Eliminating Position Bias of Language Models: A Mechanistic Approach

Ziqi Wang[1] **Hanlin Zhang**[2] **Xiner Li**[3] **Kuan-Hao Huang**[1] **Chi Han**[1]

Shuiwang Ji[3] **Sham M. Kakade**[2] **Hao Peng**[1] **Heng Ji**[1]

Abstract

Position bias has proven to be a prevalent issue of modern language models (LMs), where the models prioritize content based on its position within the given context. This bias often leads to unexpected model failures and hurts performance, robustness, and reliability across various applications. Our mechanistic analysis attributes the position bias to two components employed in nearly all state-of-the-art LMs: causal attention and relative positional encodings. Based on the above analyses, we propose to **eliminate** position bias caused by different input segment orders (e.g., options in LM-as-a-judge, retrieved documents in QA) in a **training-free zero-shot** manner. Our method changes the causal attention to bidirectional attention between segments and utilizes model attention values to decide the relative orders of segments instead of using the order provided in input prompts, therefore enabling **P**osition-**IN**variant inferenc**E** (**PINE**) at the segment level. Results on the LM-as-a-judge task show that PINE is especially useful when adapting LMs for evaluating reasoning pairs: it consistently provides 8 to 10 percentage points performance gains in most cases, and makes `Llama-3-70B-Instruct` perform even better than `GPT-4-0125-preview` and `GPT-4o-2024-08-06` on the RewardBench reasoning subset. This is a short version for workshop, full version can be found here: `https://arxiv.org/abs/2407.01100`. [1]

1 Introduction

Language models (LMs) [7, 11, 39, 1] demonstrate impressive performance in general language tasks such as dialogue [38], reasoning [11], and schema induction [21]. However, they tend to favor content at certain positions [52, 51, 42, 12, 53, 9, 23], which harms complex reasoning [9], long-context understanding [23] and model-based evaluation [52]. For example, LMs tend to favor the first when it is required to compare the quality of two candidate responses [52], which hurts their reliability when being used as evaluators (Figure 1 upper). Different from previous *ad-hoc* solutions that mitigate this problem [33, 8, 14, 16, 53], we seek to understand the causes of position bias and propose to eliminate position bias.

We start by analyzing the key components of state-of-the-art LMs – Casual Attention and Rotary Position Embedding (RoPE) [36] that enable models to understand the order of tokens so that LMs can generate meaningful outputs. We argue that they are also the only two operations in Transformers [40] that will inevitably bring undesirable position bias. This is because other operations do not change representations when position changes (Section 2). We also hypothesize RoPE has recency bias [36, 28] due to its long-form attention weight decay w.r.t. the increase of relative positions, and causal attention forces unidirectional information propagation, enabling models to pay more attention

[1]Code available at: `https://github.com/wzq016/PINE`. [1] University of Illinois Urbana-Champaign, [2] Harvard University, [3] Texas A&M University. Contact: `ziqiw9@illinois.edu`

Workshop on Scientific Methods for Understanding Deep Learning, NeurIPS 2024.

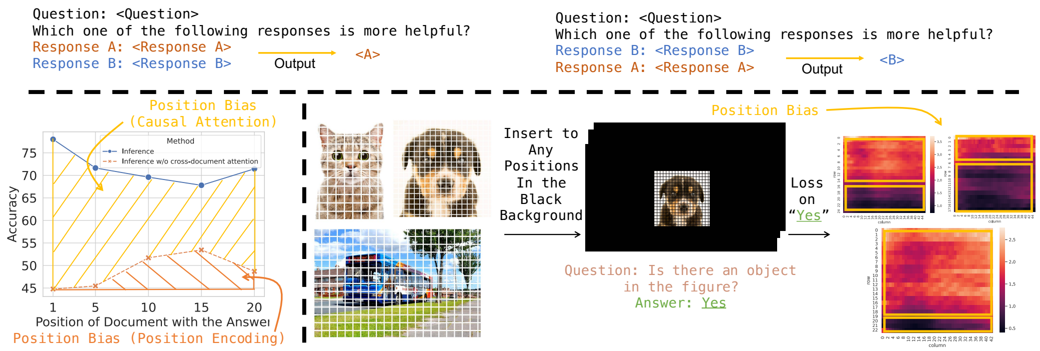

Figure 1: Motivating examples showing how position bias affects model outputs. **Upper**: LMs are prone to prefer the response positioned at first. **Lower Left**: LMs (`Llama-3-8B-Instruct`) are presented with 20 documents to answer a question, with only one document (the gold-standard document) containing the correct answer. The blue curve represents normal inference. The red curve represents inference that masks attention between documents. The height change of the yellow and orange area reflects the position bias brought by causal attention and RoPE: causal attention generally favors distant content, but RoPE prefers nearby content. **Lower Right**: We insert a real-world image to a large black background image at different positions and prompt VLMs (`Fuyu-8B` [6]) to compute the loss on the ground truth token. We observe a consistent pattern that models have lower losses (black color) when images are presented at the bottom. More examples of VLMc can be found in Appendix A.

to distant content (similar to think-dot-by-dot [29]). To verify the hypothesis, we conduct a simple analysis on the retrieval-augmented QA [23] (Figure 1 lower left). The height change of the yellow area and orange area reflects the position bias of causal attention and RoPE. Since the yellow area is mostly wider at the beginning and the orange area generally becomes wider at the end (except for the last data point), showing that the causal attention generally tends to favor distant content, while RoPE generally tends to favor nearby content.

As a solution, we propose PINE that can eliminate position bias by manipulating causal attention and RoPE to attend to different content equally. For tasks that contain position-agnostic segments, and segment orders are not expected to affect results (e.g., retrieved documents and candidate responses in retrieval-augmented QA [23] and LM-as-a-judge [52]), we make the inter-segment attention bidirectional so that attention mask will equally attend to all segments. Next, we compute importance scores between segments and use them to re-sort segment positions so that positions in the original inputs are discarded. The resulting approach enables **P**osition-**in**variant inferenc**e** (PINE) in a **training-free/zero-shot** manner that operates on pre-determined segments.

2 Methodology

We first introduce the formulation of position bias, then analyze the causes and illustrate our methods.

Formulation. We take retrieval-augmented QA as an example, where current LMs' performance may greatly suffer from position bias [23]. The task requires the model to answer a question based on a set of given retrieved documents, where only one of them contains the correct answer. A system prompt SYS describes the task definition: "`Answer the question based on the retrieved documents.`". Given a question Q, and three retrieved documents: $\mathcal{D}_1, \mathcal{D}_2$, and $\mathcal{D}_3$, we can formulate several different inputs. For example, $[\text{SYS}|\text{Q}|\mathcal{D}_1|\mathcal{D}_2|\mathcal{D}_3]$, and $[\text{SYS}|\text{Q}|\mathcal{D}_2|\mathcal{D}_3|\mathcal{D}_1]$ (See Appendix C for a prompt example). We expect models to have the same output for these inputs because $\mathcal{D}_2, \mathcal{D}_3, \mathcal{D}_1$ are **position-agnostic input segments**: their relative order is not supposed to affect the final result. However, the current LMs answer differently when presented with these different inputs and tend to answer correctly when the document contains the answer at the beginning or at the end of all documents [23]. The systematic differences of model outputs caused by relative positions of position-agnostic input segments reflect the **position bias** of the model. Therefore, current LMs cannot conduct **inter-segment position-invariant** inference, and our goal is to make the inference

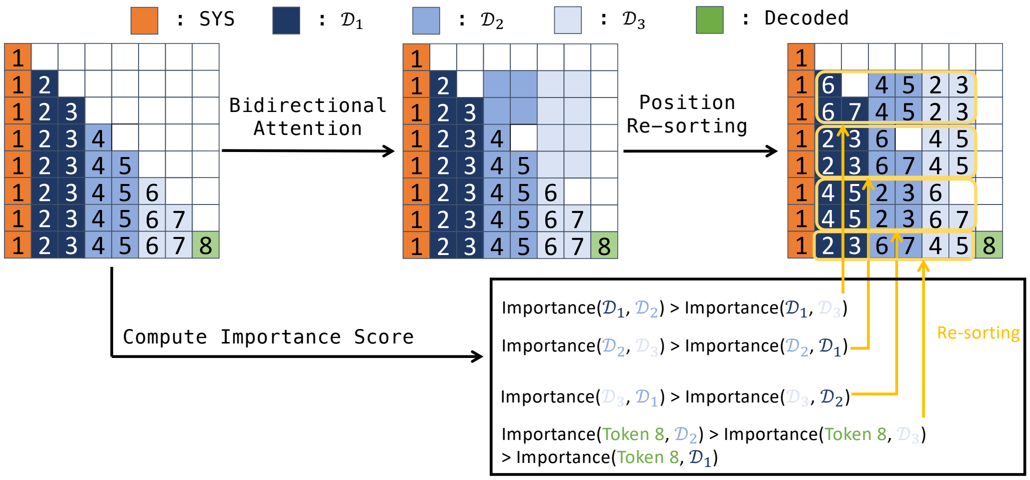

Figure 2: PINE: inter-segment position-invariant inference via bidirectional attention. The attention matrix of the running example in Section 2 is at the left of the figure, the orange, different blue, and green colors denote system prompts (1 token), three different documents (2 tokens each) and decoded tokens (1 token), respectively. The number at (i, j) in the figure, p_{ij}, denotes the position of a token j when computing the attention from query $\mathbf{q}_i$. Therefore, $p_{\cdot j}$ is equal for all i in vanilla inference. PINE enables inter-segment bidirectional attention and then uses attention scores between segments to compute their importance scores. Then, segments are re-sorted by importance scores: higher-importance-score segments are placed in closer positions. The computation of "importance score" is introduced in Section 2.

invariant w.r.t. relative segment orders. Since we will use this example in the rest of the paper, we use "document" and "segment" interchangeably.

Causal Attention and RoPE Are The Cause of Position Bias. Feed-Forward Networks (FFNs), Query, Key and Value (QKV) projections, and layer normalization in the Transformer architecture do not cause position bias, as they are invariant to relative segment positions. Rather, the attention computation that leads to the position bias:

$$\begin{aligned} \mathbf{Q}_{\text{PE}} &= \text{PE}(\mathbf{Q}, \mathbf{pos}_{\mathbf{Q}}), \mathbf{K}_{\text{PE}} = \text{PE}(\mathbf{K}, \mathbf{pos}_{\mathbf{K}}) \\ \mathbf{H} &= \text{Softmax}\left(\mathbf{Q}_{\text{PE}}\mathbf{K}_{\text{PE}}^T/\sqrt{d}\right) \odot \mathbb{1}_{\text{causal}}\mathbf{V} \end{aligned} \tag{1}$$

where $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{n \times d}$ are queries, keys, and values, PE donotes the position encoding (specifically RoPE), $\mathbf{pos}_{\mathbf{Q}}$ and $\mathbf{pos}_{\mathbf{K}}$ denote the position of queries and keys, and $\mathbb{1}_{\text{causal}}$ denotes the causal attention mask. Eq. 1 reveals that (1) the PE function yields different representations for input segments if their relative order changes, therefore affecting the importance score $Q_{\text{PE}}K_{\text{PE}}^T$ and hidden states; (2) the $\mathbb{1}_{\text{causal}}$ generates different attention masks for the input segments if we change their positions, resulting in different hidden states. To achieve inter-segment position-invariant inference, **H needs to remain the same regardless of segment orders**.

2.1 PINE: Inter-Segment Position-Invariant Inference via Bidirectional Attention.

Our goal is to obtain an inter-document position-invariant hidden state $\mathbf{H}_{\text{PINE}}$, which does not change regardless of document orders. We can mechanistically eliminate the position bias by equally attending to all documents. Therefore, we propose PINE, an approach that uses bidirectional inter-segment attention and re-assigning positions by importance scores (computed from attention score) to eliminate position bias (Figure 2).

Bidirectional Attention. We first change the attention mask so that documents can attend to each other. Specifically, we make the inter-document attention **bidirectional** but keep the intra-document attention **causal** (Figure 2, middle). Our goal is to eliminate "inter" position bias among different documents rather than "intra" position bias within each document. The latter will lose the order information of tokens, and models can degenerate into bag-of-words models, which is not what we expect.

Re-assign Positions: Sorting By Importance Scores. Re-assigning positions must consider two folds: the position of queries and keys. Each token in conventional LMs has the same position when serving as both query and key. In the bidirectional attention we use, this assignment has to be reconsidered. First, LMs are trained causally, meaning the position of the query must be larger than the keys in the attention computation. Therefore, it is necessary to manipulate positions so that each document is the **last** document when serving as queries (the diagonal of the rightmost figure in Figure 2). For tokens before and after documents, their positions are not affected when serving as queries.

Re-assigning positions for keys must be redesigned to eliminate position bias. We determine the positions of documents based on importance scores when they serve as keys (numbers in the rightmost part of Figure 2). Specifically, we first compute the attentions without position embedding involved: $\text{Importance}_{\text{token}}(i, j) = \text{Softmax}(\mathbf{q}_i \mathbf{k}_j^T / \sqrt{d})$, where d is the hidden state dimension. Then, we obtain the importance score between documents by aggregation. For example, $\text{Importance}(\mathcal{D}_1, \mathcal{D}_2) = \sum_{i \in \mathcal{D}_1, j \in \mathcal{D}_2} \text{Importance}_{\text{token}}(i, j)/|\mathcal{D}_2|$. The length normalization is to prevent assigning higher importance scores to longer documents. The importance score could also be computed between individual tokens (e.g., Token 8) and documents. Lastly, we re-assign positions by importance scores as shown in the rightmost part of Figure 2: more important documents will have closer positions to the query. The rightmost part of Figure 2 shows the concrete position re-assignment for keys (its diagonal also represents the position re-assignment for queries). To avoid confusion, we address the fact that we do not actually sort tokens and only re-assign them to different positions. In our position re-assignment, the position of keys may vary depending on the queries (numbers in column are different), which is the key difference between PINE and vanilla inference. Besides, our method is not limited to specific position embedding types.

Inter-Document Position Invariant Inference. Once we have new attention mask and position re-assignment, we can place them into Equation 1, and obtain $\mathbf{H}_{\text{PINE}}$. By applying $\mathbf{H}_{\text{PINE}}$ to every layer, attention heads, and tokens, we reach our method PINE. We prove that:

Lemma 1. *If the input $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ are inter-document position-invariant representations, then $\mathbf{H}_{PINE}$ are also inter-document position-invariant representations.*

Theorem 1. *Given an input, if $\mathbf{H}_{PINE}$ is applied to every layer, attention head, and token to replace the conventional attention computation, then the model outputs are inter-document position-invariant representations.*

The lemma can be proved by showing that (1) importance scores and position re-assignment are not a function of input document positions, and (2) bidirectional attention mask is not a function of document positions. The theorem can be proved by mathematical induction by (1) lemma, (2) FFN, QKV projection, and layer norm yield representations that are not a function of document positions, and (3) the embedding representation is not a function of document positions.

We put the complete proof in Appendix D.1. We also suggest you read the proof if you feel unclear about the workflow of PINE , as we will go through a concrete example in the proof. Some takeaways that are worth noting: (1) Both bidirectional attention mask and position re-assignment are needed to complete the proof. (2) PINE needs to be applied to every layer, attention heads, and tokens to complete the proof. (3) PINE is not limited to specific position embedding types. More discussions on possible variants can be found in Appendix D.2.

Inference Cost. PINE incurrs additional computation overhead due to extra operations. Practically, the extra big $\mathcal{O}$ computation complexity to obtain hidden states is $\mathcal{O}(nk \log k)$, where n and k denote text length and the number of input documents, respectively. The bidirectional attention does not bring extra cost, the position re-assignment brings $\mathcal{O}(k \log k)$ for each token since the sorting algorithms are involved. The real computation cost is acceptable since k is usually small (e.g., $k = 2$ in the LLM-as-a-judge task and $k = 20$ in the retrieval-augmented QA). Section E.5 shows results of real-world wall time and memory cost.

3 Experiments

We benchmark our method on the LM-as-a-judge task [52] that prompts LMs to select a better response out of two given questions. We select Rewardbench that contains 23 datasets [2] [19] as

[2]Apache-2.0 license. https://github.com/allenai/reward-bench

Table 1: Main results of RewardBench. Vanilla (GT at A) means vanilla inference with data that the ground truth chosen response is always presented at the first, and (GT at B) indicates the ground truth chosen response is always presented at the first. Therefore, Vanilla (GT at X) denotes extreme cases where chosen responses are always allocated at a fixed position, and Vanilla represents an average case where chosen responses may occur in both positions (randomly shuffled). Since LM-as-a-judge can be regarded as a binary classification problem, the random guess gives a 50% accuracy in expectation. PINE generates the same results for all three cases in experiments (i.e., GT at X and randomly shuffling. Therefore, we only report once in the table), which is consistent to Theorem 1. PINE consistently improves LM's performance across different model sizes compared with the Vanilla setting.

Method	Llama-3-Instruct 8B	Llama-3-Instruct 70B	Qwen-1.5-Chat 1.8B	Qwen-1.5-Chat 4B	Qwen-1.5-Chat 7B	Qwen-1.5-Chat 32B	Qwen-1.5-Chat 72B	Qwen-1.5-Chat 110B
RewardBench (Full set)								
Vanilla (GT at A)	67.5	78.0	36.3	29.5	61.4	74.2	79.6	87.2
Vanilla (GT at B)	66.3	76.5	66.2	76.6	59.6	74.8	69.5	75.7
Vanilla	64.8	76.0	50.3	53.1	60.9	72.8	72.8	81.1
PINE	**66.7**$_{+1.9}$	**77.4**$_{+1.4}$	**52.9**$_{+2.6}$	**58.2**$_{+5.1}$	**61.5**$_{+0.6}$	**74.8**$_{+2.0}$	**71.8**$_{-1.1}$	**82.9**$_{+1.7}$
RewardBench (Reasoning subset)								
Vanilla (GT at A)	80.3	87.8	43.3	42.8	62.1	78.3	83.0	90.0
Vanilla (GT at B)	66.0	80.3	57.2	62.3	54.3	73.6	68.7	73.0
Vanilla	65.3	78.9	48.4	54.1	59.3	66.8	68.2	78.0
PINE	**73.4**$_{+8.1}$	**87.6**$_{+8.7}$	**60.1**$_{+11.7}$	**61.0**$_{+6.9}$	**63.0**$_{+3.7}$	**76.7**$_{+9.9}$	**69.0**$_{+0.8}$	**86.2**$_{+8.2}$

our benchmark. RewardBench can be categorized into four types: Chat, Chat-Hard, Safety, and Reasoning. We use the official data split, prompts, and evaluation scripts to ensure reproducibility. We use LLaMa-3-Instruct models [3] and Qwen-1.5-Chat models [5] for experiments. To show how positions affect results, we present four results: the ground-truth response is positioned at first, second, or shuffled, and PINE results (which yield the same results for all three scenarios above). The inference temperature is set to 0 to follow previous works' settings. More details of the four tasks can be found in Appendix F. Qualitative examples of the four tasks can be found in Appendix G.

The main findings are as follows (Table 1):

- First, the first two methods (GT at X) reveal that larger models tend to have a primacy bias, whereas smaller models tend to have a recency bias.
- By comparing the last two rows of each model size, we conclude that models across different sizes perform better with the help of PINE by eliminating position bias.
- The only exception is the Qwen-1.5-72B-Chat model. We suspect this model is not well-trained since Qwen-1.5-32B-Chat performs extremely similarly to the 72B model in vanilla inference. Qwen 2 report [46] also shows that the Qwen 1.5 72B model performs even worse than 32B in reasoning. Moreover, our experiments on Qwen 2.5 72B shows PINE benefits too.
- PINE consistently improves model performance on the "reasoning" subset by a large margin: from 8 to 10 percentage points in most cases. Specifically, LlaMa-3 Instruct 70B was originally ranked 22nd generative model in the reasoning subset of RewardBench. With PINE, it achieves the 7th rank (87.6%), **outperforming** `GPT-4-0125-preview` **(the previous 8th rank,** 86.9%**),** `GPT-4o-2024-08-06` **(the previous 9th rank,** 86.6%**), and** `Llama-3.1-405B-Instruct-Turbo` **(the previous 7th rank,** 87.1%**).**[3].
- The wall-time cost of PINE is 2x of the vanilla inference, and no obvious memory gain. We do not optimize our code, and we still use a "for" loop in our implementation. Therefore, we believe there is room to improve the efficiency, and we leave this as a future work.

We put full experiment results, including baseline comparison, ablation study, and results on other tasks (retrieval-augmented QA, molecule generation, and math reasoning), into Appendix E for interested readers.

[3]Results are provided by the official leaderboard (as of Sep 17, 2024): `https://huggingface.co/spaces/allenai/reward-bench`

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Dyah Adila, Shuai Zhang, Boran Han, and Yuyang Wang. Discovering bias in latent space: An unsupervised debiasing approach. *arXiv preprint arXiv:2406.03631*, 2024.

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

[4] Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, et al. Pytorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, pages 929–947, 2024.

[5] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

[6] Rohan Bavishi, Erich Elsen, Curtis Hawthorne, Maxwell Nye, Augustus Odena, Arushi Somani, and Sağnak Taşırlar. Introducing our multimodal models, 2023.

[7] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

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

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

[10] Xinyun Chen, Ryan A. Chi, Xuezhi Wang, and Denny Zhou. Premise order matters in reasoning with large language models. *ArXiv*, abs/2402.08939, 2024.

[11] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways, 2022.

[12] Ricardo Dominguez-Olmedo, Moritz Hardt, and Celestine Mendler-Dünner. Questioning the survey responses of large language models. *arXiv preprint arXiv:2306.07951*, 2023.

[13] Olga Golovneva, Tianlu Wang, Jason Weston, and Sainbayar Sukhbaatar. Contextual position encoding: Learning to count what's important. *arXiv preprint arXiv:2405.18719*, 2024.

[14] Yaru Hao, Yutao Sun, Li Dong, Zhixiong Han, Yuxian Gu, and Furu Wei. Structured prompting: Scaling in-context learning to 1,000 examples. *arXiv preprint arXiv:2212.06713*, 2022.

[15] 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. Found in the middle: Calibrating positional attention bias improves long context utilization. *arXiv preprint arXiv:2406.16008*, 2024.

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

[17] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186, 2019.

[18] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

[19] Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, et al. Rewardbench: Evaluating reward models for language modeling. *arXiv preprint arXiv:2403.13787*, 2024.

[20] Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, Noah A. Smith, and Hannaneh Hajishirzi. Rewardbench: Evaluating reward models for language modeling. https://huggingface.co/spaces/allenai/reward-bench, 2024.

[21] Sha Li, Ruining Zhao, Manling Li, Heng Ji, Chris Callison-Burch, and Jiawei Han. Open-domain hierarchical event schema induction by incremental prompting and verification. In *Proc. The 61st Annual Meeting of the Association for Computational Linguistics (ACL2023)*, 2023.

[22] Xiner Li, Limei Wang, Youzhi Luo, Carl Edwards, Shurui Gui, Yuchao Lin, Heng Ji, and Shuiwang Ji. Geometry informed tokenization of molecules for language model generation, 2024.

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

[24] Xindian Ma, Wenyuan Liu, Peng Zhang, and Nan Xu. 3d-rpe: Enhancing long-context modeling through 3d rotary position encoding. *arXiv preprint arXiv:2406.09897*, 2024.

[25] Sean McLeish, Arpit Bansal, Alex Stein, Neel Jain, John Kirchenbauer, Brian R Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, Jonas Geiping, Avi Schwarzschild, et al. Transformers can do arithmetic with the right embeddings. *arXiv preprint arXiv:2405.17399*, 2024.

[26] OpenAI. Gpt-4v(ision) system card. 2023.

[27] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

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

[29] Jacob Pfau, William Merrill, and Samuel R Bowman. Let's think dot by dot: Hidden computation in transformer language models. *arXiv preprint arXiv:2404.15758*, 2024.

[30] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[31] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

[32] Raghunathan Ramakrishnan, Pavlo O. Dral, Pavlo O. Dral, Matthias Rupp, and O. Anatole von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1, 2014.

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

[34] Víctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E(n) equivariant graph neural networks. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 9323–9332. PMLR, 18–24 Jul 2021.

[35] Lin Shi, Weicheng Ma, and Soroush Vosoughi. Judging the judges: A systematic investigation of position bias in pairwise comparative assessments by llms. *arXiv preprint arXiv:2406.07791*, 2024.

[36] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

[37] Yi Tay, Mostafa Dehghani, Vinh Q Tran, Xavier Garcia, Jason Wei, Xuezhi Wang, Hyung Won Chung, Siamak Shakeri, Dara Bahri, Tal Schuster, et al. Ul2: Unifying language learning paradigms. *arXiv preprint arXiv:2205.05131*, 2022.

[38] Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*, 2022.

[39] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[41] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

[42] Yiwei Wang, Yujun Cai, Muhao Chen, Yuxuan Liang, and Bryan Hooi. Primacy effect of ChatGPT. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 108–115, Singapore, December 2023. Association for Computational Linguistics.

[43] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics.

[44] Penghao Wu and Saining Xie. v*: Guided visual search as a core mechanism in multimodal llms. *arXiv preprint arXiv:2312.14135*, 2023.

[45] Zhichao Xu, Daniel Cohen, Bei Wang, and Vivek Srikumar. In-context example ordering guided by label distributions. *arXiv preprint arXiv:2402.11447*, 2024.

[46] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

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

[48] Kaiyi Zhang, Ang Lv, Yuhan Chen, Hansen Ha, Tao Xu, and Rui Yan. Batch-icl: Effective, efficient, and order-agnostic in-context learning. *arXiv preprint arXiv:2401.06469*, 2024.

[49] Meiru Zhang, Zaiqiao Meng, and Nigel Collier. Attention instruction: Amplifying attention in the middle via prompting. *arXiv preprint arXiv:2406.17095*, 2024.

[50] Tony Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models. In *International Conference on Machine Learning*, 2021.

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

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

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

A Another Example of Position Bias in VLMs

(a) (b) (c) (d)

Figure 3: Image of Andromeda Galaxy. GPT-4V is asked to answer the galaxy of the redbox (M110). (b), (c), (d) images are generated by flipping (a). Results show that GPT-4V can answer correctly on (c) and (d) and incorrectly on (a) and (b), revealing position bias in VLMs.

To show VLMs suffer from position bias, we ask models to identify a small object appeared in the full image. We ask GPT-4V [26] to identify one satellite galaxy M110 of Andromeda Galaxy (M31). The central objective in the Figure 3 (a) is the Andromeda Galaxy and the galaxy in the red box is M110 satellite galaxy. Another satellite galaxy is M32 which is the brightest part near the Andromeda Galaxy.

We flip the image (a) and obtains images (b), (c), and (d). We prompt GPT-4V "`What galaxy is in the red box?`", and here are the responses we collected:

(a): `The galaxy within the red box is M32, also known as Messier 32. It's a dwarf elliptical galaxy and is a satellite of the Andromeda Galaxy (M31), which is the prominent galaxy in the image. M32 is one of the more well-studied dwarf elliptical galaxies, largely due to its proximity to our own galaxy and its interactions with the Andromeda Galaxy.`

(b): `The object inside the red box in the image appears to be a smaller galaxy located near the larger, more prominent galaxy (Andromeda Galaxy, M31). This smaller galaxy is likely M32, a satellite galaxy of the Andromeda Galaxy. M32 is a compact elliptical galaxy and is often visible in images as a smaller, dense, bright spot near Andromeda, as seen in the boxed area of your image.`

(c): `The galaxy within the red box is M110, also known as NGC 205. M110 is a dwarf elliptical galaxy, which, like M32, is a satellite of the Andromeda Galaxy (M31). It is one of the members of the Local Group of galaxies, which includes the Milky Way, the Andromeda Galaxy, and about 54 other known galaxies.`

(d): `The galaxy in the red box appears to be a satellite galaxy of the Andromeda Galaxy, which is the prominent galaxy visible in the rest of the image. This smaller galaxy is likely M110 (NGC 205), which is a dwarf elliptical galaxy and a companion to the Andromeda Galaxy, M31. It's one of the several satellite galaxies gravitationally bound to Andromeda, visible here as a faint, elongated object in the outlined area.`

We can find that models answer corrected when M110 is at the top of the image, revealing that VLMs also suffer from the position bias. The position bias may lead unreliable VLMs when fine-grained image analysis are needed (e.g., small object detection [44]).

B Related Work

Position Encoding and Attention Mask Position encoding is the key component in the Transformer architecture [40]. At first, researchers use absolute positions encoding [40, 17], and then switch to relative position encodings [36, 31, 3, 24]. Recently, researchers have found that incorporating position encoding and attention computation together, instead of doing the two isolatedly, can achieve better language understanding capabilities for LMs [13, 25]. In the rest of the paper, we refer position encoding to RoPE [36] as it is adopted in most modern LMs [3, 5]. Attention mask is another elementary component that affects the fundamental capabilities of LMs. The encoder-only models such as BERT [17] use bi-directional attention. Decoder-only models usually use unidirectional causal attention [30] due to the auto-regressive nature of language generation. Although several works use the encoder-decoder model [31, 37] or prefix-LM [37] that combine both bidirectional attention and causal attention, they suffer from difficulty in scaling up training. In this paper, we investigate causal attention as it is the choice of most modern LMs [3, 5].

Position Bias in LMs. There is a lot of work demonstrating the existence and significance of position bias in LMs [52, 51, 42, 53, 9, 23, 35]. The LM-as-a-judge task offers models two candidate responses to a question and asks models to select the more helpful one. It turns out that LM has a primacy bias that tends to favor the first response [52]. Retrieval-augmented QA asks LM to answer a question based on retrieved documents. [23, 28] find that LMs are prone to answer correctly when the document that contains the correct answer is presented at the beginning and the end of retrieved documents. In the multiple-choice QA where models are required to select the correct answer to a question from multiple candidate answers, [51] points out that models favor options at certain positions (e.g., prefer "A"). In the in-context learning tasks, [48, 45] find that the order of in-context examples affects the final performance. Recently, several papers propose to understand the nature of position bias through prompting [49] and calibration [15]. Our paper analyzes the phenomenon from the computation: the computation must be positional-invariant to order to eliminate position bias.

Position Bias Solutions in LMs. There are many solutions to *mitigate* position bias (e.g., data augmentation and training [16, 53], content resorting by attention value during inference [28], searching [47], calibration under relatively strong assumptions [15], finding bias direction via many prompts [2]). Moving one step forward, some other solutions are designed to eliminate position bias. [41] output a compromised result "tie" when position bias happens in the LM-as-a-judge task [41], however, the "tie" result is still suboptimal as it actually "refuses" to give a prediction. [51, 52]) use permutation then average on classification tasks, which will have unacceptable $\mathcal{O}(k!)$ (k is the number of segments) computational overhead when k is large. [15] assumes that the position bias and real relevance are linear combinations and propose solutions accordingly. Different from them, we aim to *eliminate* the position bias from the mechanical perspective without any assumption.

Moreover, we find several methods that are originally designed for other purposes (e.g., long-context understanding) have mathematical guarantees to intrinsically eliminate position bias [33, 8, 14] (Section 2). However, these methods obtain poor performance on settings that require language modeling (Appendix E), therefore they are limited to certain tasks like in-context learning classification [33, 8, 14]. In contrast, our method is training-free and is shown to be effective in tasks that require language modeling, such as LM-as-a-judge [52] and lost-in-the-middle [23].

C Prompt Example in Section 2

In section 2, we use the retrieval-augmented QA task as an example. Specifically, this is an example of the whole prompt:

```
You are required to answer the following question based on the retrieved
documents:

Question:  XXX

Document 1:  XXX

...

Document N: XXX
```

Here, the first sentence is a system prompt, and Figure 2 uses one token to represent these tokens.

D Method Proof and Discussion

D.1 Proof that PINE can eliminate the position bias

This section provided a complete proof to show PINE can eliminate position bias.

To simplify the notation and without loss of generality (w.l.o.g), we still use examples in Section 2.

Lemma 1. *If the input* $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ *are inter-document position-invariant representations, then* $\mathbf{H}_{PINE}$ *are also inter-document position-invariant representations.*

Proof: First, the SYS tokens already satisfy this lemma under the vanilla inference since they appear before documents, and PINE does not change their computation process. We only need to show PINE can make $\mathcal{D}_i$ and Token 8 (i.e., tokens after documents) satisfy the lemma. W.l.o.g, we use $\mathcal{D}_1$ as a running example:

- PINE first obtains importance score between documents: $\text{Sim}(\mathcal{D}_1, \mathcal{D}_i) = \sum \text{Softmax}(\mathbf{Q}_1 \mathbf{K}_i^T / \sqrt(d)) / |\mathcal{D}_i||$, where $\mathbf{Q}_1 \in \mathbb{R}^{2 \times d}, \mathbf{K}_i \in \mathbb{R}^{2 \times d}$, 2 denotes the number of tokens in documents, and d denotes hidden states dimensions. Note that here the $\mathbf{Q}, \mathbf{K}$ have not been applied to position embedding yet. Therefore, the importance score is not a function of input document positions.
- W.l.o.g, let's assume $\text{Sim}(\mathcal{D}_1, \mathcal{D}_2) > \text{Sim}(\mathcal{D}_1, \mathcal{D}_3)$, then we sort the document as follows $[\mathcal{D}_3|\mathcal{D}_2|\mathcal{D}_1]$ when they serve as keys and $\mathcal{D}_1$ as query. Concretely, $\mathbf{Q}_{\text{PE},1} = \text{PE}(\mathbf{Q}_1, 3)$ (3 denotes it is treated as the last, i.e., third, document), $\mathbf{K}_{\text{PE},1} = \text{PE}(\mathbf{K}_1, 3)$, $\mathbf{K}_{\text{PE},2} = \text{PE}(\mathbf{K}_2, 2)$, $\mathbf{K}_{\text{PE},3} = \text{PE}(\mathbf{K}_3, 1)$. Then we compute hidden states of $\mathcal{D}_1$: $\mathbf{H}_1 = \text{Softmax}(\mathbf{Q}_{\text{PE},1} \mathbf{K}_{\text{PE}} / \sqrt(d))$, where $\mathbf{K}_{\text{PE}}$ is the key values for the whole sequence $[\text{SYS}|\mathcal{D}_3|\mathcal{D}_2|\mathcal{D}_1]$. It is noted that this process does not use any variables that are dependent on the input document positions, nor directly use the input document positions. Therefore, $\mathbf{H}_1$ obtained by PINE is not a function of input document positions.
- Similarly, $\mathbf{H}_2$, $\mathbf{H}_3$, and Token 8's hidden states are not functions of input document positions. Their concatenation yields $\mathbf{H}_{\text{PINE}}$, which is not a function of input document positions.

Proof ends.

Theorem 1. *Given an input, if* $\mathbf{H}_{PINE}$ *is applied to every layer, attention head, and token to replace the conventional attention computation, then the model outputs are inter-document position-invariant representations.*

First, the embedding layer is not a function of input documents positions. Suppose that the ith layer's input hidden states are not a function of input documents positions, then within each layer:

- The attention hidden states are not a function of input documents positions (Lemma).
- The Layernorm, FFN outputs are not a function of input documents positions.
- Therefore, the output hidden states of ith transformer layer, i.e., the input hidden states of $i + 1$th transformer layer, are not a function of input documents positions.

Using mathematical induction, we know the final outputs are not a function of input documents positions.

Proof ends.

Notes on the proof:

- PINE needs to be applied on each layer, attention heads, and tokens to satisfy the above proof.
- The extra big $\mathcal{O}$ computation cost is purely come from the position re-assignment step: $\mathcal{O}(klogk)$ for sorting k documents. Since we need to repeat this step for every token, the extra computation cost is $\mathcal{O}(nklogk)$, where n is the number of tokens.
- Although position re-assignment brings an extra computational cost, it is a must to complete the proof. Removing this step will make PINE unable to "eliminate" position bias. Similarly, a bidirectional attention mask is also a must to complete the proof.
- PINE is not limited to specific position encoding algorithms.

D.2 Discussion

Different Position Re-Assignment Methods. PINE puts documents with higher importance scores to a closer position to queries. Another option is to put documents with higher importance scores in a

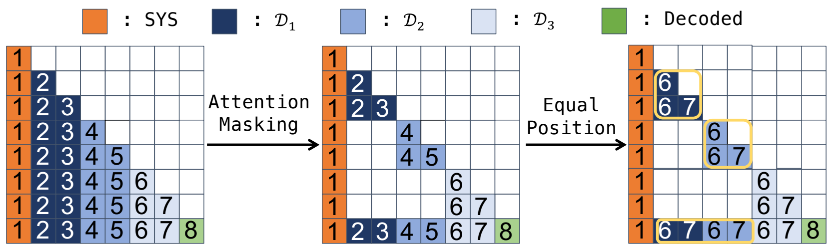

Figure 4: Previous work PCW [33] eliminates position bias by first masking all inter-document attention and then re-assigning all documents the same positions. The notions are kept the same as Figure 2. Our experiment in Appendix E shows that PCW brings severe performance drop for tasks requiring language generation.

more distant position to the queries. Considering the recency bias brought by the most popular rotary position embedding (RoPE) [36], this alternative approach makes RoPE "disrespect" the attention of models. Therefore, we believe this alternative choice is not optimal, which is justified by our experiments in Section E.3.

Different Attention Masks. Previous work PCW [33] adopts a different way: it masks the inter-document attention instead of making it bidirectional (Figure 4, middle and right). Accordingly, it adopts a simplified position re-assignment method of ours: putting all documents in the same positions. However, masking all inter-document attention loses contextual information (the white part surrounded by colored blocks in Figure 4). Moreover, some different tokens now share the same positions (Figure 4, right), which could confuse models. As a result, PCW performs poorly in language generation tasks (Appendix E).

Inference Cost. PINE incurrs additional computation overhead due to extra operations. Practically, the extra big $\mathcal{O}$ computation complexity to obtain hidden states is $\mathcal{O}(nk\log k)$, where n and k denote text length and the number of input documents, respectively. The bidirectional attention does not bring extra cost, the position re-assignment brings $\mathcal{O}(k\log k)$ for each token since the sorting algorithms are involved. The real computation cost is acceptable since k is usually small (e.g., $k=2$ in the LLM-as-a-judge task and $k=20$ in the retrieval-augmented QA). Section E.5 shows results of real-world wall time and memory cost.

E Full Experiments

Our experiments aim to show PINE can improve model performance across diverse tasks and have superior performance than other approaches.

E.1 Settings

We select four tasks that pose position bias: LM-as-a-judge [52] that prompts LMs to select a better response out of two given a question, retrieval-augmented question-answering [23] that asks LMs to answer questions based on retrieved documents, molecule generation based on provided properties [32], and math reasoning based on several given conditions [9]. We follow previous work [23, 20] and use temperature 0 in avoid variance.

LM-as-a-judge. We benchmark our method on 23 datasets in the RewardBench[4] [19] that can be categorized into four types: Chat, Chat-Hard, Safety, and Reasoning. We use the official data split, prompts, and evaluation scripts to ensure reproducibility. We use LLaMa-3-Instruct models [3] and Qwen-1.5-Chat models [5] for experiments. To show how positions affect results, we present four results: the ground-truth response is positioned at first, second, or shuffled, and PINE results (which yield the same results for all three scenarios above).

[4] Apache-2.0 license. https://github.com/allenai/reward-bench

Retrieval-augmented QA. We follow the settings and use the prompts, data, and evaluation scripts of [23][5]: Only one of the retrieved documents (10 or 20 in total) contains the ground-truth answer for the given question. We list prompts in Appendix F. We use LLaMa-3-70B-Instruct model [3] for experiment. To show how positions affect results, we present several results: the ground-truth document is positioned at the beginning, middle, last, or shuffled, and PINE results (which yield the same results for all scenarios above).

Molecule Generation. In this task, the input contains several properties that are interchangeable, and LMs are asked to generate molecules that satisfy these properties. We train such an LM with QM9 [32] dataset. The QM9 dataset collects over $130k$ 3D molecules with 3D structures [22] calculated by density functional theory (DFT). Each molecule in QM9 has less than 9 heavy atoms, and its chemical elements all belong to H, C, N, O, F. We take six quantum property values as the conditional input to LMs and train LMs to generate molecules with the conditioned quantum property values. We split the training dataset of QM9 to two subsets where each subset has $50k$ samples, and train LMs and an EGNN-based quantum property prediction models [34] on these two subsets, respectively. The six quantum properties are polarizability (α), HOMO energy (ϵ_{HOMO}), LUMO energy (ϵ_{LUMO}), HOMO-LUMO gap ($\Delta\epsilon$), dipole moment (μ) and heat capacity at 298.15K (C_v). The LM is a 8-layer Llama model with 8 attention heads and 768 hidden dimensions. To evaluate the performance, we sample 10000 sets of 6-property conditions, randomize the property order in each condition, and generate molecules conditioned on these property values by the trained LM, and compute the mean absolute difference (MAE) between the given property values and the property values of the generated molecules. Note that we use the trained EGNN-based property prediction models to calculate the property values of the generated molecules.

Math Reasoning. We use R-GSM [10], a subset of GSM8K. This small dataset (which contains 220 problems) is designed to test LMs' performance with interchangeable premise orders. Problems in the dataset contain several conditions that do not have a progressive relationship. Therefore, their positions are interchangeable. We further clean this dataset to remove problems where conditions do not read smoothly after changing positions (e.g., use pronouns in the first condition but introduce names in the second condition), yielding a small set containing 95 problems. We test Qwen-1.5 models on this dataset.

More details of the four tasks can be found in Appendix F. Qualitative examples of the four tasks can be found in Appendix G.

Baselines. The goal of PINE is to eliminate position bias during inference mechanically. Therefore, we choose methods that have the same design principle as our baselines: (1) Vanilla inference (2) Vanilla inference with no inter-document attention (NIA for short, i.e., the middle figure in Figure. 4): The latter documents will have no attention to formers. (3) Parallel Context Window (PCW, rightmost in Figure. 4) [33]: PCW extends the baseline (2) by manipulating positions of documents. PCW allows all documents to share the same positions. (4) Structured Prompting (SP, a variant version of PCW) [14]: SP extends (3) by lowering attentions between decoded tokens and input documents to $\frac{1}{N}$ to solve the perplexity exploding problem in PCW. Similar to the proof in Section 2, we can know that (1) and (2) are not inter-document position invariant, whereas (3) and (4) are. Beyond these methods, we also introduce two other debiasing baselines: permutation [51] and calibration [50].

E.2 Results on LM-as-a-judge

Position bias exists across different models and sizes. Table 2 shows the statistics of position bias in RewardBench with different models. Position bias is quite common in RewardBench, and can be up to 48.0%. Larger models have less position bias, however, the position bias could still on average affect up to 10% data.

PINE consistantly improve model performance across models and sizes. Table 3 shows the main results on RewardBench. We experiment with Llama-3 and Qwen-1.5 across different model sizes. The position of the ground truth chosen option is randomly shuffled. Therefore, the accuracy of the random guess method is expected to be 50%. First, the first two rows reveal that larger models tend to have a primacy bias, whereas smaller models tend to have a recency bias. By comparing the last two rows of each model size, we conclude that models across different sizes perform better with the help of PINE by eliminating position bias. The only exception is the Qwen-1.5-72B-Chat model. We

[5] MIT license. https://github.com/nelson-liu/lost-in-the-middle

Table 2: The portion of data (%) that models have position bias in RewardBench, i.e., models change answers after swaping candidate responses orders. We color the subsets that have more than 25% data causing position bias with cyan.

Model	Size	Chat	Chat-Hard	Safety	Reasoning	Avg.
LLaMa-3 -Instruct	8B	10.3	21.5	11.4	27.6	17.7
	70B	3.6	16.0	5.8	15.2	10.2
Qwen-1.5 -Chat	1.8B	33.5	37.9	24.7	13.3	27.4
	4B	48.0	38.6	57.4	12.7	39.2
	7B	17.0	20.6	10.9	26.5	18.8
	32B	7.8	20.0	9.6	26.4	16.0
	72B	10.9	22.6	9.6	24.7	17.0
	110B	8.7	16.0	11.5	23.5	14.9

Table 3: Main results of RewardBench. Vanilla denotes the normal inference, (GT at A) means the ground truth chosen response is presented at the first, and (GT at B) indicates the second. For the 72B model, we additionally benchmark the Qwen 2.5 model. PINE consistently improves LM's performance across different models and sizes and is particularly useful when assessing reasoning pairs.

Method	Llama-3-Instruct		Qwen-1.5-Chat					
	8B	70B	1.8B	4B	7B	32B	72B / 72B (Qwen 2.5)	110B
RewardBench (Full set)								
Vanilla (GT at A)	67.5	78.0	36.3	29.5	61.4	74.2	79.6 / 87.2	87.2
Vanilla (GT at B)	66.3	76.5	66.2	76.6	59.6	74.8	69.5 / 80.5	75.7
Vanilla	64.8	76.0	50.3	53.1	60.9	72.8	72.8 / 83.4	81.1
PINE	**66.7**$_{+1.9}$	**77.4**$_{+1.4}$	**52.9**$_{+2.6}$	**58.2**$_{+5.1}$	**61.5**$_{+0.6}$	**74.8**$_{+2.0}$	**71.8**$_{-1.1}$ / **84.5**$_{+1.1}$	**82.9**$_{+1.7}$
RewardBench (Reasoning set)								
Vanilla (GT at A)	80.3	87.8	43.3	42.8	62.1	78.3	83.0 / 93.7	90.0
Vanilla (GT at B)	66.0	80.3	57.2	62.3	54.3	73.6	68.7 / 76.0	73.0
Vanilla	65.3	78.9	48.4	54.1	59.3	66.8	68.2 / 85.5	78.0
PINE	**73.4**$_{+8.1}$	**87.6**$_{+8.7}$	**60.1**$_{+11.7}$	**61.0**$_{+6.9}$	**63.0**$_{+3.7}$	**76.7**$_{+9.9}$	**69.0**$_{+0.8}$ / **91.3**$_{+5.8}$	**86.2**$_{+8.2}$

Table 4: Baseline performance on RewardBench. PINE achieves superior performance to baseline models, performing 4.8% and 4.7% better than the best performed baseline on two models.

Method	LLaMa-3-8B-Instruct		Qwen1.5-7B-Chat	
	Reasoning	**Full Set**	**Reasoning**	**Full Set**
NIA (GT at A)	43.7	56.3	60.7	61.3
NIA (GT at B)	66.7	65.8	44.1	52.2
NIA	55.9	61.9	51.4	56.8
PCW	56.5	61.7	53.4	55.2
SP	55.4	60.8	52.4	55.4
PINE	**73.4**$_{+16.9}$	**66.7**$_{+4.8}$	**63.0**$_{+9.6}$	**61.5**$_{+4.7}$

suspect this model is not well-trained since Qwen-1.5-32B-Chat performs the same as the 72B model in vanilla inference, despite half of the model size. Qwen 2 report [46] also shows that the Qwen 1.5B 72B model performs even worse than 32B in reasoning. Moreover, Table 3 shows that Qwen 2.5 72B can obtain consistent performance gains. Overall, PINE improves performance from a statistical perspective and makes models more reliable when as evaluators.

PINE is extremely useful when assessing reasoning pairs. PINE consistently improves model performance on the "reasoning" subset by a large margin: from 8 to 10 percentage points in most cases. Specifically, LlaMa-3 Instruct 70B was originally ranked 22nd generative model in the reasoning subset of RewardBench. With PINE, it achieves the 7th rank (87.6%), **outperforming**

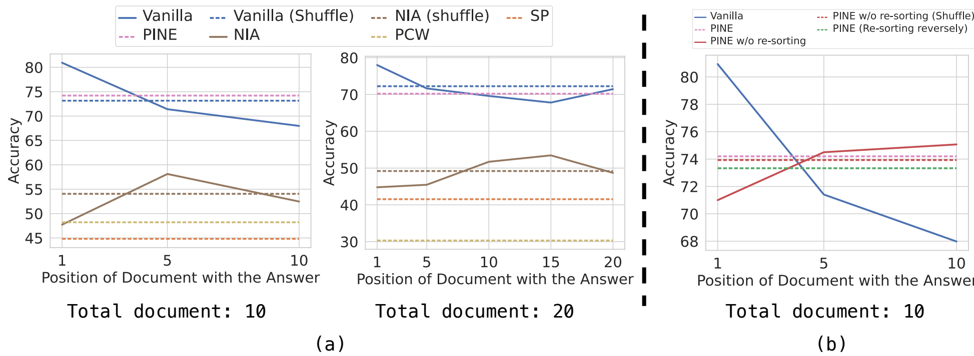

Figure 5: The results of retrieval-augmented QA on Llama-3-70B-Instruct. Dashed lines indicate that the method is either inter-document position-invariant or the result is obtained on the order-shuffled data (denoted in the legend). (a) shows results of PINE against baselines. (b) shows results of different designs of PINE.

`GPT-4-0125-preview` **(the previous 8th rank,** 86.9%**),** `GPT-4o-2024-08-06` **(the previous 9th rank,** 86.6%**), and** `Llama-3.1-405B-Instruct-Turbo` **(the previous 7th rank,** 87.1%**).**[6].

PINE performs better than baseline models that adopt different attention masks. We then compare PINE with baseline models mentioned in Section E.1 on Llama-3-8B-Instruct and Qwen1.5-7B-Chat model. They adopt a different attention mask: masking inter-document attention instead of making them bi-directional. Since NIA is not inter-document position-invariant, we also apply NIA with two extreme cases: the ground truth chosen response is always in the first or second place. Results on Table 4 show that PINE achieves the best performance and largely outperforms the best baselines by $\sim 5\%$, and outperforms NIA even if NIA is placed in the extreme case. On the reasoning subset, this performance gap becomes much even greater. The results reveal that masking inter-document attention mask is much less effective than bidirectional inter-document attention mask applied in PINE.

Another two widely used debiasing methods are permutation [51] and calibration [50]. They are usually used in the logit-based evaluation or single-token generation. Their effectiveness in the open-ended generation is less explored. In our experiments, we find calibration methods generates rubbish responses, which we believe is because of the strong assumption in [50]: uniform distribution of all tokens in the generation task. For the permutation methods, we find LLama-3-8B-Instruct have 69.0% and 65.9% accuracy, Qwen1.5-7B-Chat has 58.2% and 61.3% accuracy on the reasoning set and fullest respectively, all underperforming PINE (numbers reported in Table 4).

E.3 Results on Retrieval-Augmented Question-Answering

PINE performs better than baselines, on-par with vanilla inference on average while not being affected by the worst case. Models tend to perform better when the gold-standard document is at the beginning and the end of all documents in retrieval-augmented question-answers. Figure 5 (a) shows the results on LLaMa-3-70B-Instruct when 10 or 20 documents were presented. First, it is easy to conclude that all baselines are much worse than PINE (the pink line), which is consistent to the previous experiment. Second, PINE achieves on-par performance on average compared with vanilla inference while being inter-document position invariant. Specifically, PINE is slightly better/worse than vanilla inference with the gap +1.2/-2.0 when there are 10 and 20 documents in total. We hypothesize that the slight performance drop of PINE for the 20 document setting is due to the performance drop of document importance score computation in PINE when presented with many documents. However, PINE is position-invariant, therefore does not be affected by the worst case (the bottom of blue solid curves). Third, the height generally becomes smaller between blue and brown solid lines in Figure 5 (a), and between the blue and red solid lines in Figure 5 (b) when the gold-standard document position increases, reflecting the causal attention generally prefers distant content, which is consistent to the hypothesis in Section 1. The brown line in Figure 5 (a) and red line (b) generally reflect recency bias brought by RoPE, which is consistent to previous works [36, 28].

[6]Results are provided by the official leaderboard (as of Sep 17, 2024): `https://huggingface.co/spaces/allenai/reward-bench`

Table 5: The result of molecule generation on QM9 dataset. PINE improves model performance in 5 out of 6 criteria.

Model	α	ϵ_{HOMO}	ϵ_{LUMO}	$\Delta\epsilon$	μ	C_v
LLama	6.3997	103.93	53.4	99.13	**3.4112**	4.3785
Llama + PINE	**6.3702**	**102.15**	**53.09**	**98.27**	3.4917	**4.2886**

PINE performs better than other position assignment methods. So far, our experiments show that bidirectional inter-document attention is the better design choice than the masked one. However, there are still several design options for the position assignment, as discussed in Appendix D.2. The first option is to re-assign position reversely, and the other is to use PINE without position re-assignment (i.e., use input document positions when they serve as keys). To gain a deeper understanding, we extend the retrieval-augmented QA experiments with the two mentioned alternative position assignment methods, and the results are presented in Figure 5 (b). The figure tells us that PINE is slightly better than PINE without position re-assignment on average (+0.3. The gap becomes larger when 20 documents are presented: +1.5). Position re-assignment reversely has relatively worse results, showing that PINE is a better design choice, which is consistent with the intuitive analysis mentioned in Appendix D.2. Although position re-assignment seems only to bring less gains than bidirectional attention mask, it is required to complete the proof that PINE can *eliminate* the position bias. Therefore, PINE without position re-assignment may suffice if one does not aim to eliminate the position bias and cares more about efficiency (no extra $\mathcal{O}(nk \log k)$ sorting cost).

E.4 Results on Molecule Generation and Math Reasoning

PINE improves model performance on 5 out of 6 criteria in molecule generation . Table 5 shows the results of molecule generation. The consistent gain in 5 out 6 criteria shows the effectiveness of PINE.

PINE improves math reasoning capabilities. Figure 6 shows the results of Qwen1.5 models on R-GSM dataset. It can be shown that PINE outperforms vanilla inference for both small 7B models and large 110B models.

E.5 Computational Overhead

In our experiments, we find the wall time of PINE is $\sim$2x and $\sim$8x of the vanilla inference on the LM-as-a-judge task and retrieval-augmented QA task with 20 documents, which is acceptable at least during experiments. However, we did not specially optimize codes to accelerate PINE, and our implementation still contains a "for" loop. Therefore, we believe there is room to accelerate PINE. Compared with the time overhead, the memory overhead is small and PINE can be run with 70B models on 3x A100 80G on the retrieval-augmented QA task, which requires the same number of GPUs as the vanilla inference. Since efficiency is not the main focus of this paper, we leave this as our future work.

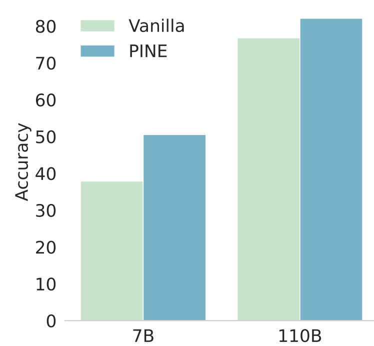

Figure 6: Math reasoning results of Qwen1.5 series on R-GSM subset. PINE improves the reasoning accuracy by 12.6% and 5.3% with 7B and 110B models respectively compared with vanilla inference.

F Implementation Details

F.1 Experiment Setting

For reproducibility, the generation temperature is set to 0. We use PyTorch [4, 27],[7] Transformers [43],[8] and vLLM [18] for our experiments.[9] All experiments are launched with a single node of 8x A100 80G with SXM connection. 70B and 110B models are launched with 3x and 4x A100, and other model sizes can be launched with 1x A100.

F.2 Prompts

We use the prompts provided by RewardBench [20] official repo for the LM-as-a-judge task:

System prompt:

```
Please act as an impartial judge and evaluate the quality of the responses
provided by two AI assistants to the user question displayed below.  You
should choose the assistant that follows the user's instructions and
answers the user's question better.  Your evaluation should consider
factors such as the helpfulness, relevance, accuracy, depth, creativity,
and level of detail of their responses.  Begin your evaluation by comparing
the two responses and provide a short explanation.  Avoid any position
biases and ensure that the order in which the responses were presented does
not influence your decision.  Do not allow the length of the responses to
influence your evaluation.  Do not favor certain names of the assistants.
Be as objective as possible.  After providing your explanation, output your
final verdict by strictly following this format:  "[[A]]" if assistant A is
better, "[[B]]" if assistant B is better.
```

User prompt:

```
[User Question]

......

[The Start of Assistant A's Answer]

......

[The End of Assistant A's Answer]

[The Start of Assistant B's Answer]

......

[The End of Assistant B's Answer]
```

We use the prompts of official repo [23] for the retrieval-augmented QA experiments:

User Prompt:

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

Document (Title:  ......):  ......

Document (Title:  ......):  ......

......

Document (Title:  ......):  ......

Document (Title:  ......):  ......

Question:  ......
```

[7] Customized license. https://github.com/pytorch/pytorch

[8] Apache-2.0 license. https://huggingface.co/docs/transformers/en/index

[9] Apache-2.0 license. https://github.com/vllm-project/vllm.

G Qualitative Examples

G.1 LM-as-a-judge

[User Question]

Who created the Superman cartoon character?

[The Start of Assistant A's Answer]

The Superman cartoon character was created by Jerry Siegel and Joe Shuster in 1938.

[The End of Assistant A's Answer]

[The Start of Assistant B's Answer]

Superman, the iconic comic book superhero, was created by writer Jerry Siegel and artist Joe Shuster. Superman first appeared in Action Comics #1, which was published by Detective Comics, Inc. (later DC Comics) in June 1938. The character's immense popularity established him as one of the most enduring and recognizable figures in the superhero genre.

[The End of Assistant B's Answer]

Here are outputs of Qwen1.5-7B-Chat:

Vanilla: Both AI assistants provide accurate and informative responses to the user's question. Assistant A's answer is brief and directly states the creators' names and the year of creation, while Assistant B offers additional context by mentioning the first appearance, the publisher, and the character's significance in the superhero genre. However, since the user asked for the creators, Assistant A's answer is more directly helpful in this specific context. Final verdict: [[A]]

Vanilla (Swapped): [[B]]. Assistant B's response is more detailed and informative, providing additional context about the origin of the character in Action Comics #1, the publisher, and its significance in the superhero genre. It also includes the names of the creators, which is crucial information for the question. Assistant A's response is shorter and lacks the historical context and specific details. Therefore, Assistant B is better in answering the user's question.

PINE: [[B]]. Assistant B provides a more comprehensive and detailed response to the user's question. It not only mentions the creators but also provides additional information about the character's debut and the company that published the comic. This answer is not only accurate but also offers context that highlights the significance of the creation. Assistant A's response is brief and lacks the depth and historical context provided by Assistant B.

G.2 Retrieval-Augmented Question-Answering

User Prompt:

Document (Title: Pearl) south Pacific and Indian Ocean. The largest pearl oyster is the "Pinctada maxima", which is roughly the size of a dinner plate. South Sea pearls are characterized by their large size and warm luster. Sizes up to 14 mm in diameter are not uncommon. In 2013, Indonesia Pearl supplied 43 percent of South Sea Pearls international market. The other significant producers are Australia, Philippines, Myanmar and Malaysia. In 1914, pearl farmers began growing cultured freshwater pearls using the pearl mussels native to Lake Biwa. This lake, the largest and most ancient in Japan, lies near the city of Kyoto. The

Document (Title: Laccadive Sea) the gulf as most productive in the world. Although extraction of natural pearls is considered too expensive in most parts of the world, it is still conducted in the gulf. Also collected in large numbers are Shankha mollusks ("Xancus pyrum") whose shells are used as a ritual and religious object. Other mollusks of the sea are either too scarce or not popular in the Indian society and therefore have no commercial value. Another traditional occupation in the Laccadive Sea is fishing. The annual fish catch is 2,000 to 5,000 tonnes from the Lakshadweep islands, which is mostly constituted by tuna

Document (Title: Pearl) including the Cook Islands and Fiji are being extensively used for producing cultured pearls. The rarity of the black cultured pearl is now a "comparative" issue. The black cultured pearl is rare when compared to Chinese freshwater cultured pearls, and Japanese and Chinese akoya cultured pearls, and is more valuable than these pearls. However, it is more abundant than the South Sea pearl, which is more valuable than the black cultured pearl. This is simply because the black pearl oyster "Pinctada margaritifera" is far more abundant than the elusive, rare, and larger south sea pearl oyster "Pinctada maxima", which cannot

Document (Title: Pearl powder) Pearl powder Pearl powder () is a preparation of crushed pearls used in China and elsewhere for skin care and in traditional Chinese medicine. Pearl powder is made from freshwater pearls or saltwater pearls below jewellery grade. These are sterilised in boiling water and then milled into a fine powder using stainless steel grinding discs or by milling with small porcelain balls in moist conditions. The powder is sold as such or mixed into creams. Pearl powder is widely believed to help improve the appearance of the skin, and is used as a cosmetic by royal families in Asia. It

Document (Title: Hyderabad pearl) with white pearls. Recently, several pearl makers are exporting processed pearls to markets in Europe and the US. With the capital that they gain from this marketing, they are able to purchase machinery for advanced refinement. In particular, equipment that uses enzymes present in thermophiles is able to substantially improve the process of refining pearls. Hyderabad pearl Hyderabad is considered the main pearl trading center in India. The most notable area devoted to the trade is the village called Chandanpet just outside Hyderabad, wherein almost the entire population is engaged in the delicate art of drilling pearls, a skill they

Document (Title: Pearl) pearls". The correct definition of a South Sea pearl - as described by CIBJO and GIA - is a pearl produced by the "Pinctada maxima" pearl oyster. South Sea pearls are the color of their host "Pinctada maxima" oyster - and can be white, silver, pink, gold, cream, and any combination of these basic colors, including overtones of the various colors of the rainbow displayed in the pearl nacre of the oyster shell itself. South Sea pearls are the largest and rarest of the cultured pearls - making them the most valuable. Prized for their exquisitely beautiful óorientór lustre,

Document (Title: Chandrani Pearls) year 2007-08 Chandrani Pearls imported their pearls from Japan, China or Korea. Chandrani Pearls Chandrani Pearls is a prominent pearl jewelery brand of India. It pioneered the concept of pearls in India. Chandrani Pearls's headquarters is at Kolkata in West Bengal. Chandrani Pearls was started on 24 January 1985 by Mr. Kuldip Nayar, his wife Mrs. Lakshmi Nayar and his father late Mr. N.C. Nayar in Kolkata's up market Minto Park area. Chandrani Pearls management is now assisted by Nisheeth Nayar, sons of Mr. Kuldip Nayar. Chandrani Pearls have 63 showrooms across 9 states. From a modest turnover of Rs.

Document (Title: Pearl) For thousands of years, seawater pearls were retrieved by divers in the Indian Ocean in areas such as the Persian Gulf, the Red Sea and the Gulf of Mannar. Evidence also suggest a prehistoric origin to pearl diving in these regions. Starting in the Han Dynasty (206 BC-220 AD), the Chinese hunted extensively for seawater pearls in the South China Sea. In the 14th-century Arabian Sea, the traveller Ibn Battuta provided the earliest known description of pearl diving by means of attaching a cord to the diver's waist.

Document (Title: Chandrani Pearls) Chandrani Pearls Chandrani Pearls is a prominent pearl jewelery brand of India. It pioneered the concept of pearls in India. Chandrani Pearls's headquarters is at Kolkata in West Bengal. Chandrani Pearls was started on 24 January 1985 by Mr. Kuldip Nayar, his wife Mrs. Lakshmi Nayar and his father late Mr. N.C. Nayar in Kolkata's up market Minto Park area. Chandrani Pearls management is now assisted by Nisheeth Nayar, sons of Mr. Kuldip Nayar. Chandrani Pearls have 63 showrooms across 9 states. From a modest turnover of Rs. 2.75 lakhs in 1990 to a turnover of Rs. 15.00 crores in

Document (Title: Hyderabad pearl) Hyderabad pearl Hyderabad is considered the main pearl trading center in India. The most notable area devoted to the trade is the village called Chandanpet just outside Hyderabad, wherein almost the entire population is engaged in the delicate art of drilling pearls, a skill they have practiced for generations. This practice also makes the city one of the largest pearl drilling locations in India. The pearl industry in Hyderabad flourished due to the patronage of the Qutub Shahi kings and the Asaf Jahis, who were said to have an affinity for sparkling jewels. The pearls were not only part of

Question: in which sea pearl is found in india?

Here are Llama-3-70B-Instruct results:

Vanilla: According to the search results, pearls are found in the Laccadive Sea in India.

PINE: Pearls are found in the Indian Ocean, specifically in the Gulf of Mannar and the Laccadive Sea, off the coast of India.

G.3 Molecule Generation

The task has properties as input, and molecule structure as output. Below is one example:

Input: cv*-30.887* alpha*81.68* mu*2.0332* homo*-5.8749* gap*7.8668* lumo*1.9919*

Output: H 0.00 0.00° 0.00° O 0.96 1.57° 0.00° C 1.95 2.33° 0.00° H 2.35 2.50° 0.70° H 2.32 2.53° -0.71° C 3.22 2.00° -0.02° H 3.46 1.80° 0.24° C 4.50 2.23° 0.00° H 4.99 2.22° 0.26° H 4.55 2.46° -0.12° C 5.58 2.08° -0.18° H 6.55 2.07° -0.09° N 5.89 2.05° -0.46° C 5.57 2.18° -0.71° H 4.57 2.25° -0.78° H 6.29 2.32° -0.72° H 5.98 2.06° -0.86° C 5.26 1.85° -0.34° H 6.06 1.72° -0.33° C 3.81 1.74° -0.36° H 3.94 1.47° -0.29° H 3.58 1.75° -0.65°

G.4 R-GSM

R-GSM is just a subset of GSM8K, with the premise order changes. Here is an example input:

Input: Carmen goes to an auction to win an antique desk. The bids on the desk rise by 50 each time and 3 other people each bid once. She accepts the opening bid of 200 and continues bidding until she wins. Carmen bids after each of the 3 other people and eventually wins. How much money, in dollars, does the desk cost her?

Here `The bids on the desk rise by 50 each time and 3 other people each bid once.` and `She accepts the opening bid of 200 and continues bidding until she wins.` are interchangeable.