$

After obtaining the scores, we reorder based on descending true utility u_j and feed this sequence back to the generator LLM, completing the MoI pipeline.[1]

3.2 Strategized propose phase

As contrasted in Section 2.2, we improve MoA in two phases: **proposing** permutations and **aggregating** as black-box by consistency, into **strategized propose** phase, of selecting informative orderings, and **efficient aggregate** phase, which disentangles utility and bias from the outcomes. Below, we elaborate on the implementation and potential optimizations for each phase.

3.2.1 Random samples

One extreme approach is to aggregate the entire "universe" set U of all $N!$ possible permutations. Instead, we propose randomly sampling a subset $S \subset U$, with $|S| = 3N$. This ensures that we have enough equations to solve for $2N$ variables (i.e., N for the u's and another N for the a's). Importantly, these calls can be executed in parallel, leading to an overall latency equivalent to a single call.

3.2.2 Comprehensiveness in sampling

We aim to strategize sampling by selecting a smaller but more "comprehensive" S.

Ideally, if we could map any ordering outside S to its "counterpart" in S—which better suits the generator's preferences—then considering only S would be comprehensive (Hwang and Chang, 2007), or equally effective to consider the entire universe set U. We approximate this notion by ensuring S to represent the broader landscape of U, as illustrated in Figure 4. Specifically, the shaded area indicates that permutations starting with passage 2 can be mapped to a representative permutation, $\phi^{(2)}$. We leave a formal definition of ϕ and an explanation on why $\phi^{(2)}$ can represent experiments on shaded permutations starting with 2, but the high-level intuition builds on a prior finding

[1] Empirical overhead of calling solvers was roughly 3% of the cost of a single forward pass on GPU, in terms of wall-clock time.

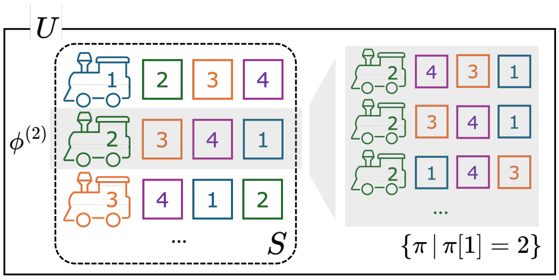

Figure 4: To approximate a comprehensive subset, we consider the set of cyclic permutations as S, encompassing diverse yet representative permutations to allow desirable ones to be surfaced.

that the first element has the greatest influence on generation (Hsieh et al., 2024; Liu et al., 2024).

Formally, we propose to pick S as the set of cyclic permutations where $|S| = N$. Desirably, (1) each passage should have equal chance of being placed at each position in S, and (2) each permutation in S should represent distinct set of permutations in U, which would map to itself. Figure 4 illustrates that our choice for S achieves both criteria: (1) it chains passages in a round-robin fashion, and (2) it ensures even coverage of all the permutations in U.

3.3 Efficient aggregate phase

Next, we explore ways to reduce the cost of each call during the aggregate phase by (a) pruning the input contexts, and (b) utilizing a smaller distilled model, or SLM, instead of an LLM, addressing our second research question.

3.3.1 Smaller input to agent

To cut down the cost of each call, we prune the input contexts by using the retriever's ranking as the reference ordering, $\phi^{(1)}$. This idea follows from Reddy et al. (2024), which demonstrated that the probability distribution over the first token accurately reflects the order intended by a reranker trained to generate passage ID sequences.

Rather than decoding the entire sequence, examining the model's prediction for the first passage ID significantly reduces costs. Similarly, we use the prefix containing the first $L < N$ passages of each permutation in S to approximate the full N. We denote the pruned permutation shifted by $k-1$ positions, in which p_k is placed at first, as:

$$\phi_p^{(k)} = [p_k, p_{k+1}, \cdots, p_{k+L-1}], \tag{5}$$

for $k \leqslant N - L$, or otherwise as

$$\phi_p^{(k)} = [p_k, \cdots .p_N, p_1, \cdots, p_{k+L-N-1}]. \tag{6}$$

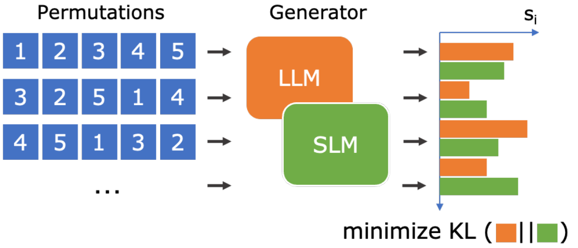

Figure 5: The distribution of s_i from an LLM is distilled to a smaller model by minimizing KL between the normalized probability distributions after softmax. Values colored orange can be pre-computed.

This pruning strategy replaces the full permutation $\phi^{(k)}$ while still preserving the essential information for generation.

3.3.2 Smaller agent

Another way to reduce the cost of each call is to delegate calls to a smaller model than the generator LLM. For this purpose, we propose preference distillation, of turning a smaller agent to align and replacing LLM, thereby featuring smaller memory and compute footprint.

First, we compute permutation-wise saliency score defined in Eq. 3 for K random permutations of passages for each query using the LLM, to construct an offline dataset. During training, we randomly select K' permutations for each query and compute $\tilde{s}_i$'s using the small model. For those K' permutations, softmax operation is applied to s_i's and $\tilde{s}_i$'s to obtain probability distributions, and then the KL divergence between the two is minimized, as described in Figure 5.

Our preference distillation enjoys the following advantages over training a bridge network Ke et al. (2024), that learns to directly output the reranked sequence of subset of passages: First, the train data preparation is much cheaper and easily parallelizable, than to repeat generating and evaluating the answer to iteratively build a pseudo-reference sequence. Second, distillation exposes the model to dense supervisory signals, as opposed to presenting a single sequence per query as a positive demonstration, or, sparse supervision. Our goal of distilling preference is more feasible than training a small model to directly output the desirable ranking, which eliminates additional round of reinforcement learning training as in Ke et al. (2024).

	MS MARCO		HotpotQA		CRAG
Ranking	R-L	GPT-4	EM	GPT-4	GPT-4
Retriever	37.75	57.28	–	–	52.43
Random	35.92	51.56	48.54	73.88	51.94
RankGPT (Sun et al., 2023)	37.53	51.45	52.22	74.88	50.49
Bayes saliency (Merth et al., 2024)	37.64	54.37	52.22	77.84	48.06
Bayes saliency + (Merth et al., 2024)	34.47	52.43	50.25	75.37	47.58
QG (Sachan et al., 2023)	36.78	55.34	48.28	73.89	53.40
LongLLMLingua (Jiang et al., 2023)	33.21	50.49	49.75	74.39	50.49
Self-consistency	38.45	58.26	51.72	76.85	54.37
MoI (Ours)	**44.30**	**63.11**	**55.67**	**79.81**	**59.23**

Table 1: Results on different question answering benchmarks with LLaMa 3 8B as the generator and various reranking methods applied. For all metrics considered, higher the better.

4 Experimental Results

4.1 Experimental settings

Tasks/benchmarks While we have mainly focused on question answering (QA) task, we also report results on other tasks, namely citation generation and fact verification. For QA benchmarks, we employed the widely used MS MARCO dataset for single-hop reasoning scenarios, HotpotQA (Yang et al., 2018) for 2-hop reasoning, and CRAG (Yang et al., 2024) for challenging multi-hop reasoning. For citation generation and fact verification task, we used TREC-RAGgy (Pradeep et al., 2024) and FEVER (Thorne et al., 2018), respectively.

For the backbone generator LLM, we used the publicly available LLaMA-3 and Phi-3 model families. Additionally, following prior work, we used greedy decoding to generate answers to ensure both efficiency and deterministic outputs.

Metrics For automatic evaluation of the generated answers, we adhered to the established evaluation protocols widely adopted for each benchmark. ROUGE-L (Lin and Och, 2004) was used for MS MARCO, and exact match (EM) for HotpotQA, both of which are reference-based metrics that compare the predicted answers to ground-truth answers based on lexical overlap. We also employed GPT-4 for automatic evaluation, following Yang et al. (2024), which allowed us to assess answer quality more flexibly by accommodating responses with minor lexical variation while maintaining the core correctness of the answer.[2] To support this decision, in Appendix C, we provide results from our user study: our finding is consistent with prior literature on LLM-as-a-judge, which showed LLM evaluation exhibits higher correlation with human judgment than traditional metrics do. The evaluation prompt can be found in Appendix E.

Baselines We have considered the following baseline methods to assess the effectiveness of MoI as a bridge between the retriever and the generator, most of which aim to rerank the passages using pointwise or listwise signals from the generator. Other than RankGPT, we have reimplemented each baseline's score computation, of which validity can be ensured from retrieval results such as in Table 7. Names are provided in accordance with those in Table 1.

- 'Retriever' uses the initial ranking from the retriever. For some benchmarks such an ordering is unavailable.
- RankGPT (Sun et al., 2023) asks an LLM to sort the passages in descending order of relevance to the query. GPT-4 (`gpt-4o`) was used as the backbone for this ranking purpose. We directly used their code[3] for running experiments.
- Bayes saliency (Merth et al., 2024) uses the Bayes saliency score defined in Eq. 1 to rank the passages. Originally, the score was used to prune irrelevant contexts.
- Bayes saliency + (Merth et al., 2024) uses the Bayes saliency score computed iteratively as in Eq. 2 to rank the passages.
- QG (Sachan et al., 2023) uses the probability the model assigns to the query conditioned on each passage as the score, i.e., $u_p = P(q \,|\, p)$.

[2] While the original scoring from Yang et al. (2024) outputs scores in the range of $[-100, 100]$, we rescale the score and report values in $[0, 100]$.

[3] `github.com/sunnweiwei/RankGPT`

	TREC-RAGgy	
Ranking	FP	FN
Retriever (BM25)	12.05	28.34
MoI (ours)	**11.40**	**24.76**

Table 2: Percentage ratio of sentences with false positive (FP) and false negative (FN) citation errors on TREC-RAGgy dev set. Metrics are lower the better.

- LongLLMLingua (Jiang et al., 2023) defines an importance score per passage as the sum of the following token-level score over the tokens in the query condition:

$$u_p = \sum_l P(q_l \mid p; q_{<l}) \log P(q_l \mid p; q_{<l}).$$

- Self-consistency (Wang et al., 2023a) considers 30 random permutations of the retrieved passages to generate 30 answers, and chooses the answer most frequently appeared. For comparison, we also reported the average score over those permutations, denoted as 'Random.'

4.2 Effectiveness of MoI

Table 1 presents downstream performance of several reranking strategies on the question-answering task, highlighting the superior performance achieved by MoI. Rerankers generally exhibit poor performance, regardless of whether they model absolute relevance (e.g., RankGPT) or use signals from the generator. In contrast, self-consistency provides consistent performance improvements across benchmarks, though the gains are smaller compared to those from MoI. We provide further qualitative analyses of the rankings determined by MoI and baselines in Appendix D.

The baselines are indeed stronger as retrievers, as shown by their retrieval accuracy (MRR) presented and discussed in more detail in Section 5: This again supports our key finding that stronger retrieval performance does not necessarily lead to higher generation quality, when compared to the standard approach of using retriever-produced rankings. This is consistent with the findings from Cuconasu et al. (2024) that adding noise to the retriever, which would make it 'weaker' as a retriever, may lead to improvements in generation quality. Our contribution is optimizing interventions towards bridging retriever and generator.

Ranking	Acc
Retriever (DPR)	83.11
Random	83.42
RankGPT	83.88
Self-consistency	84.07
MoI	**85.03**

Table 3: Fact verification performance on FEVER benchmark. We used the top-5 retrieved passages in Wang et al. (2023b).

Additionally, we demonstrate that MoI can be applied beyond its role in question-answering systems to any RAG task. To this end, we used LLaMA 3 8B as a citation generator to identify the passages supporting each sentence in a long-form response to a query on TREC RAGgy development set. Table 2 shows that the ordering of retrieved contexts (and how they are numbered for identification) also affects the output in this scenario, while MoI effectively reduces both types of errors.

We also observed consistent results on another knowledge-intensive task, fact verification, using the FEVER benchmark. Given the top-5 passages retrieved using DPR (Karpukhin et al., 2020) from Wikipedia, the generator was asked to classify the given statement as either true or false. The accuracy reported in Table 3 again validates the effectiveness of our method across various tasks, outperforming baselines.

4.3 Cost-effective proposal and aggregation

Model substitution To optimize the cost associated with intervention in MoI, we presented several designs in Section 3.2. We start by finding a balance between cost and performance through the use of a smaller substitute model as the agent. Table 4 demonstrates that replacing Phi-3 7B with an off-the-shelf Phi-3 3B retains 80% of the performance gains over the random baseline, at approximately half the cost. This can be attributed to that models from the same family generally being pre-aligned and sharing similar preferences for passage permutations, as they are often trained on the same or very similar set of preference data.

Preference distillation If a smaller model is not readily available for a given LLM, a suitable one can be created through preference distillation. Table 5 shows that the Phi-3 3B model is not effective as a direct substitute for the LLaMA 3 8B generator. However, after performing preference distillation,

	HotpotQA	
Ranking	EM	GPT-4
MoI	55.67	79.81
+ replace w/ Phi-3 3B	54.18	78.82
Random	48.36	71.64

Table 4: Replacing Phi-3 7B with Phi-3 3B cuts the cost nearly 50% while 80% of the performance improvement over the random baseline is maintained.

	HotpotQA	
Ranking	EM	GPT-4
MoI	55.67	79.81
+ replace w/ Phi-3 3B	49.26	74.39
+ distillation	53.69	79.81
Random	48.54	73.88

Table 5: Results on HotpotQA with LLaMA3 8B model as generator. While replacing it with Phi-3 3B is not effective, after preference distillation 70/100% of the gain in terms of the two metrics over the random baseline can be retained at ∼40% inference cost.

	CRAG
Ranking	GPT-4
MoI	59.23
+ Propose from cyclic	53.40
+ Pruning	54.37
+ Variable Pruning	54.86
Random	49.26

Table 6: Leveraging retriever prior in both reducing the number of calls and the cost of each call on CRAG with Phi-3 7B as generator.

	MS MARCO	
Ranking	MRR	ROUGE-L
Retriever (Bing)	.338	37.75
Bayes Saliency	.353	37.64
Question Generation	.435	36.78
RankGPT	**.634**	37.53
MoI (ours)	.464	**44.30**
Gold at 2nd	.500	40.27
Gold at 3rd	.333	39.69
Gold at 4th	.250	36.05

Table 7: Retrieval performance measured in MRR and downstream RAG performance measured in ROUGE-L. MoI outperforms others with similar or higher mean reciprocal rank, by strategically ranking the gold lower.

the Phi-3 3B model can achieve the same performance score in GPT-4 evaluations at around 40% of the inference cost. The training details are provided in Appendix A.

Retriever prior As discussed earlier, we leverage prior knowledge from the retriever for efficiency in two ways. First, we consider cyclic permutations based on the retriever's ranking to reduce the number of calls. Next, sequences are pruned to a shorter length, which reduces the cost of each call. In this process, if the scores from the retriever are also available, they can further enhance the outcome, as shown in Table 6. By adopting cyclic permutations and fixed-length pruning, we achieved 90+% cost savings while maintaining 50% of the relative performance gains compared to the random baseline, while variable pruning with retriever scores provided additional improvements.

5 Analysis

Downranking gold if desirable If there is no bias, ranking gold higher should optimize RAG output. In contrast, if there is bias, downranking a relevant passage, and an effective debiasing algorithm should identify downrankings that may improve output accuracy. Table 7 shows that, in the rank determined by MoI, the gold passage does not necessarily surface higher, yet this still results in the generation of more accurate answers. The performance of MoI and baseline methods is also compared to scenarios where gold passages are consistently placed in certain positions; notably, MoI outperforms methods that achieve similar average gold passage rankings.

Optimality of ranking In line with the spirit of studying the 'reversal curse,' suggesting LLM's ability to process reversed inputs would drop significantly when the original input order is desirable (Berglund et al., 2024), we explored reversing the ranking of contexts. Our findings reveal a significantly greater performance drop with our method compared to baseline approaches. As shown in Table 8, reversing the sequence identified by MoI results in an 18-point drop in EM, while for RankGPT, the decrease is less than 5 points. This shows the ranking identified by MoI through intervention is ideal, such that adversarially perturbing by reversing the rank would harm the performance greatly.

	MS MARCO	
Ranking	EM	GPT-4
Random	35.92	51.56
RankGPT	37.53	51.45
RankGPT (reversed)	32.98	44.67
MoI	44.30	63.11
MoI (reversed)	26.26	39.81

Table 8: Not only the debiased ranking found by MoI leads to better performance with large margin, it exhibits higher polarity, incurring notable performance degradation when the ordering is reversed.

	HotpotQA	
Ranking	EM	GPT-4
Random	53.05	78.89
Self-consistency	54.68	80.79
MoI	**56.65**	**82.27**

Table 9: Results on HotpotQA with LLaMA-3 70B as the backbone LLM.

Effect of model scale We provide evidence that larger models still suffer from position bias and can benefit from applying MoI as well. As shown in Table 9, we observe consistent results with LLaMA-3 70B as the backbone LLM.

Quantified position bias Figure 6 illustrates the average values of positional coefficients a_j across different models, showing a monotonically decreasing trend as the passage's position moves from the beginning to the end of the prompt. This quantifies a significant position bias, showing earlier passages contribute more to the final generated output.

Tasks with natural inductive bias For tasks involving reasoning chains, the order the evidence appears may play an important role in the generation quality (Chen et al., 2024). While we have showed that MoI also work well for multi-hop reasoning scenarios on HotpotQA, here we provide more detailed discussion regarding the compatability of multi-hop reasoning and MoI.

To this end, we first identified cases with dependency, that is, those define a 'natural order' of two subquestions, or corresponding gold passages in HotpotQA. As a proxy for categorizing dependent subquestions, we prompted GPT-4 to decompose each query into two subquestions then categorize those with dependencies. We found that approximately 1/4 of HotpotQA queries were non-dependent, meaning the two subquestions could be answered in any order. On the remaining subset with dependencies, our method still demonstrated significant improvements over the random baseline, as shown in Table 10, which suggests a degree of robustness with respect to dependency. However, further investigation is needed for datasets with stronger dependencies, such as questioning temporal dependencies, as well as for more accurate categorization of such scenarios.

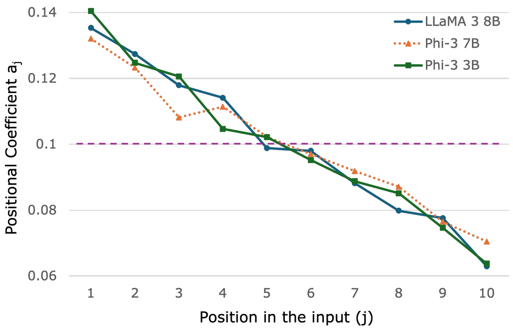

Figure 6: The values of computed positional coefficients a_j's for each position j, averaged across datapoints for different models on HotpotQA. Dashed violet line represents the ideal case of zero position bias.

	HotpotQA-dep	
Ranking	EM	GPT-4
Random	45.79	71.67
MoI	**51.92**	**77.57**

Table 10: QA performance on HotpotQA subset of queries with inherent sequential dependency between the decomposed subquestions.

The prompt used for decomposition and categorization can be found in Appendix E.

6 Conclusion

We proposed MoI, a novel inference-time scaling method for bridging the retriever and generator in RAG. By modeling the position bias of LLMs from aggregated observations over multiple interventions, MoI disentangles the impact of position from utility, enabling it to determine a debiased ranking of the contexts. We also demonstrated that leveraging the retriever's prior knowledge can reduce the search space of permutations, lowering both the number of LLM calls and the cost of each call. Finally, we showcased the effectiveness of MoI across several benchmarks in question answering and other RAG tasks.

Limitations

While we have presented results with LLaMA-3 70B as the generator in Section 5, experiments with more capable and sophisticated models are further needed to deepening our understanding of the sensitivity to input ordering of LLMs in RAG.

In addition, the proposed method increases inference compute usage as it invokes multiple forward passes for intervention. However, there are many scenarios in which improving the performance is of more critical consideration than saving inference compute , e.g., healthcare. We also discussed budget-constrained scenarios, for which we reduce both the number and latency of invocations in Section 3.2.

Meanwhile, in extreme scenarios where only one invocation is allowed, intervention can be moved to training time, replacing inference cost with training compute for similar gains. This can be promising future directions and we leave it as next work.

References

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. 2018. Ms marco: A human generated machine reading comprehension dataset. *Preprint*, arXiv:1611.09268.

Lukas Berglund, Meg Tong, Maximilian Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz Korbak, and Owain Evans. 2024. The reversal curse: Llms trained on "a is b" fail to learn "b is a". In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Xinyun Chen, Ryan A. Chi, Xuezhi Wang, and Denny Zhou. 2024. Premise order matters in reasoning with large language models. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Florin Cuconasu, Giovanni Trappolini, Federico Siciliano, Simone Filice, Cesare Campagnano, Yoelle Maarek, Nicola Tonellotto, and Fabrizio Silvestri. 2024. The power of noise: Redefining retrieval for RAG systems. *CoRR*, abs/2401.14887.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *Preprint*, arXiv:2312.10997.

Cheng-Yu Hsieh, Yung-Sung Chuang, Chun-Liang Li, Zifeng Wang, Long Le, Abhishek Kumar, James Glass, Alexander Ratner, Chen-Yu Lee, Ranjay Krishna, and Tomas Pfister. 2024. Found in the middle: Calibrating positional attention bias improves long context utilization. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 14982–14995, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Seung-won Hwang and Kevin Chen-Chuan Chang. 2007. Optimizing top-k queries for middleware access: A unified cost-based approach. *ACM Trans. Database Syst.*, 32(1):5.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 874–880, Online. Association for Computational Linguistics.

Gautier Izacard, Patrick S. H. Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2023. Atlas: Few-shot learning with retrieval augmented language models. *J. Mach. Learn. Res.*, 24:251:1–251:43.

Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023. Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression. *CoRR*, abs/2310.06839.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Zixuan Ke, Weize Kong, Cheng Li, Mingyang Zhang, Qiaozhu Mei, and Michael Bendersky. 2024. Bridging the preference gap between retrievers and LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10438–10451, Bangkok, Thailand. Association for Computational Linguistics.

Shankar Kumar and William Byrne. 2004. Minimum Bayes-risk decoding for statistical machine translation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, pages 169–176, Boston, Massachusetts, USA. Association for Computational Linguistics.

Chin-Yew Lin and Franz Josef Och. 2004. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 605–612, Barcelona, Spain.

Xi Victoria Lin, Xilun Chen, Mingda Chen, Weijia Shi, Maria Lomeli, Rich James, Pedro Rodriguez, Jacob Kahn, Gergely Szilvasy, Mike Lewis, Luke Zettlemoyer, and Scott Yih. 2023. RA-DIT: retrieval-augmented dual instruction tuning. *CoRR*, abs/2310.01352.

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

Thomas Merth, Qichen Fu, Mohammad Rastegari, and Mahyar Najibi. 2024. Superposition prompting: Improving and accelerating retrieval-augmented generation. *Preprint*, arXiv:2404.06910.

Niklas Muennighoff. 2022. SGPT: GPT sentence embeddings for semantic search. *CoRR*, abs/2202.08904.

Ronak Pradeep, Nandan Thakur, Sahel Sharifymoghaddam, Eric Zhang, Ryan Nguyen, Daniel Campos, Nick Craswell, and Jimmy Lin. 2024. Ragnarök: A reusable rag framework and baselines for trec 2024 retrieval-augmented generation track. *Preprint*, arXiv:2406.16828.

Nir Ratner, Yoav Levine, Yonatan Belinkov, Ori Ram, Inbal Magar, Omri Abend, Ehud Karpas, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. Parallel context windows for large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6383–6402, Toronto, Canada. Association for Computational Linguistics.

Revanth Gangi Reddy, JaeHyeok Doo, Yifei Xu, Md Arafat Sultan, Deevya Swain, Avirup Sil, and Heng Ji. 2024. First: Faster improved listwise reranking with single token decoding. *Preprint*, arXiv:2406.15657.

Devendra Sachan, Mike Lewis, Mandar Joshi, Armen Aghajanyan, Wen-tau Yih, Joelle Pineau, and Luke Zettlemoyer. 2022. Improving passage retrieval with zero-shot question generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3781–3797, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Devendra Singh Sachan, Mike Lewis, Dani Yogatama, Luke Zettlemoyer, Joelle Pineau, and Manzil Zaheer. 2023. Questions are all you need to train a dense passage retriever. *Transactions of the Association for Computational Linguistics*, 11:600–616.

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.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, New Orleans, Louisiana. Association for Computational Linguistics.

Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. 2024a. Mixture-of-agents enhances large language model capabilities. *Preprint*, arXiv:2406.04692.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023a. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Zhiruo Wang, Jun Araki, Zhengbao Jiang, Md. Rizwan Parvez, and Graham Neubig. 2023b. Learning to filter context for retrieval-augmented generation. *CoRR*, abs/2311.08377.

Ziqi Wang, Hanlin Zhang, Xiner Li, Kuan-Hao Huang, Chi Han, Shuiwang Ji, Sham M. Kakade, Hao Peng, and Heng Ji. 2024b. Eliminating position bias of language models: A mechanistic approach. *Preprint*, arXiv:2407.01100.

Kejuan Yang, Xiao Liu, Kaiwen Men, Aohan Zeng, Yuxiao Dong, and Jie Tang. 2023. Revisiting parallel context windows: A frustratingly simple alternative and chain-of-thought deterioration. *CoRR*, abs/2305.15262.

Xiao Yang, Kai Sun, Hao Xin, Yushi Sun, Nikita Bhalla, Xiangsen Chen, Sajal Choudhary, Rongze Daniel Gui, Ziran Will Jiang, Ziyu Jiang, Lingkun Kong, Brian Moran, Jiaqi Wang, Yifan Ethan Xu, An Yan, Chenyu Yang, Eting Yuan, Hanwen Zha, Nan Tang, Lei Chen, Nicolas Scheffer, Yue Liu, Nirav Shah, Rakesh Wanga, Anuj Kumar, Wen tau Yih, and Xin Luna Dong. 2024. Crag – comprehensive rag benchmark. *Preprint*, arXiv:2406.04744.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages

2369–2380, Brussels, Belgium. Association for Computational Linguistics.

A Implementation Details

For main experiments we have used LLaMA 3 8B Instruct, Phi-3 mini (3B) and small (7B) models available on huggingface as backbone models. As mentioned in Section 4.1, we have employed greedy decoding for generating the answer.

For preference distillation, we annotated about 20k examples in HotpotQA train set using the teacher model, scoring $K = 30$ random permutations of the passages per query to build an offline preference dataset. The student model, Phi-3 3B, was trained with LoRA at bf16 precision. The relevant hyperparameter configuration was as follows: for LoRA related settings, we used rank of $r = 8$, $\alpha = 32$, and dropout 0.1. For general configuration, we used learning rate of 1e-4, effective batch size of 4; we trained the model for 5 epochs with weight decay of 0.01 applied. We did not conduct hyperparameter search to determine these values, which leaves further rooms for improvement by performing one to find a better recipe.

Preference distillation did not introduce any degenerating behavior to the student model, such as a notable drop in QA performance.

B Comprehensive Sampling

To argue comprehensiveness of S more formally, we formalize cyclic permutations for N passages to refer to the following set of permutations

$$S = \left\{ \phi^{(k)} \,\middle|\, 1 \leqslant k \leqslant N \right\} \tag{7}$$

where ϕ refers to some referential ordering

$$\phi = \phi^{(1)} = [p_1, \cdots, p_N] \tag{8}$$

and $\phi^{(k)}$ denotes a permutation in which passages are shifted left by $k - 1$ so that p_k is placed at the beginning, that is,

$$\phi^{(k)} = [p_k, p_{k+1}, \cdots, p_N, p_1, \cdots, p_{k-1}] \tag{9}$$

for $k > 1$. For example, a cyclic permutation by $2 = 3 - 1$ position to the left would give

$$\phi^{(3)} = [p_3, p_4, p_5, p_1, p_2] \tag{10}$$

for $N = 5$.

Here, the mapping $\mathcal{M} : U \to S$ from any permutation ϕ in U to an element in S is given as all permutations starting with the same passage. This divides U into N non-empty and disjoint subsets, each of which maps to $\phi^{(1)}, \ldots, \phi^{(N)}$, respectively.

Figure 7 illustrates these concepts again as in Figure 4, in which the permutations starting with passage 2 as the first item are all mapped to $\phi^{(2)}$ to form the subset S. In order to confirm the common finding from previous literature that the first passage exerts the highest influence on generation, we show that the average distance between (two) permutations is closer in each partition, than between partitions. The distance between two permutations π_1 and π_2 was measured by the L1 distance between two probability distributions, namely the generator's prediction on the first token of the response given the permutations:

$$d(\pi_1, \pi_2) = \sum_{y_1 \in \mathcal{V}} |P(y_1 \,|\, \pi_1) - P(y_1 \,|\, \pi_2).| \tag{11}$$

This distance captures how similar the model prediction would be given two different permutations of the same set of passages, suggesting that a permutation close to another can replace it without altering the generator's prediction greatly.

C Soundness of GPT-4 Evaluation

Metric	Score
Kohen's κ	.874
Kendall's τ	.828
Fleiss' κ	.694

Table 11: Agreement between human annotators and human-LLM judgment.

We conducted a small-scale human study on the soundness of evaluation using GPT-4 and obtained results showing GPT-4's evaluation is indeed highly correlated to human judgment as presented in Table 11. For obtaining Table 11, 3 annotators were tasked with classifying 100 samples from MS MARCO as correct or incorrect, by comparing model generated responses against the ground truths. We report the agreement between this human judgment and GPT-4's evaluation used in our paper, alongside the inter-annotator agreement, where strong agreement is indicated in all cases. Human-GPT-4 agreement was measured by Cohen's kappa after majority voting and Kendall's tau correlation after soft label aggregation, while the inter-annotator agreement was measured by Fleiss' kappa.

D Qualitative Analysis

Table 12 shows an example of a winning case for MoI compared to baseline methods, where it up-

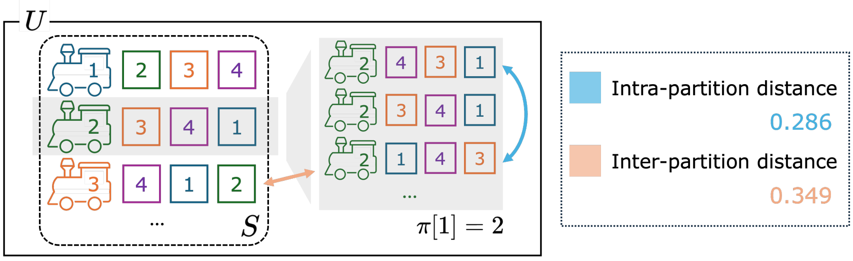

Figure 7: The set of all permutations U can be partitioned into disjoint subsets based on the first item. Distance between two permutations can be measured by the L1 distance between the generator's predicted probability distribution on the first token of the response. Permutations from the same partition exhibit smaller distance in between in average, compared to permutations from different partitions.

ranks the gold passage to produce the correct answer. Among all the winning cases in pairwise comparison with Bayes saliency or Retriever baseline in terms of GPT-4 evaluation, about 79% fall into this scenario, mainly accounting for the performance gain with MoI.

In this example, none of the passages directly mentioned the entity 'graduate marketers' as it appears on the query, but the proposed method successfully resolved it as 'MBA graduates' rather than 'marketing managers' or similar ones, thanks to its approach of producing a passage score that is aware of the whole context by considering several permutations to mitigate the position bias. In contrast, the pointwise baselines which predicts the relevance of each passage to the query separately fail to prevent passages about entities like 'marketing managers' ranked higher, leading to undesirable answers. Also, pointwise baselines are more prone to noises in presentation of passages such as repetition, as evidenced by p_6 in this example, which consists of a near duplicate of a three-sentence chunk is ranked the highest by the Bayes saliency method. After removing the repetition, the rank determined by the Bayes saliency method drastically changes, while still leading to an incorrect answer.

On the other hand, Table 13 shows another example where the proposed method wins while not necessarily upranks the gold passage. Specifically, the proposed method again avoids highlighting *distracting* passages, that would lead the generator to producing wrongful answers by effectively considering the whole passages. Due to its high lexical similarity, the distracting passage (p_{10}) was paid undesirable attention by the pointwise baseline.

E LLM Prompt

Here, we provide the prompts used for LLM-as-a-judge evaluation and question decomposition, in Figure 8 and 9, respectively. For evaluating accuracy, after obtaining correctness of the answer, those classified as 'False' were assigned different score based on whether the model response was 'I don't know' or not.

Question: How much would *graduate marketers* make?
Answer from Ours: (Correct) According to the passages, the average starting salary for MBA graduates can be as high as **\$110,000 to \$130,000** per year . . . *Top-3 Passages from Ours:* p_3: At present, the average starting salary for **MBA graduates** can be as high as **\$110,000 to \$130,000** . . . p_7: A general brand manager will make about \$5,000 less than a package design manager, . . . p_5: The BLS reports that the upper 10 percent of **marketing managers** earned more than \$187,200, as did managers . . .
Answer from Bayes saliency: (Wrong) According to the passages, **graphic designers** in the top 10 percent made more than **\$77,370**, and **marketing managers** . . . *Top-3 Passages from Bayes saliency:* p_6: . . . **Graphic designers** in the top 10 percent made more than **\$77,370**, . . . p_2: **Marketing managers** earned a median annual wage of \$123,220 in 2013, . . . p_7: A general brand manager will make about \$5,000 less than a package design manager, ...
Answer from R-then-G: (Wrong) According to the passages, **marketing managers** earned a median annual wage of **\$123,220** in 2013, . . . *Top-3 Passages from retriever:* p_1: **Marketing managers** made salaries ranging from about **\$58,000 to \$166,000** per year in 2010, . . . p_2: **Marketing managers** earned a median annual wage of **\$123,220** in 2013, . . . p_3: At present, the average starting salary for **MBA graduates** can be as high as **\$110,000 to \$130,000** . . .

Table 12: An example from MS MARCO development set where ours produces the correct answer as it upranks the gold passage p_2. Models were provided with all the 10 passages to generate the answer, while due to space limit the top-3 of them are presented here. The subscript identifying each passage is the rank of the passage determined by the retriever, Bing search engine in this case.

Question: How (many) ounces in (a) cup?
Answer from Ours: (Correct) **8** fluid ounces to a cup. *Top-3 Passages from Ours:* p_5: ... The mark at **8** fluid ounces indicates 1 cup. For **8** fluid ounces, use a measuring cup. . . . p_6: **8** fluid ounces to a cup. This is liquid measure. However the 16 fluid ounces that make the pint . . . p_7: In the US, 1 cup = **8** fluid ounces (*not identical to the avoirdupois ounce which is weight) . . .
Answer from Bayes saliency: **(Distractor: highly similar lexically, but semantic outlier in the list)** There are **0.12500000001479** cup in a ounce. *Top-3 Passages from Bayes saliency:* p_9: This is a very easy to use ounces to cup converter. First of all just type the ounces (fl oz) value in the text field . . . p_3: If you asked about the ounce that is rougly 28 grams in weight, then you should realize that . . . p_{10}: There are **0.12500000001479** cup in a ounce. ...

Table 13: Another example from MS MARCO where ours produces the correct answer, while maintaining the rank of the gold passage which is not included in the top-3 passages.

Evaluation Prompt for Accuracy

# Task:
You are given a Question, a model Prediction, and a list of Ground Truth answers, judge whether the model Prediction matches any answer from the list of Ground Truth answers. Follow the instructions step by step to make a judgement.
1. If the model prediction matches any provided answers from the Ground Truth Answer list, “Accuracy” should be “True”; otherwise, “Accuracy” should be “False.”
2. If the model prediction says that it couldn’t answer the question or it doesn’t have enough information, “Accuracy” should always be “False.”
3. If the Ground Truth is “invalid question,” “Accuracy” is ‘True” only if the model prediction is exactly “invalid question.”

# Output:
Respond with only a single JSON string with an “Accuracy” field which is “True” or “False.”

Input fields are:
Question: `{question}`
Ground-truth: `{list of ground-truth answers}`
Prediction: `{model generated answer}`

Output fields are:
Accuracy: `{correctness of response}`

Figure 8: Prompt for evaluating generated answer against ground-truths. Instances classified as ‘False’ are further processed if the model responded with “I don’t know.”

Prompt for Question Decomposition and Categorization of Multi-hop Questions

You are given a complex query that can be decomposed into two subquestions. You should first decompose the original query into two subquestions, and then identify if there is a sequential dependency between the two. In other words, you should decide whether or not we must answer one of the subquestions first to be able to answer the other. Also, you will be given two passages that can together answer the original query. If there is a sequential dependency between two subquestions, order the two passages according to that order. Your final answer must be either ‘Passage 1 first’, ‘Passage 2 first’, or ‘No dependency.’

Input fields are:
Question: `{multi-hop question}`
Passage 1: `{first gold passage}`
Passage 2: `{second gold passage}`

Output fields are:
Decomposed Subquestion 1: `{subquestion that can be answered by passage 1}`
Decomposed Subquestion 2: `{subquestion that can be answered by passage 2}`
Decision: `{dependency between the two}`

Figure 9: Prompt for decomposing and identifying dependency in multi-hop questions from HotpotQA.