RoToR: Towards More Reliable Responses for Order-Invariant Inputs

Soyoung Yoon[1*] **Dongha Ahn**[12] **Youngwon Lee**[1] **Minkyu Jung**[2]
HyungJoo Jang[2] **Seung-won Hwang**[1†]
[1]Seoul National University [2]Channel Corporation
{soyoung.yoon, seungwonh}@snu.ac.kr

Abstract

Mitigating positional bias of language models (LMs) for **listwise** inputs is a well-known and important problem (e.g., lost-in-the-middle). While zero-shot order-invariant LMs have been proposed to solve this issue, their success on practical listwise problems has been limited. In this work, as a first contribution, we identify and overcome two limitations to make zero-shot invariant LMs more practical: **(1)** training and inference distribution mismatch arising from modifying positional ID assignments to enforce invariance, and **(2)** failure to adapt to mixture of order-invariant and sensitive inputs in practical listwise problems. Then, to overcome these issues we propose **(1)** RoToR, a zero-shot invariant LM for genuinely order-invariant inputs with minimal modifications of positional IDs, and **(2)** Selective Routing, an adaptive framework that handles both order-invariant and order-sensitive inputs in listwise tasks. On the Lost in the middle (LitM), Knowledge Graph QA (KGQA), and MMLU benchmarks, we show that RoToR with Selective Routing can effectively handle practical listwise input tasks in a zero-shot manner.[1]

1 Introduction

Language conveys meaning in part through positional information, such as word placement and sentence structure. Given this nature, Language Models (LMs) that learn from human language are trained sensitive to positional information related to the ordering of segments. However, there are some listwise inputs that require neutrality to positional information. For example, for inputs such as sets, tables, databases, or multiple-choice questions, the ordering of the input **segments**—e.g., rows in a table or elements in an unordered set—require an order-agnostic understanding. We refer to such inputs as "order-invariant inputs," on which LMs reportedly struggle. For example, in LLM-as-a-judge scenarios, LMs exhibit a preference of up to 75% for the first answer in pairwise inputs (Zheng et al., 2024b), and ranking between LMs can change up to 8 positions in different orderings of multiple choice questions on MMLU (Alzahrani et al., 2024). Such results question the reliability of LMs on order-invariant inputs. Meanwhile, existing methods for enforcing invariance to LMs showed limited effectiveness in real-world tasks, which we hypothesize to arise from the following limitations.

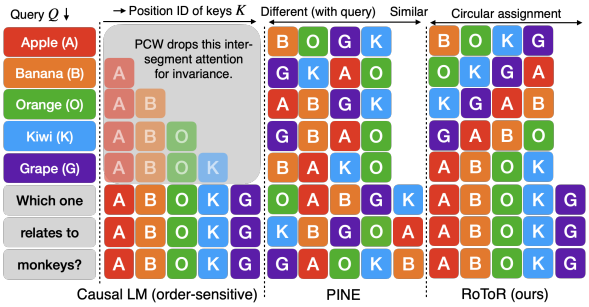


Figure 1: Self-attention alteration from order-invariant models. (a) PCW by elimination (b) PINE by re-assignment of position IDs based on query-based pairwise ordering. In contrast, (c) RoToR minimizes the distribution mismatch by global ordering with circular assignment.

First, training and inference distribution mismatch due to the positional ID re-assignment of zero-shot order-invariant LMs: Fig. 1 illustrates how self-attention is altered in these models. Unlike the original non-invariant model which always assigns position IDs in a causal, ascending manner, order-invariant models either eliminate inter-segment attention, such as PCW (Ratner et al., 2023) in Fig. 1a, or re-assign position IDs as in PINE (Wang et al., 2024) in Fig. 1b, re-ordering segments using pairwise similarity, placing simi-

[*] Work done during an internship at Channel Corporation.
[†] Corresponding author.
[1] https://github.com/soyoung97/RoToR

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 18739–18760
July 27 - August 1, 2025

lar segments closer to the query. For each query segment, it computes segment-wise query-key attention (for each attention head in each decoder layer) and re-assigns position IDs of segments as keys. This query-dependent segment ordering leads to excessively frequent alterations of positional ID assignments. Frequent re-assignments can also confuse the model and risk collisions which violate the invariance property (e.g., multiple key segments having the same similarity to a query).

To overcome this, we propose a query-agnostic global sorting with circular arrangement for order-invariant positional ID assignment. Ours is named **RoToR**, inspired by the word *rotary* to express circular assignment, and also a palindrome, to reflect order invariance. Fig. 1c contrasts with PINE in Fig. 1c, where RoToR only needs a single global ordering (e.g., `A->B->O->K->G`) with no extra attention computation. The ordering of segments on suffix tokens remains in a fixed order, since it does not rely on their similarity to the query. Finally, we propose three different global sorting algorithms for RoToR, and demonstrate that they consistently outperform previous order-invariant models.

Second, for practical listwise inputs, order-invariant tasks may partially include order-sensitive inputs that require order-specific understanding. For example, the `(d) None of the above` option in MMLU cannot be reordered. Such a "mixed" nature requires handling each of the cases adaptively, for which we propose a simple Selective Routing method. Selective Routing adapts to a given input by routing between two models, invariant and non-invariant (original), based on the confidence scores of their predictions. Experiments on the MMLU benchmark show that Selective Routing effectively handles datasets with order-invariant and sensitive inputs, and achieves better order robustness while maintaining the original performance.

In summary, our contributions are as follows: **1. Clarifying key challenges to robust understanding of listwise inputs.** We pinpoint the distribution mismatch and positional ID assignment complexities that hinder zero-shot order-invariance in LMs, and the need to adaptively handle order-invariant and order-sensitive inputs. **2. A stable, order-invariant solution (RoToR):** We propose a query-agnostic global ordering with minimal positional ID modifications, resulting in stable and efficient order-invariance. **3. Adaptive handling of listwise inputs (Selective Routing):** We introduce a simple routing method that switches between the original and invariant LMs based on confidence. On MMLU, we show that Selective Routing can adaptively deal with both types of input, leading to better stability. To this end, we aim to develop a model that excels at processing a wide range of listwise inputs reliably and efficiently.

2 Related Works

2.1 Positional bias of LLMs

Problem statement. Recent works on (zero-shot) retrieval augmented generation (RAG) with LLMs have found that the models exhibit unwanted bias on the *ordering* of the retrieved documents (Chhabra et al., 2024). Widely known as the lost-in-the-middle problem (Liu et al., 2024), many prior studies (Chen et al., 2024; Gupta et al., 2024; Pezeshkpour and Hruschka, 2023; Zhao et al., 2023; Zhou et al., 2024; Wei et al., 2024; Alzahrani et al., 2024; Zheng et al., 2024a) also investigate the impact of positional bias, extending the domain to structured knowledge grounding (SKG) tasks (Zhao et al., 2023; Zhou et al., 2024) and multiple-choice questions (Gupta et al., 2024) where changing the ordering of rows, schemas, or choices greatly degrades performance.

Considerations for decoder-only LMs. While successful approaches are presented to mitigate this issue for encoder-only (Yang et al., 2022) and encoder-decoder (Yen et al., 2024; Cai et al., 2023) models, they leave decoder-only models, which account for the current frontier LLMs, for more consideration. In contrast to transformer encoders that use bidirectional attention which is invariant by nature (Lee et al., 2019), transformer decoders use causal attention to learn causal relation signals, which is not invariant by nature (Haviv et al., 2022a). Therefore, positional bias for decoder-only models is known to stem from *both* positional encoding and causal attention mask (Yu et al., 2024; Wang et al., 2024) and is harder to mitigate.

2.2 Zero-shot order-invariance for LLMs

Long context modeling. Zero-shot approaches for mitigating positional bias in LLMs were first raised in long-context tasks, with a goal to correctly handle relevant information located in the *middle* of lengthy inputs[2]. Nonetheless, these works focus primarily on understanding long texts without losing precision (Li et al., 2023; Zhang et al., 2024a; An et al., 2023; Bai et al., 2024), whereas positional

[2] github.com/gkamradt/LLMTest_NeedleInAHaystack

bias is a more general problem that can occur even on multiple-choices questions with relatively short contexts (Alzahrani et al., 2024). Technically, this line of works modify the attention mechanism by altering the positional encoding to adapt an LLM to longer contexts (Peng et al., 2023; Hsieh et al., 2024; Peysakhovich and Lerer, 2023; Chen et al., 2023; Junqing et al., 2023; Xu et al., 2023; Yu et al., 2024; Zhang et al., 2024b). But since they do not modify the causal mask which also contributes to positional bias, order-invariance is not guaranteed in general (Haviv et al., 2022b).

(Zero-shot) order-invariance. Recent line of works focused on achieving order-invariance by mechanistically altering both positional encoding and causal masking. While several works require training (Junqing et al., 2023; Zhu et al., 2023), we focus on zero-shot approaches for practicality, namely PCW (Ratner et al., 2023), Set-Based Prompting (McIlroy-Young et al., 2024), and PINE (Wang et al., 2024), which we explain in detail at Sec. 3.1. Another line of works based on self-consistency try to mitigate positional bias simply by running inference multiple times with different orderings of contexts (Zheng et al., 2024a). However, in principle, this requires evaluating $n!$ forward passes in total, enforcing Monte Carlo approximations (Tang et al., 2024). More recent work optimizes the number or passes (Lee et al., 2025b) with similar comprehensiveness (Hwang and Chang, 2007), or replaces with contrastive training objective (Lee et al., 2025a). In contrast, our method guarantee invariance with a *single* forward pass, without requiring any approximations.

3 Methodology

3.1 Baseline: Order-invariant causal LMs

In this section, we briefly overview the existing work on endowing decoder-only models on order-invariance by adjusting attention mechanism, and review their limitations.

Isolated parallel processing Prior works like PCW (Ratner et al., 2023) and Set-Based Prompting (McIlroy-Young et al., 2024) have modified the attention mask and positional ID assignments of the language model to isolate the processing of each segment and apply same positional embeddings are applied across segments, and thus achieve order invariance: However, this design completely prevents one segment from attending to the others, and aggregating the information from different segments is solely handled at suffix and generated tokens, significantly hindering the LM's cross-segment contextualized understanding of the text. Yang et al. (2023) have argued that this essentially degenerates to mere ensemble of conditioning on each context separately. Such information bottleneck and train-test time discrepancy limits the applicability, more severely as the number of segments is increased.

Bidirectional processing with Q-K similarity A more recent work, PINE (Wang et al., 2024) has addressed these issues through a bidirectional attention mechanism that allows each segment to attend to all other segments. To achieve this within decoder-only models while maintaining order invariance, PINE dynamically modifies positional IDs based on whether a token acts as a **query** or **key** in the attention computation.

The key insight is that PINE creates an "illusion" for each query segment: it assigns the query segment the largest positional IDs among all segments, enabling it to attend to all other segments bidirectionally. The ordering of key segments is then determined by their relevance scores computed without positional embeddings ($\mathrm{Attn}_{\mathrm{NoPoS}}$), ensuring that more relevant segments appear closer to the query.

Consider the example in Fig. 2 with input `[T1'Given", S1[Apple"], S2[Ban", ana"], S3[Orange"], T6'which one", .. T10'red?"]`. The prefix token `'Given"` (T1) and suffix tokens `'which one .. red?"` (T6-T10) maintain their original positions and follow standard causal attention. For the segments S1-S3, PINE applies its order-invariant mechanism:

Dynamic positional ID assignment: When a token from segment S2 (e.g., `'ana"` at T4) acts as a **query**, PINE: (1) Assigns S2 the highest positional IDs (4-5) among all segments, placing it last. (2) Maintains internal causal order within S2: `'ban"` gets position 4, `'ana"` gets position 5. Then, it (3) Reorders other segments (S1, S3) based on their $\mathrm{Attn}_{\mathrm{NoPoS}}$ scores with S2. Conversely, when the same token acts as a **key** for another query (e.g., from S1), its position depends on the relevance score between S2 and the query segment. If $\mathrm{Attn}_{\mathrm{NoPoS}}(\mathrm{S1}, \mathrm{S2}) > \mathrm{Attn}_{\mathrm{NoPoS}}(\mathrm{S1}, \mathrm{S3})$, then S2 is placed closer to S1 than S3.

This dynamic reassignment occurs for every attention computation: each query token sees a different positional arrangement of the key tokens, determined by their relevance scores. Prefix, suffix, and generated tokens do not participate in this re-

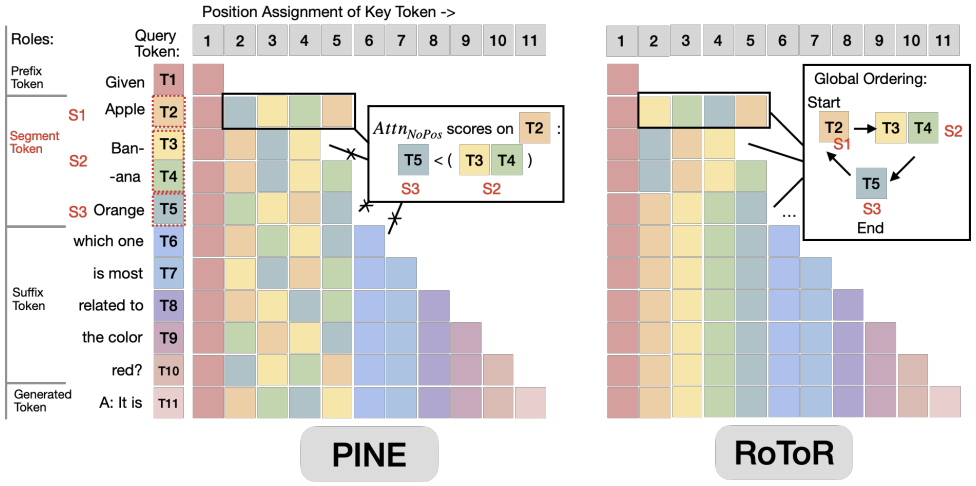


Figure 2: Attention mask and positional ID modifications for segment-wise order invariance using example input "`Given Apple Banana Orange...`". Each block represents a single token arranged by positional ID assignment. In practice, tokens retain original positions but receive reassigned positional IDs as shown. White areas indicate masked attention. **Key difference**: PINE requires per-query sorting while RoToR reuses one global ordering across all queries.

ordering and always maintain their standard causal positions. However, when these non-segment tokens act as queries, they still see the segments reordered by their relevance scores.

3.2 RoToR: minimal OOD from positional ID assignments

Challenges with PINE While PINE achieves order-invariance by contextualization across segments, its query-specific ordering scheme introduces (1) significant train-test behavior discrepancy as well as (2) unnecessary complexity and numerical instability, which limits its scalability. During decoding with PINE, position IDs are assigned differently for every query token (each token in the suffix), decoder layer, and attention head, as the query-key attention score $\mathrm{Attn}_{\mathrm{NoPos}}$ determines the position IDs. This complexity introduces **excessively frequent alterations on position IDs**: As the base LM is trained with fixed positional IDs and causal masks, this causes hidden activations higher risk of out-of-distribution (OOD) for it to process properly. Moreover, ordering segments based on attention is **computationally expensive** and introduces **numerical instability**. As computing the attention value of one query segment requires computing the KV attention over every other number of segments, PINE invokes $\mathcal{O}(n^2)$ cost overhead for each segment for input length n, which is further multiplied by the number of all combinations of layers, heads, and the number of suffix and generated tokens. Also, in practice, calculating attention without RoPE results in a very narrow range of values. `bfloat16` numeric type lacks precision to distinguish these values, leading to non-determinism originating from several tied values. The outcome may then depend on the initial ordering.

Motivation & Theoretical Foundation While investigating ways to overcome the limitations of PINE, our central goal is to preserve order invariance while minimizing the complexity of the reassignment of positions. We reason that defining a *single* global ordering scheme, not necessarily relying on attention scores, and re-using them across all queries can solve the problems stemming from query-dependent ordering. A circular assignment of a global order seems as a practical solution. The idea of using global sorting to achieve ordering invariance has been studied in set/graph ML domain (Murphy et al., 2019a,b), but to the best of our knowledge, using circular assignment of the global ordering, and the application to pre-trained language models, are our novel contributions. As a result, we propose RoToR(Fig. 3), which uses one **global ordering** that is not a function of the initial ordering of segments (e.g., canonical ordering by lexical sorting) and assigns IDs for tokens in different segments based on **circular arrangement**.

Global ordering Instead of re-computing the relative order of segments for each query, we reuse a globally shared single ordering, avoiding costly recomputation of numerically unstable attention

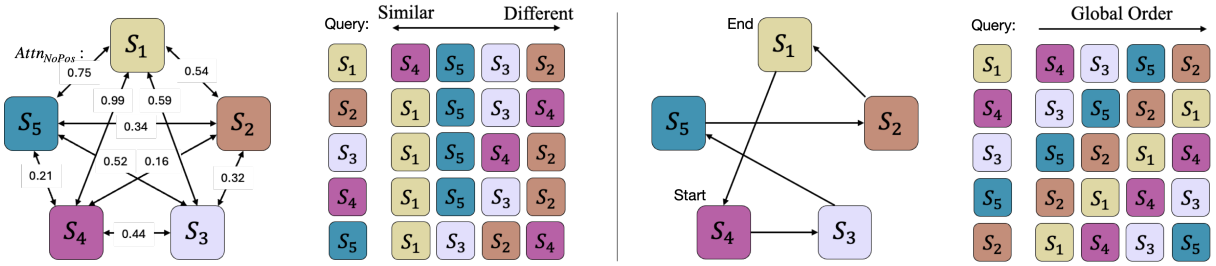


Figure 3: Comparing the ordering of 5 segments (S1 - S5) of PINE (Wang et al., 2024; left) and RoToR (Ours; right). PINE sorts segments using aggregated attention scores. In order to be fully ordering-invariant, segment sorting is changed per token in suffix level, causing confusion. In contrast, we define one global sorting of segments and conduct circular assignment between segments. With this, we simply use the global sorting for position id assignment on suffix tokens, without harming invariance.

scores. Moreover, this further reduces the gap between the LLM's pretrained behavior and test-time behavior, as consistent position IDs are assigned across layers/heads/across suffix tokens. Global ordering allows to preserve the relative placement of segments, further closing the gap induced from introducing order invariance to causal LMs. For example, in Figure 3, due to the global ordering, segments S_5 and S_2 are always placed in adjacent positions with RoToR (right side), while it is not satisfied and constantly changed with PINE (left side). We consider three separate global sorting algorithm to be used in RoToR: (1) simple **lexicographical sorting** which can be obtained with minimal overhead based on tokenized sequence of segments, (2) using a **pointwise reranker** (Nogueira et al., 2020)[3] to score relevancy of each row with respect to the question, or (3) simple **frequency**-based sorting which normalizes token ids based on the inverse frequency of each token (Details at Appendix Fig. 6). Empirically, we find that using simple lexicographical sorting is sufficient to obtain improvements over PINE.

Circular arrangement To mimic bidirectionality with causal LMs, each segment should be assigned position IDs so that they appear to themselves as being placed at the end of the sequence of segments. To achieve this with a shared global ordering, we employ circular arrangement, each segment taking turns to be placed at the end while their relative ordering is preserved. Given the global ordering, we can construct a single directed graph by combining the front and last parts. Then, we assign orderings for each segment as query by following the path from the graph, starting from the query segment, which is illustrated in Fig. 3. For all suffix and generated tokens, segments are arranged according to the initial front and last part of the global ordering. Compared to PINE where we have to assign different orderings of segments for each suffix and generated tokens, RoToR assign the same positional ID, acting merely the same as the original token. This also accounts for reducing the distributional gap between the original model.

Computational overhead We report only operations executed beyond vanilla self-attention cost $O(n^2d)$, where n is total input length, d is hidden dimension, and k is the number of segments. PINE requires two additional operations: (1) computing attention scores without rotary position embeddings ($\mathcal{O}(n^2d)$) and (2) sorting k segments for each query token ($\mathcal{O}(nk \log k)$), totaling $\boldsymbol{\mathcal{O}(n^2d + nk \log k)}$ (Wang et al., 2024)[4]. In contrast, our lexicographical sorting requires only a single global sort of k segments ($\mathcal{O}(k \log k)$), each with length $\mathcal{O}(n)$, achieving $\boldsymbol{\mathcal{O}(nk \log k)}$ and eliminating the expensive $\mathcal{O}(n^2d)$ term entirely. This can be further optimized to $\boldsymbol{\mathcal{O}(nk)}$ using radix sort.[5] We empirically validate significantly faster performance than PINE as k increases (Tab. 4).

3.3 Selective Routing for handling order-sensitive inputs

Since many practical benchmarks such as MMLU involves semi-invariant inputs, we propose a routing mechanism that uses the order-invariant model in conjunction with the standard causal model for further applicability. Our design is partly based on the finding from Wei et al. (2024) that there is

[3] castorini/monot5-base-msmarco-10k

[4] The PINE paper reports $\mathcal{O}(nk \log k)$ by absorbing the $\mathcal{O}(n^2d)$ term into baseline; we expose it explicitly for fair comparison.

[5] https://en.wikipedia.org/wiki/Radix_sort

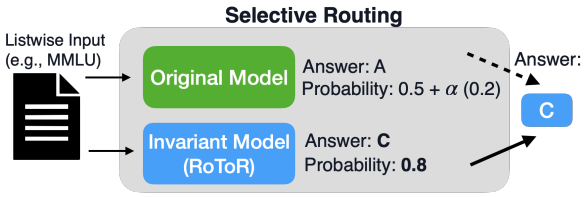


Figure 4: Illustration of Selective Routing (Sec. E).

correlation between task difficulty (which is in turn correlated with confidence values) and the model's sensitivity to ordering. Selective Routing, illustrated in Fig. 4, combines confidence, the model output probability for the generated answer, from two different model versions—the original model and the order-invariant model—on the same input and choose a more confident answer. Both models first produce a maximum probability over possible answer tokens (e.g., `A, B, C, D` for MMLU) and a corresponding answer choice. We then compare the original model's maximum probability, plus a bias term α, to the invariant model's maximum probability. If the original model's adjusted score is higher, we take its answer; otherwise, the invariant model's answer is chosen. α is a hyperparameter that controls how strongly the original model is favored, which was selected as 0.2 according to hyperparameter search on the validation subset (Appendix Sec. E).

4 Experiment setup

4.1 Baselines

Original causal LM with no modifications (Orig.) were compared, which processes text sequentially. Also, we compare RoToR against previous zero-shot order-invariant LMs discussed in Sec. 3.1, namely PCW (Ratner et al., 2023), PINE (Wang et al., 2024), and Set-Based Prompting (McIlroy-Young et al., 2024). We use the LLaMA 3.1 (AI, 2024) 8B-Instruct[6] 70B-Instruct, Qwen1.5-4B-Chat, Qwen1.5-7B-Chat[7], and Qwen1.5-72B-Chat as our backbone model for our experiments. As our method **doesn't need training**, a single A6000 GPU was sufficient to run all of the experiments except for the Llama-3.1-70B-Instruct and Qwen1.5-72B-Chat model. We also conduct experiments on a subset of benchmarks (LitM and MMLU) on the runtime latency, perplexity, and collision rate of PINE and RoToR, to further validate our claims on Sec. 3.2.

[6] `meta-llama/Meta-Llama-3.1-8B-Instruct`
[7] `Qwen/Qwen1.5-4/7B-Chat`

4.2 Benchmarks with listwise inputs

We experiment with three benchmarks involving real-world listwise input data. Examples of exact inputs and outputs are provided in Appendix G. All reported scores are rounded to the nearest tenth, except for the standard deviation (rounded to the second decimal place).

Knowledge Graph QA (KGQA) In KGQA tasks, the model takes facts over knowledge graphs represented as (subject, relation, object), and answers the given question based on the given facts. We basically follow the KGQA dataset preprocessing and evaluation setup from Baek et al. (2023), which uses Mintaka (Sen et al., 2022) with Wikidata for knowledge source, and use the Exact Match (EM), Accuracy (Acc), and F1 score metric for evaluation. We also use MPNet (Song et al., 2020) as a dense retriever to retrieve top-k facts over each question, and experiment with segment size of 30 and 50. Replication details and example dataset format are at Appendix Sec. C and Fig. 10. Along with measuring the performance of the initial input ordering, we report performance after we **shuffle** the order of segments with 3 different seeds to see shuffle robustness.

Lost in the middle (LitM) We use the Lost in the Middle (LitM) benchmark (Liu et al., 2024), which draws from 2655 queries in the Natural Questions (NQ) dataset. It provides sets of (10, 20, 30) passages, placing the gold passage at predetermined positions (e.g., 0, 4, 9) and filling the remaining slots with irrelevant passages. Following Liu et al. (2024), the `best_subspan_em` metric is used. Experiments on LitM found that eliminating the effect of index bias is another important detail for measuring true order robustness: (Appendix Sec. H). Thus, we report experiments with index bias eliminated. The exact prompt and full results including index bias is reported at Appendix Fig. 8 and Sec. A.

MMLU The Massive Multitask Language Understanding (MMLU) benchmark (Hendrycks et al., 2021) (prompts at Appendix Fig. 11) consists of 57 diverse sub-tasks with a total of 14,015 queries to measure general performance of LMs about the knowledge of the world. Despite its popularity, many works report that performance fluctuates heavily depending on the order of choices (Gupta et al., 2024; Pezeshkpour and Hruschka, 2023; Wei et al., 2024; Alzahrani et al., 2024; Zheng et al., 2024a) and is widely investigated to measure the positional bias of the model. We notice that a lot

Total ndoc (segments)	10			20					30						
Gold idx at:	0	4	9	0	4	9	14	19	0	4	9	14	19	24	29
Llama-3.1-8B-Instruct															
Original	54.7	53.0	50.2	54.8	52.6	52.8	52.4	51.0	55.6	51.5	52.4	52.8	52.1	52.3	53.0
PCW	12.4	11.9	12.2	3.7	4.0	4.0	4.0	3.9	2.3	1.8	2.0	2.0	2.1	2.0	2.0
Set-Based Prompting	42.5	42.5	42.5	26.3	26.3	26.3	26.3	26.3	14.1	14.1	14.1	14.1	14.1	14.1	14.1
PINE	58.6	58.8	59.0	56.2	55.7	55.5	55.7	55.5	54.2	54.8	54.3	53.7	54.8	54.2	54.0
RoToR-lexical	61.4	61.6	61.6	**61.4**	59.8	59.6	59.6	59.8	59.2	59.5	59.4	59.1	59.0	59.3	59.1
RoToR-reversed lexical	**61.6**	**61.8**	**61.8**	58.9	59.3	58.8	58.6	58.7	57.9	58.2	57.9	57.4	57.9	57.6	57.5
RoToR-MonoT5	61.2	61.4	61.2	60.9	**61.0**	**61.2**	**61.2**	**61.2**	**60.9**	**60.7**	**60.7**	**60.7**	**60.8**	**60.8**	**60.7**
RoToR-Freq.	61.0	61.1	61.1	60.4	60.3	58.6	60.2	60.0	59.3	60.4	59.7	59.5	59.5	59.6	59.2
Llama 3.1-70B-Instruct															
Original	66.2	65.7	65.7	65.2	64.3	65.0	66.2	64.8				–			
PINE	67.9	67.8	67.5	65.9	65.7	65.9	65.8	65.5				–			
RoToR	**69.6**	**69.5**	**69.3**	**67.6**	**67.8**	**67.8**	**67.7**	**67.9**				–			
RoToR-MonoT5	68.9	69.0	68.8	67.5	67.5	67.7	67.5	67.6				–			
Qwen1.5-4B-Chat															
Original	61.3	54.8	53.1	**59.5**	49.1	47.9	45.9	48.3	**56.8**	45.6	44.9	44.6	45.3	43.5	48.3
PINE	57.2	57.4	57.0	48.6	48.2	48.2	48.1	48.9	46.4	45.9	46.7	46.6	46.4	46.4	46.3
RoToR	58.5	58.4	58.1	49.9	49.7	49.6	49.8	49.9	44.6	44.8	44.7	44.7	44.9	44.8	44.7
RoToR-MonoT5	**58.9**	**58.5**	**58.7**	52.2	**52.1**	**52.1**	**52.2**	**52.6**	50.6	**50.7**	**50.5**	**50.6**	**50.5**	**50.6**	**50.4**
RoToR-Freq.	56.7	56.9	56.9	51.9	51.5	51.8	51.6	52.4	46.8	46.7	46.7	46.4	47.0	46.8	46.6
Qwen1.5-7B-Chat															
Original	**72.5**	63.3	62.9	**72.5**	58.5	56.1	56.0	58.2	**73.1**	58.6	55.8	53.3	53.2	52.5	57.5
PINE	65.4	65.5	66.3	59.1	59.4	59.1	58.6	59.2	58.0	55.3	55.7	56.3	55.1	55.8	56.1
RoToR	68.6	68.7	68.6	62.6	62.9	62.7	63.0	62.7	57.0	57.3	59.7	57.4	57.3	**62.8**	57.0
RoToR-MonoT5	68.8	**69.4**	**69.0**	65.2	**65.5**	**65.0**	**64.9**	**65.0**	62.6	**62.8**	**62.9**	**62.7**	**62.9**	**62.8**	**62.5**
RoToR-Freq.	68.2	68.4	68.4	62.6	62.9	62.8	62.7	62.3	59.5	59.8	59.7	59.6	59.7	59.7	59.7

Table 1: The `best_subspan_em` (%) scores on the **lost in the middle (LitM)** benchmark, with indexing bias removed, across varying numbers of documents (ndoc $\in \{10, 20, 30\}$) and models. RoToR shows the best performance across all setups. Experiments on ndoc=30 for the Llama 70B model were unable to report due to resource constraints.

of proportions consist of ordering-sensitive inputs, which showed the effectiveness of adaptively applying Selective Routing. We additionally report the average performance for all possible (4!-1) reorderings.

5 Results & Analysis

We report results for KGQA in Tab. 2, and results for MMLU in Tab. 3. Results for LitM are in Tab. 1, with a visualization in Appendix Fig. 5. We use lexical sorting for RoToR unless stated otherwise.

Effectiveness of RoToR We observe that shuffling input segments leads to non-trivial performance degradations in the original model, which exhibits a statistically significant performance drop on our experimented dataset (two-tailed t-test, $p < 0.05$, Appendix I). In contrast, our proposed RoToR model does not show a statistically significant difference in performance before and after shuffling, indicating that it is more robust against such perturbations. On LitM (Tab. 1), we notice PCW and Set-Based Prompting has impractical performance, with PINE degrading heavily as number of documents (k) increases, while RoToR is less affected. On KGQA (Tab. 2), we show RoToR outperform PINE with lower standard deviation across shuffled segments, consistent with different model architectures.

Improvements from PINE Experiments against comparing RoToR with PINE (Tab. 4) we analyze **FLOPs:** [8] RoToR consistently reduces the floating point operations overhead across segment counts and different model backbones compared to PINE. This is because RoToR does not require computing additional attention scores: it only performs tensor operations for circular arrangement. In contrast, PINE requires more cost due to attention-based reassignment. **Runtime, scalability:** Actual inference times (Appendix Sec. F) find that RoToR outperforms PINE substantially, with efficiency gains increasing alongside n. For instance, on LitM (30 docs), RoToR achieves a 43% reduction in total runtime. Practical scalability with increasing k is critical, but we find that previous order-invariant LMs struggle handling larger k (on KGQA and LitM). In contrast, RoToR shows better perfor-

[8] We used the FlopsProfiler of the DeepSpeed library to measure FLOPs.

Method	Llama-3.1-8B-Instr.			Llama-3.1-70B-Instr.			Qwen1.5-4B-Chat			Qwen1.5-7B-Chat			Qwen1.5-72B-Chat		
	Acc.	EM	F1	Acc.	EM	F1	Acc.	EM	F1	Acc.	EM	F1	Acc.	EM	F1
$N = 30$															
Initial, no shuffling of segments															
Original	50.2	44.0	51.9	61.6	57.7	63.6	30.7	27.9	34.9	31.5	27.8	35.4	41.4	37.7	43.7
PINE	51.5	45.0	52.6	63.1	58.7	64.8	31.6	28.7	35.6	32.3	28.8	36.4	46.7	42.9	49.0
RoToR	**53.1**	**46.5**	**54.1**	**63.6**	**59.1**	**65.2**	32.0	29.0	35.7	**34.3**	**29.8**	**37.7**	**47.5**	**43.2**	**49.2**
RoToR-MonoT5	51.6	45.0	52.5		–		**32.3**	29.1	**36.2**	32.9	28.4	36.3		–	
RoToR-Freq.	52.6	46.1	53.7		–		**32.3**	**29.2**	36.0	33.7	29.5	37.2		–	
After shuffling segments, averaged over 3 seeds															
Original	49.5	43.3	51.0	62.1	57.8	64.0	30.1	27.5	34.7	31.4	27.3	35.0	41.0	37.6	43.6
↪ stdev. (±)	0.07 / 0.14 / 0.17			0.37 / 0.40 / 0.27			0.41 / 0.34 / 0.43			0.26 / 0.28 / 0.29			0.75 / 0.40 / 0.33		
PINE	51.8	45.2	52.8	63.3	58.8	64.9	31.5	28.7	35.6	32.3	28.8	35.7	46.9	**43.3**	**49.2**
↪ stdev. (±)	0.05 / 0.07 / 0.16			0.13 / 0.04 / 0.10			0.20 / 0.18 / 0.13			0.17 / 0.20 / 0.13			0.18 / 0.20 / 0.20		
RoToR	**52.8**	**46.2**	**53.8**	**63.5**	**59.1**	**65.3**	31.8	28.8	35.5	**34.2**	**29.9**	**37.7**	**47.4**	43.1	49.1
↪ stdev. (±)	0.05 / 0.05 / 0.02			0.11 / 0.07 / 0.08			0.05 / 0.02 / 0.09			0.09 / 0.07 / 0.06			0.06 / 0.04 / 0.07		
RoToR-MonoT5	51.6	45.0	52.6		–		**32.4**	29.2	**36.3**	33.0	28.8	36.5		–	
↪ stdev. (±)	0.12 / 0.06 / 0.10				–		0.04 / 0.02 / 0.13			0.12 / 0.09 / 0.07				–	
RoToR-Freq.	52.5	45.9	53.5		–		32.3	**29.3**	36.0	33.8	29.6	37.4		–	
↪ stdev. (±)	0.10 / 0.15 / 0.11				–		0.13 / 0.16 / 0.09			0.04 / 0.00 / 0.09				–	
$N = 50$															
Initial, no shuffling of segments															
Original	50.0	44.0	51.7	62.6	58.5	64.5	31.6	28.6	35.8	31.7	28.0	35.7	42.1	38.7	44.5
PINE	51.6	45.1	52.6	64.1	59.8	65.8	31.6	28.8	35.3	32.0	28.5	35.9	48.0	44.1	49.9
RoToR	52.9	46.0	53.6	**64.6**	**60.0**	**66.2**	**32.7**	**29.6**	**36.2**	**34.3**	**30.1**	**38.0**	**48.4**	**44.3**	**50.3**
RoToR-MonoT5	52.4	45.4	52.8		–		32.3	29.3	35.9	32.9	28.9	36.6		–	
RoToR-Freq.	**53.1**	**46.4**	**53.7**		–		32.3	29.2	36.1	33.5	29.5	37.2		–	
After shuffling segments, averaged over 3 seeds															
Original	49.7	43.5	51.0	62.8	58.5	64.5	30.3	27.6	35.0	31.6	27.9	35.5	42.1	38.9	44.7
↪ stdev. (±)	0.34 / 0.28 / 0.46			0.29 / 0.28 / 0.05			0.26 / 0.24 / 0.35			0.40 / 0.56 / 0.42			0.30 / 0.40 / 0.35		
PINE	51.8	45.3	52.7	64.3	59.8	65.9	31.5	28.7	35.3	31.7	28.2	35.7	**48.0**	**44.3**	50.0
↪ stdev. (±)	0.15 / 0.16 / 0.19			0.16 / 0.15 / 0.14			0.17 / 0.20 / 0.21			0.18 / 0.16 / 0.14			0.02 / 0.04 / 0.05		
RoToR	52.7	45.9	53.5	**64.5**	**60.0**	**66.1**	**32.5**	**29.6**	**36.1**	**34.2**	**30.1**	**38.0**	**48.3**	**44.3**	**50.3**
↪ stdev. (±)	0.05 / 0.09 / 0.04			0.02 / 0.02 / 0.01			0.11 / 0.06 / 0.09			0.06 / 0.05 / 0.04			0.05 / 0.09 / 0.05		
RoToR-MonoT5	52.2	45.2	52.8		–		32.3	29.4	35.9	32.8	28.8	36.5		–	
↪ stdev. (±)	0.16 / 0.18 / 0.18				–		0.16 / 0.13 / 0.07			0.16 / 0.09 / 0.07				–	
RoToR-Freq.	**53.1**	**46.4**	**53.7**		–		32.4	29.3	**36.1**	33.7	29.6	37.4		–	
↪ stdev. (±)	0.02 / 0.07 / 0.03				–		0.09 / 0.04 / 0.06			0.04 / 0.16 / 0.22				–	

Table 2: **Mintaka (KGQA)** results, with **Initial** and **After-shuffle** settings, across different model parameter size and backbones. N refers to the number of top-k segments per query. Rows with "↪ stdev." report standard deviation over 3 seeds.

mance with improved efficiency and robustness. **Perplexity:** Lower generation perplexity indicates input representations are closer to in-distribution. On the same LitM dataset, RoToR's reduced perplexity implies its positional ID assignment effectively mitigates out-of-distribution effects. **Collision Rate:** PINE's similarity-based ordering often collides: on average, only 17.3 of 30 similarity values are unique, causing 42% of the segments to be indistinguishable and thus breaking invariance. In contrast, RoToR with lexical sorting only collides if the segment texts are identical. On LitM, this yields zero collisions, preserving full invariance.

Selective Routing MMLU (Tab. 3) is a representative of a task that involves not only order-invariant, but also order-sensitive (e.g., "None of the above"), inputs. Therefore, single use of order-invariant models does not always outperform the original model, limiting applicability of order-invariant models to practical listwise tasks, i.e., we observe significant performance drops for Set-based Prompting in MMLU, falling short of half the performance of the original model on initial ordering. However, using RoToR with Selective Routing to handle order-sensitive inputs outperforms, or is at least competitive as the original model in all possible orderings of candidate choices. Selective Routing improves the generalizability and extends the applicability on practical listwise tasks by adaptively handling order-sensitive inputs. The RoToR + Selective Routing (Oracle) performance on Tab. 3 was evaluated using a relaxed accuracy metric based on the union of predictions from the original and the invariant (RoToR-lexical) model. This improves significantly, which highlights the potential of Selective Routing for further accuracy

	Llama-3.1-8B-Instruct			Qwen1.5-4B-Chat			Qwen1.5-7B-Chat		
Method	**Init.**	**Rev.**	**Avg.**	**Init.**	**Rev.**	**Avg.**	**Init.**	**Rev.**	**Avg.**
Orig.	68.3	64.8	65.5 ± 1.0	53.6	51.9	52.6 ± 0.6	**60.1**	56.6	58.6 ± 0.9
PCW	57.0	55.1	56.1 ± 1.1		–			–	
Set-Based Prompting	31.1	33.0	31.6 ± 0.8		–			–	
PINE	64.8	63.3	63.6 ± 0.7	50.5	49.3	49.4 ± 0.5	57.0	54.4	55.8 ± 0.9
RoToR	63.2	62.6	62.8 ± 0.7	49.6	47.7	48.3 ± 0.7	56.5	55.8	56.2 ± 0.6
↪ + S.R.	**68.5**	65.1	65.7 ± 0.9	53.7	51.8	$\mathbf{52.6 \pm 0.6}$	**60.1**	**57.4**	$\mathbf{58.8 \pm 0.7}$
RoToR - MonoT5	64.2	62.9	63.5 ± 0.5	49.7	47.6	48.7 ± 0.7	56.2	54.4	55.5 ± 0.7
↪ + S.R.	68.4	65.2	65.8 ± 0.9	**53.8**	51.9	$\mathbf{52.6 \pm 0.6}$	**60.1**	57.3	58.7 ± 0.8
RoToR - Freq.	64.3	63.6	63.8 ± 0.6	49.9	47.6	48.7 ± 0.5	56.4	54.7	55.7 ± 0.7
↪ + S.R.	**68.5**	**65.3**	$\mathbf{65.8 \pm 0.8}$	53.7	**52.3**	$\mathbf{52.6 \pm 0.6}$	60.0	57.3	58.6 ± 0.8
RoToR + S.R. (Oracle)	75.0	71.9	72.7 ± 1.0	61.8	60.1	61.1 ± 1.0	68.1	66.2	67.2 ± 0.7

Table 3: Improving applicability to general listwise tasks (**MMLU**, N=4) with **Selective Routing** (S.R), which includes **both** order-invariant **and** order-sensitive examples. Init./Rev. refer to original/reversed orderings, Avg. is the average selection ratio across all (4!-1) re-orderings with standard deviation. S.R (Oracle) represents the upper bound with perfect routing accuracy. RoToR with Selective Routing shows improved performance and stability across input re-orderings.

Model	Benchmark	PINE	RoToR	Reduction
(a) Overhead FLOPs, relative to original model				
Llama-3.1-8B-Instruct	MMLU, $N = 4$	0.59×	**0.55**×	7.6%
	LitM, $N = 10$	7.07×	**4.81**×	31.9%
	LitM, $N = 30$	22.43×	**15.05**×	32.9%
Llama-3.1-70B-Instruct	KGQA, $N = 30$	1.27×	**0.94**×	26.0%
	KGQA, $N = 50$	1.82×	**1.29**×	29.0%
Qwen1.5-72B-Chat	KGQA, $N = 30$	0.45×	**0.01**×	98.0%
	KGQA, $N = 50$	0.58×	**0.03**×	94.8%
(b) End-to-end latency (s)				
Llama-3.1-70B-Instruct	LitM, $N = 10$	57,352	**44,219**	22.9%
	LitM, $N = 20$	87,091	**58,680**	32.6%
Llama-3.1-8B-Instruct	MMLU, $N = 4$	7,371	**6,608**	10.4%
	LitM, $N = 10$	18,551	**14,264**	23.1%
	LitM, $N = 30$	41,664	**23,569**	43.4%
(c) Perplexity & Collision rate, (on LitM)				
Llama-3.1-8B-Instruct	Perplexity ($N = 20$)	6.91	**6.65**	–
	Collision rate ($N = 30$)	42.3%	**0 (None)**	–

Table 4: **Unified efficiency comparison of RoToR vs. PINE,** reporting (a) Additional FLOPs, (b) Latency, and (c) Perplexity & Collision rate. Columns list each metric for PINE and RoToR, and the relative reduction. Yellow rows separate sub-sections.

gains through optimizing design choices on routing methods, which we plan to explore in future work. Additional analysis on the selection ratio of Selective Routing is reported at Appendix Section L.

Impact of global ordering algorithm While most of our experiments focus on the simplest lexical sorting method, RoToR supports any global sorting approach. To demonstrate this flexibility, we report experiments with various global sorting strategies, including reversed lexical sorting, MonoT5-based reranking, and token frequency-based sorting. Lexical sort is presented as a baseline (lower bound) - a simple algorithm ensuring global sorting. Our experiments on Tab. 1 show that any type of global sorting, with the use of circular assignment is *superior than PINE*, which relies on pairwise attention arrangements.

Extension to LLMs, other scenarios Experiments on Llama-3.1-70B-Instruct and Qwen-1.5-72B-Chat for LitM and KGQA show consistent and significant improvements over both the original implementation and the PINE baseline, demonstrating RoToR's generalizability to larger-scale LLMs. We further evaluate robustness on longer-context inputs (LongBench-2WikiMultiHopQA (Bai et al., 2024) in Appendix Section J) and different task templates (KGQA template swap in Appendix Section K). Results indicate that our method retains benefits across longer input scenarios and diverse task templates, including those without explicit input format requirements.

6 Conclusion

Our work addresses order-invariance in listwise inputs by identifying core issues in distribution mismatch and adaptive handling of mixed inputs. Our proposed RoToR by modifying self-attention by global sorting and circular arrangement provides a stable zero-shot order-invariant solution that reduces the complexity of positional ID modification, while Selective Routing adaptively routes between invariant and sensitive LMs to handle real-world scenarios. Together, these methods demonstrate improved performance and reliability on LitM, KGQA, and MMLU benchmarks.

7 Limitations

Our method can utilize any kind of deterministic sorting algorithm, but we have only experimented with limited global sorting algorithms due to time and resource constraints. We plan to investigate potentially better sorting algorithms in the future. Also, current ordering-invariant models are limited to inputs given as prefix + parallel + suffix. It would be beneficial to support more complex structures, such as ability to process multiple order-invariant contexts interleaved with serial text.

8 Acknowledgments

We thank the Channel Corporation for providing GPU resources to run the experiments, and their AI team for providing helpful feedback. We thank the members of LDILab and Jinwoo Kim for their constructive comments and the anonymous reviewers for their valuable suggestions. We are also grateful to our former intern, Yezun Chung (KAIST CS), for assisting with experiments on early versions of the Selective Routing approach.

This work was partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No. RS-2021-II212068, Artificial Intelligence Innovation Hub) and Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) [NO.RS-2021-II211343, Artificial Intelligence Graduate School Program (Seoul National University)]

References

Meta AI. 2024. Build the future of ai with meta llama 3.

Norah Alzahrani, Hisham Abdullah Alyahya, Yazeed Alnumay, Sultan Alrashed, Shaykhah Alsubaie, Yusef Almushaykeh, Faisal Mirza, Nouf Alotaibi, Nora Altwairesh, Areeb Alowisheq, M Saiful Bari, and Haidar Khan. 2024. When benchmarks are targets: Revealing the sensitivity of large language model leaderboards. *Preprint*, arXiv:2402.01781.

Chenxin An, Shansan Gong, Ming Zhong, Xingjian Zhao, Mukai Li, Jun Zhang, Lingpeng Kong, and Xipeng Qiu. 2023. L-eval: Instituting standardized evaluation for long context language models. *Preprint*, arXiv:2307.11088.

Jinheon Baek, Soyeong Jeong, Minki Kang, Jong C. Park, and Sung Ju Hwang. 2023. Knowledge-augmented language model verification. *Preprint*, arXiv:2310.12836.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2024. Longbench: A bilingual, multitask benchmark for long context understanding. *Preprint*, arXiv:2308.14508.

Tianle Cai, Kaixuan Huang, Jason D. Lee, and Mengdi Wang. 2023. Scaling in-context demonstrations with structured attention. In *Workshop on Efficient Systems for Foundation Models @ ICML2023*.

Xinyun Chen, Ryan A Chi, Xuezhi Wang, and Denny Zhou. 2024. Premise order matters in reasoning with large language models. *arXiv preprint arXiv:2402.08939*.

Yuhan Chen, Ang Lv, Ting-En Lin, Changyu Chen, Yuchuan Wu, Fei Huang, Yongbin Li, and Rui Yan. 2023. Fortify the shortest stave in attention: Enhancing context awareness of large language models for effective tool use. *arXiv preprint arXiv:2312.04455*.

Anshuman Chhabra, Hadi Askari, and Prasant Mohapatra. 2024. Revisiting zero-shot abstractive summarization in the era of large language models from the perspective of position bias. *Preprint*, arXiv:2401.01989.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2024. A framework for few-shot language model evaluation.

Vipul Gupta, David Pantoja, Candace Ross, Adina Williams, and Megan Ung. 2024. Changing answer order can decrease mmlu accuracy. *Preprint*, arXiv:2406.19470.

Adi Haviv, Ori Ram, Ofir Press, Peter Izsak, and Omer Levy. 2022a. Transformer language models without positional encodings still learn positional information. In *Findings of the Association for Computational Linguistics: EMNLP 2022*.

Adi Haviv, Ori Ram, Ofir Press, Peter Izsak, and Omer Levy. 2022b. Transformer language models without positional encodings still learn positional information. *Preprint*, arXiv:2203.16634.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Preprint*, arXiv:2009.03300.

Cheng-Yu Hsieh, Yung-Sung Chuang, Chun-Liang Li, Zifeng Wang, Long T Le, Abhishek Kumar, James Glass, Alexander Ratner, Chen-Yu Lee, Ranjay Krishna, et al. 2024. Found in the middle: Calibrating positional attention bias improves long context utilization. *arXiv preprint arXiv:2406.16008*.

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–es.

He Junqing, Pan Kunhao, Dong Xiaoqun, Song Zhuoyang, Liu Yibo, Liang Yuxin, Wang Hao, Sun Qianguo, Zhang Songxin, Xie Zejian, et al. 2023. Never lost in the middle: Improving large language models via attention strengthening question answering. *arXiv preprint arXiv:2311.09198*.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R. Kosiorek, Seungjin Choi, and Yee Whye Teh. 2019. Set transformer: A framework for attention-based permutation-invariant neural networks. In *ICML*.

Youngwon Lee, Seung won Hwang, Daniel Campos, Filip Graliński, Zhewei Yao, and Yuxiong He. 2025a. Cord: Balancing consistency and rank distillation for robust retrieval-augmented generation. *NAACL*.

Youngwon Lee, Seung won Hwang, Daniel Campos, Filip Graliński, Zhewei Yao, and Yuxiong He. 2025b. Inference scaling for bridging retrieval and augmented generation. *NAACL*.

Jiaqi Li, Mengmeng Wang, Zilong Zheng, and Muhan Zhang. 2023. Loogle: Can long-context language models understand long contexts? *arXiv preprint arXiv:2311.04939*.

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.

Reid McIlroy-Young, Katrina Brown, Conlan Olson, Linjun Zhang, and Cynthia Dwork. 2024. Set-based prompting: Provably solving the language model order dependency problem. *Preprint*, arXiv:2406.06581.

Ryan L. Murphy, Balasubramaniam Srinivasan, Vinayak Rao, and Bruno Ribeiro. 2019a. Janossy pooling: Learning deep permutation-invariant functions for variable-size inputs. *Preprint*, arXiv:1811.01900.

Ryan L. Murphy, Balasubramaniam Srinivasan, Vinayak Rao, and Bruno Ribeiro. 2019b. Relational pooling for graph representations. *Preprint*, arXiv:1903.02541.

Rodrigo Nogueira, Zhiying Jiang, and Jimmy Lin. 2020. Document ranking with a pretrained sequence-to-sequence model. *Preprint*, arXiv:2003.06713.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. 2023. Yarn: Efficient context window extension of large language models. *arXiv preprint arXiv:2309.00071*.

Alexander Peysakhovich and Adam Lerer. 2023. Attention sorting combats recency bias in long context language models. *arXiv preprint arXiv:2310.01427*.

Pouya Pezeshkpour and Estevam Hruschka. 2023. Large language models sensitivity to the order of options in multiple-choice questions. *arXiv preprint arXiv:2308.11483*.

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.

Priyanka Sen, Alham Fikri Aji, and Amir Saffari. 2022. Mintaka: A complex, natural, and multilingual dataset for end-to-end question answering. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1604–1619, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. Mpnet: Masked and permuted pre-training for language understanding. *Preprint*, arXiv:2004.09297.

Raphael Tang, Xinyu Zhang, Xueguang Ma, Jimmy Lin, and Ferhan Ture. 2024. Found in the middle: Permutation self-consistency improves listwise ranking in large language models. *Preprint*, arXiv:2310.07712.

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

Sheng-Lun Wei, Cheng-Kuang Wu, Hen-Hsen Huang, and Hsin-Hsi Chen. 2024. Unveiling selection biases: Exploring order and token sensitivity in large language models. *Preprint*, arXiv:2406.03009.

Peng Xu, Wei Ping, Xianchao Wu, Lawrence McAfee, Chen Zhu, Zihan Liu, Sandeep Subramanian, Evelina Bakhturina, Mohammad Shoeybi, and Bryan Catanzaro. 2023. Retrieval meets long context large language models. *arXiv preprint arXiv:2310.03025*.

Jingfeng Yang, Aditya Gupta, Shyam Upadhyay, Luheng He, Rahul Goel, and Shachi Paul. 2022. Tableformer: Robust transformer modeling for table-text encoding. *Preprint*, arXiv:2203.00274.

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. *Preprint*, arXiv:2305.15262.

Howard Yen, Tianyu Gao, and Danqi Chen. 2024. Long-context language modeling with parallel context encoding. *Preprint*, arXiv:2402.16617.

Yijiong Yu, Huiqiang Jiang, Xufang Luo, Qianhui Wu, Chin-Yew Lin, Dongsheng Li, Yuqing Yang, Yongfeng Huang, and Lili Qiu. 2024. Mitigate position bias in large language models via scaling a single dimension. *arXiv preprint arXiv:2406.02536*.

Lei Zhang, Yunshui Li, Ziqiang Liu, Jiaxi yang, Junhao Liu, Longze Chen, Run Luo, and Min Yang. 2024a. Marathon: A race through the realm of long context with large language models. *Preprint*, arXiv:2312.09542.

Zhenyu Zhang, Runjin Chen, Shiwei Liu, Zhewei Yao, Olatunji Ruwase, Beidi Chen, Xiaoxia Wu, and Zhangyang Wang. 2024b. Found in the middle: How language models use long contexts better via plug-and-play positional encoding. *arXiv preprint arXiv:2403.04797*.

Yilun Zhao, Chen Zhao, Linyong Nan, Zhenting Qi, Wenlin Zhang, Xiangru Tang, Boyu Mi, and Dragomir Radev. 2023. Robut: A systematic study of table qa robustness against human-annotated adversarial perturbations. *Preprint*, arXiv:2306.14321.

Chujie Zheng, Hao Zhou, Fandong Meng, Jie Zhou, and Minlie Huang. 2024a. Large language models are not robust multiple choice selectors. In *The Twelfth International Conference on Learning Representations*.

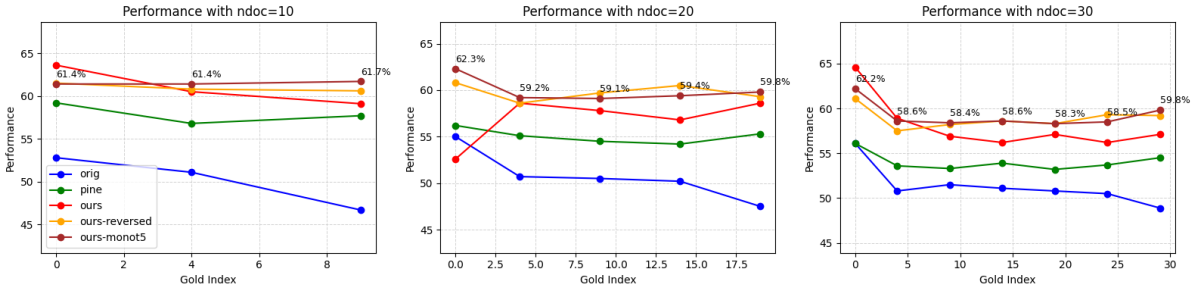
Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2024b. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36.

Wei Zhou, Mohsen Mesgar, Heike Adel, and Annemarie Friedrich. 2024. Freb-tqa: A fine-grained robustness evaluation benchmark for table question answering. *Preprint*, arXiv:2404.18585.

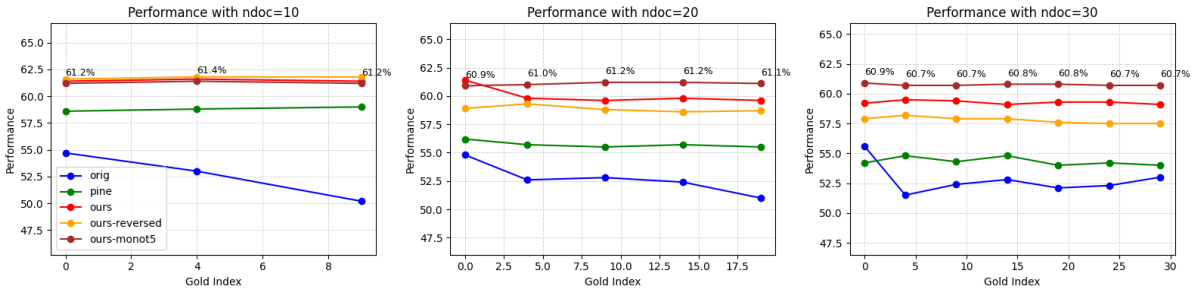
Lianghui Zhu, Xinggang Wang, and Xinlong Wang. 2023. Judgelm: Fine-tuned large language models are scalable judges. *arXiv preprint arXiv:2310.17631*.

Appendix

A Full results on the Lost in the Middle Benchmark



(a) Results **with index bias** (indexed by numbers). (Example input at Appendix Fig. 7)



(b) Results **without index bias** (indexed by title) (Example input at Appendix Fig. 8)

Figure 5: Results on the Lost-in-the-middle benchmark. Visualization of the `best_subspan_em` results at Appendix Tab. 1. RoToR (dark red, red, yellow) generally performs the best regardless of the position of the gold index, with less fluctuations when we remove index bias. Ours is RoToR with lexical sort, and ours-reversed is the one with the reversed lexical ordering. For brevity, only the performance of RoToR with reranking sort (MonoT5) is annotated as numbers, and the performance of PCW and Set-Based Prompting are reported only at the Table (Appendix Tab. 1) due to its low performance.

Impact of removing index bias on LitM Tab.5 presents the full results on the Lost in the Middle (LitM) benchmark, comparing scenarios where indexing bias is present versus removed. Fig.5 provides a visual representation of these results.

As shown in Appendix Tab.5, invariant LMs exhibit stable performance regardless of the gold index, especially when index bias is removed (as described in Sec.4.2; see also Appendix Fig.5b). However, when index bias is present, performance fluctuations are observed (Appendix Fig.5a). Notably, RoToR achieves the highest performance across all setups, demonstrating its effectiveness in mitigating positional bias in a zero-shot setting while maintaining overall performance.

These findings suggest that index bias acts as an implicit source of additional positional bias and that invariant LMs benefit significantly from its removal.

B Illustration of the global sorting method

We show the three different global sorting algorithms presented in our paper at Fig. 6.

C Details about preprocessing and evaluation of datasets

C.1 General.

All inferences were done with a **single** NVIDIA RTX A6000 48GB GPU. Note that all of the baseline models including our method can be applied directly in a zero-shot, training-free manner. For reproducibility, we fix the seed and disabled random sampling (i.e., used greedy decoding), set the maximum number of new generated tokens to 500, and set the pad_token_id to the same value as the eos_token_id

Total ndoc (segments)	10				20						30							
Gold idx at:	0	4	9	avg.	0	4	9	14	19	avg.	0	4	9	14	19	24	29	avg.
Indexing bias present																		
Original	52.8	51.1	46.7	50.2	55.0	50.7	50.5	50.2	47.5	50.7	56.1	50.8	51.5	51.1	50.8	50.5	48.9	51.4
PCW	12.0	11.9	12.1	12.0	3.4	3.7	3.8	3.9	3.6	5.1	2.1	2.1	2.0	1.9	2.0	2.2	2.0	2.0
Set-Based Prompting	40.8	40.7	40.8	40.8	25.6	25.8	25.7	25.5	25.3	25.6	15.8	15.9	16.1	16.0	16.1	15.7	15.8	15.9
PINE	59.2	56.8	57.7	57.9	56.2	55.1	54.5	54.2	55.3	55.5	56.1	53.6	53.3	53.9	53.2	53.7	54.5	54.0
RoToR-lexical	**63.6**	60.5	59.1	61.1	52.6	58.6	57.8	56.8	58.6	57.6	**64.6**	**58.9**	56.9	56.2	57.1	56.2	57.1	58.1
RoToR-reversed lexical	61.5	60.8	60.6	61.0	60.8	58.6	**59.7**	**60.5**	59.3	60.0	61.1	57.5	58.2	**58.6**	**58.3**	**59.3**	**59.2**	58.9
RoToR-reranking	61.4	**61.4**	**61.7**	**61.5**	**62.3**	**59.2**	59.1	59.4	**59.8**	**60.2**	62.2	58.6	**58.4**	**58.6**	**58.3**	58.5	**59.8**	**59.2**
RoToR-freq	62.8	61.1	59.5	61.1	62.9	58.8	56.7	57.4	58.0	59.1	61.7	58.2	56.9	56.1	56.4	55.4	56.8	57.4
Indexing bias removed (main paper)																		
Original	54.7	53.0	50.2	52.6	54.8	52.6	52.8	52.4	51.0	52.7	55.6	51.5	52.4	52.8	52.1	52.3	53.0	52.8
PCW	12.4	11.9	12.2	12.2	3.7	4.0	4.0	4.0	3.9	3.9	2.3	1.8	2.0	2.0	2.1	2.0	2.0	2.0
Set-Based Prompting	42.5	42.5	42.5	42.5	26.3	26.3	26.3	26.3	26.3	26.3	14.1	14.1	14.1	14.1	14.1	14.1	14.1	14.1
PINE	58.6	58.8	59.0	58.8	56.2	55.7	55.5	55.7	55.5	55.7	54.2	54.8	54.3	53.7	54.8	54.2	54.0	54.3
RoToR-lexical	61.4	61.6	61.6	61.5	**61.4**	59.8	59.6	59.6	59.8	60.0	59.2	59.5	59.4	59.1	59.0	59.3	59.1	59.2
RoToR-reversed lexical	**61.6**	**61.8**	**61.8**	**61.8**	58.9	59.3	58.8	58.6	58.7	58.8	57.9	58.2	57.9	57.4	57.9	57.6	57.5	57.8
RoToR-reranking	61.2	61.4	61.2	61.3	60.9	**61.0**	**61.2**	**61.2**	**61.2**	**61.1**	**60.9**	**60.7**	**60.7**	**60.7**	**60.8**	**60.8**	**60.7**	**60.8**
RoToR-freq	61.0	61.1	61.1	61.1	60.4	60.3	58.6	60.2	60.0	59.9	59.3	60.4	59.7	59.5	59.5	59.6	59.2	59.6

Table 5: The `best_subspan_em` (%) scores on the lost in the middle (LitM) benchmark. For RoToR, we test three different global ordering strategies (lexical, reversed lexical, and MonoT5-base reranking) across varying numbers of documents (ndoc $\in \{10, 20, 30\}$). Appendix Fig. 5 visualizes the fluctuations across different gold positions. RoToR shows the best performance across all setups, and is especially more stable when indexing bias in the input is removed.

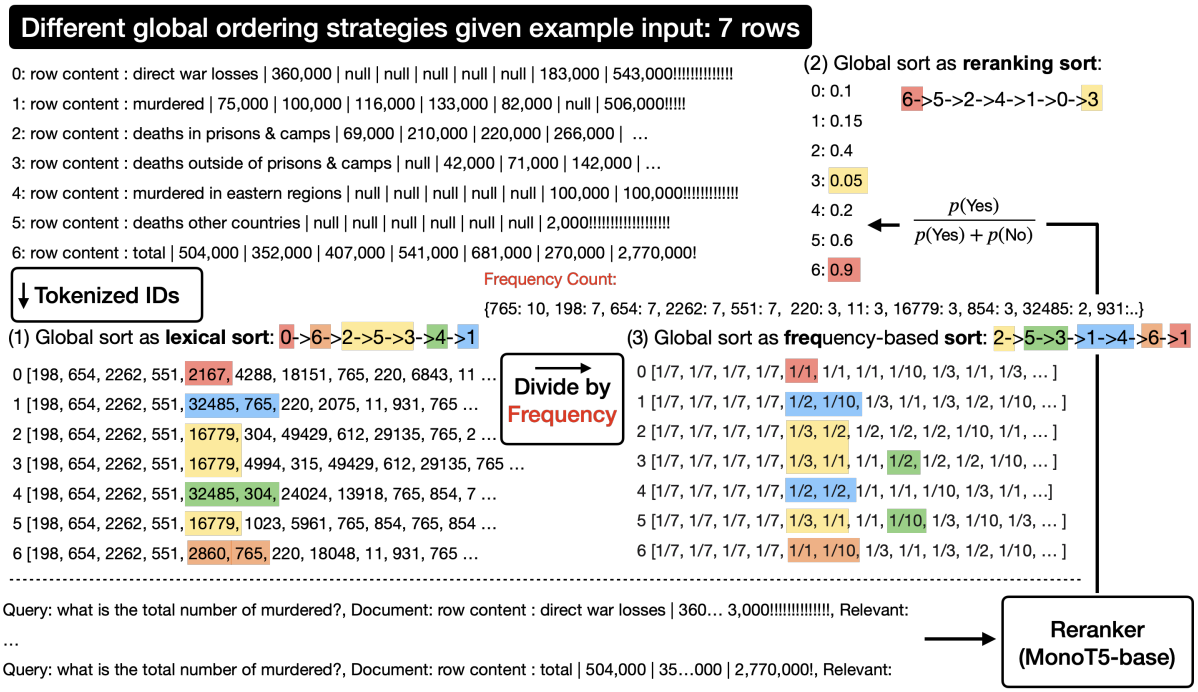


Figure 6: Illustration of ordering 7 rows by 2 different global sort options, (1) lexical sort based on token ids, or (2) reranking sort based on a reranker model (MonoT5-base in this case). (3) frequency based sorting applies lexical sorting, but the definitive token ids are mapped into the inverse frequency.

for all experiments. For all datasets we tested, we separate the input to 3 parts: prefix, parallel contexts, and suffix and feed them accordingly to the positional-invariant model. For the case of the **original** model, we simply join the prefix, context, and suffix text to make one sequential text. For testing with the **PCW** model, we concat each prefix to the parallel context due to the architectural limitations of the PCW model, which doesn't have the prefix part (which is processed casually before parallel contexts). For example, on testing the PCW model on MMLU, we append the question and each answer options, which generally result in longer input sequences. For PCW, we use the `pcw_generate` function, and we additionally

α =	no Selective Routing	-0.5	-0.4	-0.3	-0.2	-0.1	0	0.1	**0.2**	0.3	0.4	0.5
Validation set (1531)		<— bias towards invariant model —— bias towards orig model —>										
Selective Routing (orig, pine)	64.9	65.3	65.8	66.4	67.3	67.4	67.6	67.5	**67.9**	67.8	68.0	67.9
Selective Routing (orig, ours-lexical)	63.9	64.1	64.5	65.4	65.8	67.0	67.4	68.1	**68.2**	68.1	67.9	67.9
Selective Routing (orig, ours-monot5)	66.0	66.1	66.3	66.7	67.0	67.6	68.0	68.1	**68.1**	68.1	67.7	67.9
Test set (14015)												
Selective Routing (orig, pine)	64.8	64.9	65.4	66.0	66.8	67.5	68.4	68.5	**68.5**	68.4	68.3	68.3
Selective Routing (orig, ours-lexical)	63.2	63.5	64.2	65.2	66.2	67.3	68.0	68.4	**68.5**	68.5	68.4	68.3
Selective Routing (orig, ours-monot5)	64.2	64.4	64.8	65.5	66.4	67.3	68.1	68.5	**68.4**	68.5	68.3	68.3

Table 6: Reporting full ablation results on application of Selective Routing. α = 0.2 was the best for the validation set, which was then applied to obtain the reported results for all models.

utilized the `RestrictiveTokensLogitsProcessor` provided at the official PCW repository for MMLU classification, to have a similar setup with the log_likelihood option used for other models.

C.2 Knowledge Graph Question Answering

For evaluation with Mintaka, we follow the same setup as Baek et al. (2023). Given the gold answer and model generated answer, the EM score counts if both are exactly the same; Accuracy measures if the generated answer includes the gold answer, and F1 score measures the precision and recall among overlapping words. Since we are testing on a non-trained zero-shot version of the model, we enforce the model to output in json format to make it easier to parse. For the row shuffling setup, we fix the seed to 0, 1, 2 on shuffling rows and report the average scores.

C.3 Lost in the Middle

Specifically, we use the dataset provided in the official repository[9], use the same prompt as the llama 2 chat model with only the instruction tokens adjusted to llama 3 (removed `[Inst]` and changed to `<|begin_of_text|>` and etc.,), and evaluate using the `best_supspan_em` metric.

C.4 MMLU

We follow the publicly acknowledged lm-evaluation harness (Gao et al., 2024) prompt design by eluther.ai. We measure accuracy between the gold answer and the token with the highest likelihood (probability) among possible answer tokens [‘ A’, ‘ B’, ‘ C’, ‘ D’].

D Further impact scenarios on general conversation.

We shortly discuss about how this method may be applied to general conversational scenarios of LLMs. For processing contexts such as chronological history of conversations, the ordering is important, and the original LLM remains the better choice for this case. However, in subsets of conversational tasks requiring order invariance (e.g., Sets, Tables, or RAG contexts), our method enhances unbiased understanding, as demonstrated mainly in Lost-in-the-Middle benchmark. Here, RoToR achieves a significant 7-9% average accuracy gain over the original LLM, very consistently across all setups (doc indexing and ndoc) for all choices of the ordering algorithm, with lower standard deviation than the original model.

E Selection of α for Selective Routing on MMLU

We report α is a hyperparameter that can be tuned per-dataset. We searched its value in the range of -0.5 to 0.5 with a step size of 0.1 using the validation split of MMLU[10] on RoToR with lexical sorting, and applied the found value (0.2) on the test set to obtain the reported results for all models. We report the full variation of Selective Routing results on the investigated α value at Tab. 6.

[9] github.com/nelson-liu/lost-in-the-middle

[10] https://huggingface.co/datasets/cais/mmlu/viewer/abstract_algebra/validation

F Replication details on the runtime experiment

Apart from the theoretical runtime efficiency, we measured the actual end-to-end runtime in seconds, to better analyze the practical runtime efficiency between PINE and RoToR. The runtime of each experiment was measured on an ASUS ESC8000-E11 server featuring dual 4th Gen Intel Xeon Scalable processors, 64 CPU threads, 1.1 TB of RAM across 32 DIMM slots, and 8 NVIDIA A6000 GPUs with 48 GB of memory each. We Except for the experiments on Llama-3.1-70B-Instruct, we only use a single A6000 GPU for all of the experiments.

G Input data examples

To illustrate the input and output formats used in our experiments, we provide example inputs for the Lost-in-the-Middle (LitM), Knowledge Graph Question Answering (KGQA), and MMLU datasets. For experiments using the Qwen-Chat model, special tokens were adjusted accordingly. While the example prompts are based on the Llama-3.1-8B-Instruct model, the specific differences in token usage for the Qwen-Chat variants can be observed by comparing the prompts in Fig. 8 and Fig. 9. This adjustment is consistently applied across all datasets. Note that no special tokens are added for the MMLU benchmark, which aligns with the `lm-evaluation harness` setup.

lost in the middle

Prefix:
<|begin_of_text|><|start_header_id|>system<|end_header_id|>

You are a helpful, respectful and honest assistant. Always answer as helpfully as possible, while being safe. Please ensure that your responses are socially unbiased and positive in nature. If a question does not make any sense, or is not factually coherent, explain why instead of answering something not correct. If you don't know the answer to a question, please don't share false information.<|eot_id|><|start_header_id|>user<|end_header_id|>

Write a high-quality answer for the given question using only the provided search results (some of which might be irrelevant).

Parallel texts:
Document [1](Title: List of Nobel laureates in Physics) The first ...
...
Document [10](Title: Nobel Prize in Chemistry) on December 10, the ...

Suffix:
Question: who got the first nobel prize in physics<|eot_id|><|start_header_id|>assistant
<|end_header_id|>

Figure 7: Example input for the lost in the middle dataset.

lost in the middle no indexing

Prefix:
<|begin_of_text|><|start_header_id|>system<|end_header_id|>

You are a helpful, respectful and honest assistant. Always answer as helpfully as possible, while being safe. Please ensure that your responses are socially unbiased and positive in nature. If a question does not make any sense, or is not factually coherent, explain why instead of answering something not correct. If you don't know the answer to a question, please don't share false information.<|eot_id|><|start_header_id|>user<|end_header_id|>

Write a high-quality answer for the given question using only the provided search results (some of which might be irrelevant).

Parallel texts:
[Document Title: List of Nobel laureates in Physics] The first ...
...
[Document Title: Nobel Prize in Chemistry] on December 10, the ...

Suffix:
Question: who got the first nobel prize in physics<|eot_id|><|start_header_id|>assistant
<|end_header_id|>

Figure 8: Example input for the lost in the middle dataset, without indexing by numbers. Prompt for the Llama-3.1-8B-Instruct model.

lost in the middle no indexing (Qwen variant)

Prefix:
<|im_start|>system
You are a helpful, respectful and honest assistant. Always answer as helpfully as possible, while being safe. Please ensure that your responses are socially unbiased and positive in nature. If a question does not make any sense, or is not factually coherent, explain why instead of answering something not correct. If you don't know the answer to a question, please don't share false information.<|im_end|><|im_start|>user

Write a high-quality answer for the given question using only the provided search results (some of which might be irrelevant).

Parallel texts:
[Document Title: Thorax] when deep breaths are attempted. Different people ...
...
[Document Title: Chest pain] present with chest pain, and carry a significantly higher ...

Suffix:
Question: for complaints of sudden chest pain patients should take a<|im_end|>
<|im_start|>assistant

Figure 9: Example input for the lost in the middle dataset, without indexing by numbers, prompt for the Qwen1.5-Chat model.

Mintaka

Prefix:
<|begin_of_text|><|start_header_id|>system<|end_header_id|>

Below are the facts in the form of the triple meaningful to answer the question. Answer the given question in a JSON format, such as "Answer": "xxx". Only output the JSON, do NOT say any word or explain.

<|eot_id|><|start_header_id|>user<|end_header_id|>

Parallel texts:
(Super Bowl XLII, winner, New York Giants)
(Super Bowl XLII, participating team, New York Giants)
(Super Bowl XLII, point in time, time: +2008-02-03)
(Super Bowl XLII, followed by, Super Bowl XLIII)
(Super Bowl XLII, location, State Farm Stadium)
...
(Super Bowl XLII, sport, American football)
(Super Bowl XLII, instance of, Super Bowl)

Suffix:
Question: which team did the super bowl xlii mvp play for?, Answer: <|eot_id|><|start_header_id|> assistant <|end_header_id|>

Gold Answer(s):
('NYG', 'Giants', 'NY Giants', 'New York Giants')

Example generated output:
{"Answer": "New York Giants"} (Parsed to: New York Giants)

Figure 10: Example input for the Mintaka dataset.

MMLU

Prefix:
The following are multiple choice questions (with answers) about moral disputes.

Norcross agrees that if a being is incapable of moral reasoning, at even the most basic level, then it cannot be

Parallel texts:
A. a being of value.
B. an object of moral sympathy.
C. a moral agent.
D. a moral patient.

Suffix:
Answer:

Figure 11: Example input for the MMLU benchmark.

H Why is removing index bias an important detail for invariant models to be effective?

The alphabetic index (A/B/C/D) introduced in Fig. 1 associated with each segment, reportedly introduces token bias (Wei et al., 2024) of preferring the choice marked as 'A.' The same thing can be applied to listwise inputs with simple numeric indexing (1/2/3/4), which was the case for the lost-in-the middle benchmark. While a standard model with no modifications on positional encoding correctly places contexts indexed A before contexts indexed with D by positional encoding, an invariant model sees contexts in an order-agnostic way, meaning that the alphabetical indexing may not always be interpreted sequentially and thus can confuse the model from accurately interpreting the contexts. For example, even for cases where the index ordering of the input was in alphabetical order (A->B->C->D), the ordering-invariant model may interpret contexts with (C->A->B->D) at one point (e.g., when the query is D on self attention), which can cause unnatural, out-of-distribution representation, leading to decreased performance.

I Statisticial significance before and after shuffling segments

We conducted **paired two-tailed *t*-tests** (Table 8) for both the baseline ("original") model and our proposed method (RoToR), using the results in Table 7. Our goal was to determine whether the performance differences between the initial ordering and shuffled ordering are statistically significant. We excluded the Lost-in-the-Middle (LitM) dataset because it does not provide an initial ordering. Specifically, the tests evaluate whether the mean performance difference (Before Shuffle - After Shuffle) significantly deviates from zero.

For KGQA, we selected the F1 score as the representative metric among the three available, gathering data points from various task configurations and different models. For MMLU, the results are based on our RoToR variant with selective routing. As shown in Table 8, the original model shows a statistically significant drop in performance when the segments are shuffled, while RoToR does not, indicating increased robustness to segment-order perturbations. [11]

	Original Model			RoToR-lexical		
	Before Shuff.	After Shuff.	Diff.	Before Shuff.	After Shuff.	Diff.
Mintaka, Llama3.1-8B-Instruct, ndoc=30	51.9	51.0	0.9	54.1	53.8	0.3
Mintaka, Llama3.1-8B-Instruct, ndoc=50	51.7	51.0	0.7	53.6	53.5	0.1
Mintaka, Qwen1.5-4B-Chat, ndoc=30	34.9	34.7	0.2	35.7	35.5	0.2
Mintaka, Qwen1.5-4B-Chat, ndoc=50	35.8	35.0	0.8	36.2	36.1	0.1
Mintaka, Qwen1.5-7B-Chat, ndoc=30	35.4	35.0	0.4	37.7	37.7	0
Mintaka, Qwen1.5-7B-Chat, ndoc=50	35.7	35.5	0.2	38.0	38.0	0
MMLU, Llama3.1-8B-Instruct	68.3	65.5	2.8	68.5	65.7	2.8
MMLU, Qwen1.5-4B-Chat	53.6	52.6	1	53.7	52.6	1.1
MMLU, Qwen1.5-7B-Chat	60.1	58.6	1.5	60.1	58.8	1.3

Table 7: Performance of the **Original model** and **RoToR** before and after shuffling.

Derivation for the original model. Let the nine paired differences (Before − After) be $\{d_1, d_2, d_3, ...d_8, d_9\}$. **Mean Difference:** $\bar{d} = \frac{1}{9}\sum_{i=1}^{9} d_i$. In this case, $\bar{d} \approx 0.9444\%$. **Sample Standard Deviation:** $s_d = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(d_i - \bar{d})^2} \approx 0.7632$. **Standard Error (SE):** $\text{SE} = \frac{s_d}{\sqrt{n}} \approx 0.2544$. ***t*-Statistic:** $t = \frac{\bar{d}}{\text{SE}} \approx 3.7124,\ (df = 8)$. Since the critical value at $df = 8$ and $\alpha = 0.05$ is 2.306, we have $3.71 > 2.306$. Therefore, the difference is statistically significant.

[11] All statistical calculations were validated using an online t-test calculator: https://www.mathportal.org/calculators/statistics-calculator/t-test-calculator.php

	Original	RoToR
Degrees of Freedom	8	
Mean Difference	0.94	0.66
t-Statistic	3.71	2.23
Critical Value	2.306	
Statistically Significant	**Yes**	**No**

Table 8: Paired two-tailed t-test results comparing the original model and ours.

Derivation for the RoToR model. Under the same procedure, $\bar{d} \approx 0.6556\%$, $s_d \approx 0.8833$, SE ≈ 0.2944. ***t*-Statistic:** $t \approx 2.2265$. Since $2.2265 < 2.306$, there is no significant difference in performance before and after shuffling for RoToR.

Conclusion. While the original model shows a statistically significant performance drop with shuffled inputs, RoToR remains unaffected, demonstrating greater robustness to segment-order perturbations.

J Application to Long-Context Inputs

Benchmark and protocol. To test whether RoToR consistantly performs well to inputs with longer contexts, we extend our evaluation to the **LongBench** (Bai et al., 2024)–**2WikiMultihopQA** task, whose multi-document questions yield input lengths from 5 k to 15 k tokens. Because our current RoToR and PINE implementations do *not* yet support advanced parallelization (e.g., FlashAttention or SDPA), GPU memory becomes prohibitive beyond ∼10 k tokens, especially for PINE, whose memory footprint is larger than ours. We therefore truncate the context at 10 k tokens, which already exceeds the lengths used in our main experiments.

Following the official LongBench guidelines[12], we segment each long context into 512-token "listwise" inputs, enabling a direct comparison with our listwise reranking pipeline. To ensure robustness, we test three input-order perturbations:

1. **Initial**: original chunk order $[1, 2, 3, 4, 5]$;
2. **Reversed**: $[5, 4, 3, 2, 1]$;
3. **Center-flipped**: first and last halves swapped, [3,2,1,5,4].

Results. Table 9 reports F1 scores for the Original model variant, RoToR, and PINE with **Llama 3.1-8B-Instruct** and **Qwen 1.5-7B-Chat**. Across every ordering and token-length band, RoToR retains a clear advantage, while PINE fails to run when the longest (8 k+) chunks are kept. These findings confirm that RoToR scales to substantially longer inputs across different input-order perturbations.

Take-aways. Even without specialised long-context kernels, RoToR consistently outperforms both the original reranker and PINE, and remains robust to severe input-order perturbations. This suggests our approach can generalize to substantially longer inputs once memory and kernel constraints are alleviated.

K Robustness to task templates

We evaluated whether the proposed selective-routing methods remain effective when the surrounding task instructions are re-phrased. Concretely, we performed a *template-swap* experiment on the KGQA benchmark, specifically on the initial ordering setup. The original prompt (Figure 10) began with

> "Below are the facts in the form of triples meaningful to answer the question."

[12]Maximum generation length 32 and `qa_f1_score` (LongBench-E) as the evaluation metric.

Order	Method	Llama 3.1-8B-Instruct				Qwen 1.5-7B-Chat			
		0–4k	4–8k	8k+	Total	0–4k	4–8k	8k+	Total
	Count	25	131	144	300	23	121	156	300
Initial (e.g., 1,2,3,4,5)	Orig.	48.3	56.8	34.0	45.1	65.6	47.9	26.7	38.2
	PINE	51.0	47.6	–	–	70.2	45.1	–	–
	RoToR	**59.0**	52.7	**41.8**	**48.0**	**75.7**	**47.8**	**31.0**	**41.2**
Reversed (e.g., 5,4,3,2,1)	Orig.	57.0	51.5	39.0	46.0	53.4	43.3	**34.2**	39.3
	PINE	43.0	49.8	–	–	64.1	**48.9**	–	–
	RoToR	**59.0**	**52.0**	**41.0**	**47.3**	**72.8**	47.6	30.8	**40.8**
Center flip (e.g., 3,2,1,5,4)	Orig.	47.0	47.7	35.6	41.8	61.0	40.6	32.7	38.1
	PINE	46.3	49.2	–	–	70.2	43.5	–	–
	RoToR	**59.0**	**52.5**	**41.5**	**47.8**	**77.1**	**47.3**	**30.9**	**41.0**

Table 9: F1 scores (%) on LONGBENCH–2WikiMultihopQA with ∼10k-token contexts. "Count" is the number of examples per length bucket; "–" denotes out-of-memory.

	Llama-3.1-8B-Instruct						Qwen-1.5-4B-Chat						Qwen-1.5-7B-Chat					
	$N = 30$			$N = 50$			$N = 30$			$N = 50$			$N = 30$			$N = 50$		
Method	Acc.	EM	F1	Acc.	EM	F1	Acc.	EM	F1	Acc.	EM	F1	Acc.	EM	F1	Acc.	EM	F1
Original template, Initial ordering																		
Original	50.2	44.0	51.9	50.0	44.0	51.7	30.7	27.9	34.9	31.6	28.6	35.8	31.5	27.8	35.4	31.7	28.0	35.7
PINE	51.5	45.0	52.6	51.6	45.1	52.6	31.6	28.7	35.6	31.6	28.8	35.3	32.3	28.8	36.4	32.0	28.5	35.9
RoToR	**53.1**	**46.5**	**54.1**	52.9	46.0	53.6	32.0	29.0	35.7	**32.7**	**29.6**	**36.2**	**34.3**	**29.8**	**37.7**	**34.3**	**30.1**	**38.0**
RoToR-MonoT5	51.6	45.0	52.5	52.4	45.4	52.8	**32.3**	29.1	**36.2**	32.3	29.3	35.9	32.9	28.4	36.3	32.9	28.9	36.6
RoToR-Freq.	52.6	46.1	53.7	**53.1**	**46.4**	**53.7**	**32.3**	**29.2**	36.0	32.3	29.2	35.9	33.7	29.5	37.2	33.5	29.5	37.2
Template-swap																		
Original	50.0	44.1	51.8	50.2	44.3	51.9	31.1	27.8	34.9	31.7	28.3	35.3	31.4	27.6	35.2	32.0	28.0	35.7
PINE	52.0	45.7	52.9	52.0	45.3	52.8	31.9	28.8	35.8	31.7	28.6	35.4	31.9	28.5	36.0	31.5	28.2	35.7
RoToR	**52.7**	**46.4**	**54.0**	**52.9**	46.4	53.7	31.8	28.2	35.0	32.4	29.0	35.6	**34.1**	29.8	**37.6**	**34.0**	**29.9**	**37.7**
RoToR-MonoT5	51.5	45.2	52.6	52.5	45.7	53.1	**32.4**	**29.0**	**36.3**	**32.6**	**29.3**	**35.9**	32.9	28.5	36.4	32.6	28.5	36.3
RoToR-Freq.	52.3	46.2	53.7	**52.9**	**46.5**	**53.8**	31.9	28.4	35.4	32.4	28.8	35.6	34.0	**29.9**	37.4	33.8	29.6	37.4

Table 10: Results on the Mintaka (KGQA) dataset on different models, before (top block, also reported at main paper) and after (bottom block) the template-swap. N refers to number of top-k segments per query. RoToR variants consistently outperform the Original and PINE baselines, and their performance is stable under the swapped template, indicating robustness to instruction wording.

and required the model to output only a JSON object. In the swapped template we replaced the first sentence with

> "Below are knowledge statements expressed as triples meaningful to answer the question."

leaving all other instructions unchanged. Table 10 reports the results. Across all three backbone models and both retrieval depths (N=30, 50), RoToR and its variants retain similar absolute scores and continue to outperform both the Original and PINE baselines, indicating strong robustness to superficial wording changes in the task template.

L Additional Statistics on Selective Routing Assignment

Table 11 complements the main results in Table 3 by reporting the *selection ratio*, accounting for the percentage of evaluation queries for which the RoToR branch is chosen over the vanilla branch, under **Selective Routing** (SR).[13] We break the analysis down by (i) the global sorting strategy (Lexical, MonoT5, or Freq.), and (ii) three model backbones. The table distinguishes three order–based conditions:

For table 11, Init. refers to the original ordering (e.g., in abcd order), Rev. refers to the reversed ordering (e.g., in dcba but assigned as abcd), and Avg. is the average selection ratio for all possible (4!-1) re-orderings, with standard deviation. Empirically, the RoToR model tends to be selected more frequently

[13] All figures are computed over the full evaluation set of 14,015 queries.

	Llama-3.1-8B-Instr.			Qwen1.5-4B-Chat			Qwen1.5-7B-Chat		
Sorting	Init.	Rev.	Avg.	Init.	Rev.	Avg.	Init.	Rev.	Avg.
Lexical	7.0	8.5	7.3±0.8	5.9	6.2	6.2±0.4	10.3	10.6	9.9±0.6
MonoT5	6.9	7.6	6.7±1.5	8.0	12.5	9.8±2.1	10.7	10.9	10.7±0.7
Freq.	6.4	6.7	6.9±0.5	8.5	10.9	9.4±1.6	10.7	11.1	11.1±0.8

Table 11: Selection ratio (%) of the RoToR variant under SR. Higher values indicate more frequent routing to RoToR.

under reversed orderings, whereas under the original ordering, the vanilla model is chosen slightly more often. The exact ratio varies by model and sorting strategy.