Distance between Relevant Information Pieces Causes Bias in Long-Context LLMs

Anonymous ACL submission

Abstract

Positional bias in large language models (LLMs) hinders their ability to effectively process long inputs. A prominent example is the "lost in the middle" phenomenon, where LLMs struggle to utilize relevant information situated in the middle of the input. While prior research primarily focuses on single pieces of relevant information, real-world applications often involve multiple relevant information pieces. To bridge this gap, we present LONGPIBENCH, a benchmark designed to assess positional bias involving multiple pieces of relevant information. It includes various tasks and input lengths. Thorough experiments are conducted with three commercial and six open-source models. These experiments reveal that while most current models are more robust against the "lost in the middle" issue, there also exist noticeable biases related to the spacing of relevant information pieces. These findings highlight the importance of evaluating and reducing positional biases for long-context LLMs[1]

1 Introduction

Large language models (LLMs) (Zhao et al., 2023; Minaee et al., 2024) have made significant progress in various natural language processing tasks (Hendrycks et al., 2021; Han et al., 2021). In particular, applications such as code repository analysis (Chen et al., 2021) and information extraction (Kočiský et al., 2018) often require processing long texts, with context lengths reaching up to 200,000 tokens (Li et al., 2024; Zhang et al., 2024a). To address these demands, researchers have focused on enhancing LLMs' ability to handle extended inputs effectively (Chen et al., 2023; Han et al., 2024). As a result, multiple LLMs have been developed (Dubey et al., 2024; Team et al., 2024; OpenAI, 2024) which support context lengths of up to one million tokens.

Recent studies have shown that the position of relevant information significantly affects the performance of long-context LLMs (Liu et al., 2023; Lei et al., 2024; Hsieh et al., 2024). In "needle in a haystack" tasks, models struggle to utilize information located in the middle of the input, which is known as the "lost in the middle"

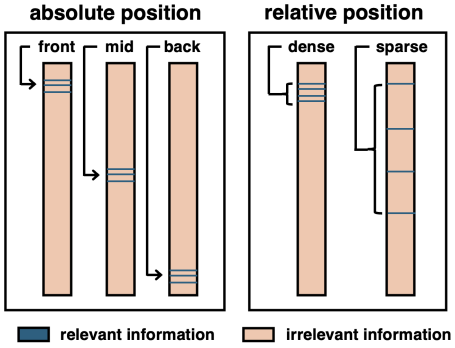

Figure 1: Illustration of absolute position and relative position. Absolute position refers to the location of relevant information within the entire context sequence, while relative position represents the distribution and distance between multiple relevant information pieces.

effect (Liu et al., 2023). This evaluation method is commonly used to analyze positional bias (Hengle et al., 2024; Nelson et al., 2024). These analyses (Liu et al., 2023) focused on single relevant information pieces and their positions in the input sequence (front, middle, back), which we refer to as **absolute positions**.

However, real-world tasks like data analysis (Zhang et al., 2024a) often involve multiple pieces of relevant information. This introduces a new characteristic: the distance between relevant information pieces, or how densely they are distributed, which we term as **relative position**. Evidence from two types of extreme cases indicates that varying relative position may lead to significant bias, impairing LLMs' long-context performance (Lei et al., 2024). However, this kind of biases have not been systematically studied so far, which highlights the need for thorough investigation.

To bridge the gap, we introduce LONGPIBENCH, a benchmark designed to evaluate positional bias with multiple relevant pieces. It assesses positional bias in two categories: (1) **absolute positions**, referring to the location of relevant information within the entire context, and (2) **relative positions**, referring to the distribution and distance between multiple relevant information pieces. It includes diverse tasks of different complexity and spans four input lengths from 32K to 256K tokens. To the best of our knowledge, LONGPIBENCH is the most comprehensive benchmark for isolating and analyzing positional bias in long text models.

We evaluated nine popular LLMs. Our experimental

[1] anonymous repo link available.

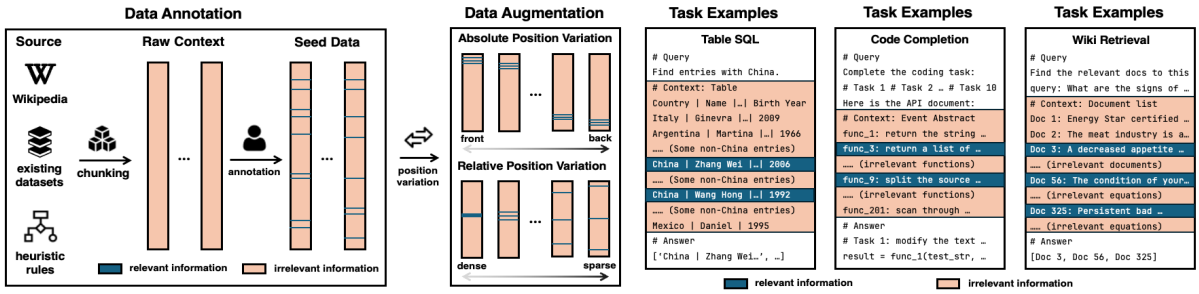

Figure 2: Construction and task examples of LONGPIBENCH. We manually annotated seed data and varied the positions of relevant information for data augmentation.

analysis yields several key findings: (1) most current models demonstrate enhanced robustness against "lost in the middle" phenomenon. (2) However, they show biases related to the spacing of relevant information (i.e.**relative positions**), especially in retrieval tasks. (3) Additionally, we discuss the impact of model size and query-aware contextualization on this issue.

These findings emphasize the importance of evaluating and mitigating positional biases to advance long-context LLM capabilities.

2 LONGPIBENCH

LONGPIBENCH is a dataset designed to evaluate positional bias with multiple relevant information pieces. As shown in Figure 2, we first manually annotated several seed examples and then augmented them by varying the positions of relevant information. More details can be found in Appendix A.

2.1 Core Statistics

LONGPIBENCH contains 3 different tasks, 4 different input length levels[2]: ($32k$, $64k$, $128k$, and $256k$). To analyze the impact of positional bias, we set 16 different absolute and relative location levels respectively. The benchmark is composed of 7,040 instances, each containing around 10 pieces of relevant information. The whole dataset comprises to $845M$ tokens.

2.2 Seed Data Annotation

We manually labeled 15-20 seed data points for three tasks: *Table SQL*, *Code Completion*, and *Wiki Retrieval*, which represent typical use cases in real-world applications of long-context models (Lei et al., 2024; Jimenez et al., 2024; Ajith et al., 2024). Each instance contains 10 relevant pieces of information. This selection was based on an examination of long-context application scenarios, where the number of relevant elements typically falls around the order of magnitude of ten, although it varies across different tasks (Bai et al., 2023; Wang et al., 2024; Dong et al., 2024). Detailed task definitions, examples, and other pertinent details are provided in Appendix A.

[2]measured with GPT2Tokenizer (Radford et al., 2019)

2.3 Data Augmentation

To analyze the positions of relevant information, we augmented the data by altering the absolute and relative positions of the relevant pieces while keeping all other features unchanged.

We broke down the context into elements based on natural information units: table entries for *Table SQL*, API instances for *Code Completion* and documents for *Wiki Retrieval*. We labeled each element as relevant or irrelevant in a reversal way. We select some elements to be relevant, and then form queries around them, and add irrelevant ones to form the context. By introducing varying amounts of irrelevant information, the context lengths are varied at four levels: 32K, 64K, 128K, and 256K. We then shuffled the element positions to introduce positional variations. Notice that changing the order of elements does not compromise the coherence of the context.

Absolute Position. To analyze the impact of absolute position on LLM performance, we manipulated where relevant information appears in the context. Each context was divided into 16 equal segments from start to end. We placed all 10 relevant pieces within a single segment to keep their relative positions consistent. By moving this segment from the first to the last position, we varied the absolute position from the start to the end of the input. The average position of these relevant pieces served as the absolute position metric which is calculated as:

$$\text{Average Location} = \left(\frac{l-1}{N-1}\right) \times L,$$

where l is the current level, N is the total number of levels (16), and L is the length of the context.

This setup allowed us to assess how model performance changes as relevant information is placed further back in the context.

Relative Position. To examine the effect of spacing between relevant information pieces on LLM performance, we created 16 levels of distribution density. Each level represents a different spacing configuration among the 10 relevant pieces. At the densest level, all relevant pieces are adjacent with no irrelevant information between them. At the sparsest level, they are evenly

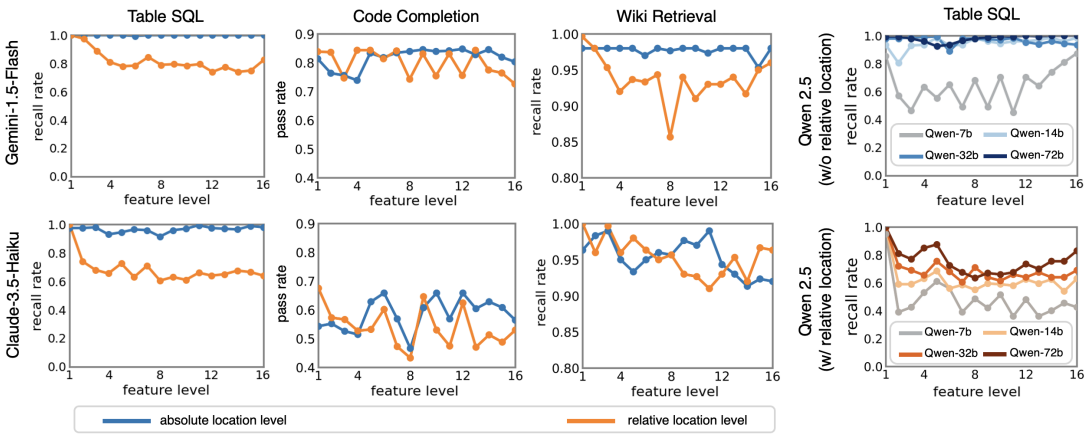

Figure 3: The impact of relevant information's absolute and relative position on Geimini-1.5-Flash (Team et al., 2024), Claude-3.5-Haiku (Anthropic, 2024) and Qwen 2.5 model family (Qwen, 2024). A higher absolute position feature level indicates locations closer to the end of input, while a higher relative position feature level indicates a greater distance between relevant pieces of information.

distributed throughout the context with equal intervals of irrelevant information. Intermediate levels gradually increase spacing from adjacent to evenly spaced. The distance between each relevant piece is calculated as:

$$\text{Distance} = \left(\frac{L}{n-1}\right) \times \left(\frac{l-1}{N-1}\right),$$

where n is the number of relevant pieces (10), l is the current level ranging from 1 to N, N is the total number of levels (16), and L is the length of the context.

To control for absolute position effects, we randomized the starting position of the first relevant piece in each example. This ensures that any observed performance differences are due to relative spacing rather than absolute positions within the context.

3 Experimental Setup

To evaluate the influence of context information positioning on long-text LLMs, we conducted experiments using popular long-context language models.

Models. We assessed a total of nine LLMs, comprising six open-source and three commercial options. The selection of open-source models includes the 70B model from Llama-3.1-Instruct series (Dubey et al., 2024), the 7B, 14B, 32B, 72B models from Qwen-2.5 family (Qwen, 2024), the 8×22B model of WizardLM-2 (Xu et al., 2023). The commerical models we selected are Gemini-1.5-Flash (Team et al., 2024), Claude-3-Haiku (Anthropic, 2024) and GPT-4o-mini (OpenAI, 2024). The selected models are good representatives of popular and top-performance long-context models. Due to computational limitations, we evaluated the open-source model only on the *Table SQL* task.

Metric. For both the *Table SQL* and *Wiki Retrieval* tasks, performance is measured using recall rate. This metric evaluates the proportion of relevant items included in the output. Formally, given a set of reference items $D = \{d_1, \ldots, d_n\}$ and a set of retrieved/generated items $\hat{D}$, the recall rate is:

$$M_{\text{Recall}} = \frac{|D \cap \hat{D}|}{|D|}.$$

In *Table SQL*, D represents target entries, and $\hat{D}$ represents the entry present in the output. In *Wiki Retrieval*, D represents the set of relevant documents, and $\hat{D}$ represents the top 10 documents retrieved by the model.

For the *Code Completion* task, performance is evaluated with the pass rate across 8-12 test cases $T = \{t_1, \ldots, t_m\}$. The pass rate is computed as:

$$M_{\text{Code}} = \frac{1}{|T|} \sum_{j=1}^{|T|} \mathbf{1}[G \text{ passes } t_j].$$

All metrics range from 0.0 to 1.0, where 0.0 means complete failure, and 1.0 means perfect performance.

Context Length. Since 32k tokens is the minimal context length supported by tested LLMs, we standardized the context length to 32k[3] tokens for all experiments.

Detailed discussions on parameter settings and prompt configurations are provided in Appendix B.

4 Results and Discussion

In this section, we analyze the impact of absolute and relative positional bias. And we further analyze these phenomena from two perspectives: the number of parameters and query-aware contextualization. Full Experimental results are available in Appendix C.

4.1 Impact of Absolute Position

As illustrated by the **blue lines** in Figure 3, we progressively shift the interval of relevant information from the beginning to the end.

[3]The minimal context size is 64k, but some tokenizers expand our 64k inputs to nearly 80k, exceeding the limit.

We observe that (1) some open-source models like Qwen 2.5 (7B) (Qwen, 2024) still suffer heavily from the severe "lost in the middle" phenomenon but (2) commercial models and larger open-source models are more robust to the bias of absolute position. Although absolute position still significantly affects the recall rate in the *Code Completion* experiments, this bias becomes less severe in the *Table SQL* and *Wiki Retrieval* tasks.

4.2 Impact of Relative Position

As illustrated by the **orange lines** in Figure 3, we progressively increase the distance between relevant pieces of information.

We observe that both open-source and commercial models exhibit noticable biases toward different relative positions. In the case of *Code Completion*, this bias is prominent. As the relative positions of relevant information pieces shift from being fully adjacent to uniformly distributed across the context, the model's performance fluctuates by 20-30%. For tasks with a stronger retrieval nature, such as Table SQL and *Wiki Retrieval*, the bias even displays certain patterns. Specifically, performance initially declines sharply and then decreases more gradually.

These findings indicate that the relative positioning among multiple relevant pieces of information is a serious and unresolved issue, which may substantially undermine the effectiveness of long-text language models in practical applications.

4.3 Further Analysis

Effect of Parameter Size. When selecting models for evaluation, we included four variants from the Qwen 2.5 Family (Qwen, 2024) with differing parameter sizes. These models exhibit no significant differences in architecture, training methods, or training data. By analyzing their performance under identical positional information features, we can isolate the impact of parameter size on the robustness to positional bias. We use *Table SQL* task, where the pattern is most significant

As illustrated in Figure 3, for absolute position bias, we found that simply increasing the model parameters from 7B to 14B—while keeping architecture, training methods, and data constant substantially mitigates the "lost in the middle" (Liu et al., 2023) issue. This suggests that robustness to absolute positions may be an "emergent ability" (Wei et al., 2022) and increasing the number of parameters can significantly enhances it.

In contrast, regarding biases related to relative positional information, augmenting the number of parameters only yielded minor quantitative improvements and did not alter the pronounced bias trend. This trend remains largely unchanged even in commercial models with approximately hundreds of billions of parameters. These findings indicate that merely increasing parameter size is insufficient to develop robustness to relative positions, and new techniques may be necessary.

Effect of Query-Aware Contextualization. Liu et al. (2023) demonstrated that the placement of the query (beginning or end of the context) significantly affects the performance of decoder-only models due to unidirectional attention. When the query is placed after the context, the LLM cannot attend to the query token while processing the context tokens.

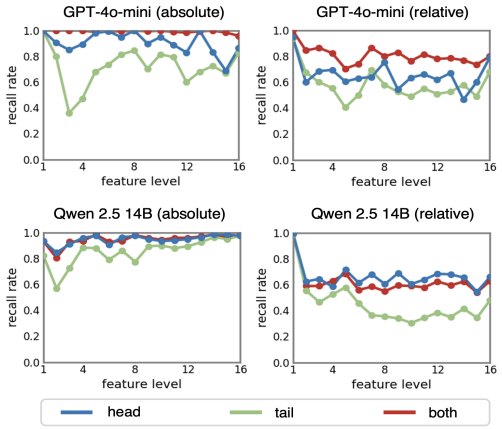

Figure 4: Impact of query placement (beginning, end, both) on the performance of GPT-4o-mini (OpenAI, 2024) and Qwen-2.5-14B (Qwen, 2024) models.

As shown in Figure 4, our experiments with GPT-4o-mini (OpenAI, 2024) and Qwen-2.5-14B (Qwen, 2024) on *Table SQL* corroborate this observation and confirm that it also holds for bias caused by relative position changes. When the query is placed at the end of the context, the model performs much worse than when the query is at the beginning or both at the beginning and end. However, the difference between placing the query only at the beginning and at both the beginning and end depends on the model. This indicates that for decoder-only long-text models, the position of the query is also crucial in influencing biases related to the absolute and relative positions of relevant information.

5 Conclusion

This study investigates a new category of positional bias involving multiple relevant pieces of information in long-context LLMs through three key contributions.

(1) Benchmark Development: We introduce LongPiBench, the most comprehensive benchmark for evaluating positional bias in long-text LLMs, assessing both absolute and relative biases.

(2) Comprehensive Evaluation: Using LongPiBench, we evaluated nine popular LLMs, investigated the "lost in the middle" phenomenon, and identified novel yet significant biases related to the relative positioning of multiple relevant pieces of information.

(3) Findings: Our experiments show that while LLMs have improved robustness against absolute positional biases, they are still sensitive to relative positional biases, especially for retrieval-intensive tasks. We also explore how model size and query-aware contextualization impact these biases.

These findings emphasize the necessity of continuously mitigating positional biases in long-text models.

Limitation

Lack of In-depth Analysis. Our systematic experiments demonstrate that two types of positional bias exist when multiple related pieces of information are present in the context. We also analyzed how these biases relate to the number of parameters and query contextualization. However, we are currently unable to explain the reasons behind these two positional biases.

Focus on Specific Models. The evaluation was conducted on a set of nine popular large language models (LLMs), including both open-source and commercial options. However, the findings are limited to these models. The study does not account for the performance of other emerging or less popular models, which might exhibit different results regarding positional biases.

Ethical Considerations

Human Annotation. Our seed construction process involves manual annotation. This annotation was carried out by some of the authors, who are researchers with substantial knowledge in LLM evaluation. Consent was obtained from the individuals whose data we are using or curating. The data collection protocol was approved.

Data Security. Some data in our Table SQL task may appear to pertain to personal information. However, this data is not actual personal information. Instead, it is generated by us through specific heuristics, eliminating the risk of personal information leakage.

Use of AI assistants We use GPT-4o (OpenAI, 2024) for expression modification and grammar sanity check during the composition process.

Related Works

Many benchmarks have been proposed to evaluate long-context performance of LLMs by designing a variety of tasks with different context length. This field is relatively saturated at present, with some of the representative benchmarks including Long Range Arena(Tay et al., 2021), Scrolls(Shaham et al., 2022), ZeroScrolls(Shaham et al., 2023), Longbench(Bai et al., 2023), L-Eval(An et al., 2023), Longbench(Bai et al., 2023), LV-Eval(Yuan et al., 2024), and ∞Bench(Zhang et al., 2024b).

However, these benchmarks tend to provide only a general conclusion regarding which task types are more challenging, without offering in-depth analysis on positional bias like this paper proposes.

Levy et al. (2024) explored the impact of input length on reasoning performance using a similar data augmentation approach, adding irrelevant elements to context-relevant elements. While their method shares some similarities with ours, our focus is fundamentally different, leading to entirely distinct conclusions. Their study centers on the overall input length which has nothing to do with positional bias. But we investigate the distance between relevant information pieces, where the input length is fixed.

References

Anirudh Ajith, Mengzhou Xia, Alexis Chevalier, Tanya Goyal, Danqi Chen, and Tianyu Gao. 2024. Litsearch: A retrieval benchmark for scientific literature search. *arXiv preprint arXiv:2407.18940*.

Chenxin An, Shansan Gong, Ming Zhong, Mukai Li, Jun Zhang, Lingpeng Kong, and Xipeng Qiu. 2023. L-Eval: Instituting Standardized Evaluation for Long Context Language Models. *arXiv e-prints*, pages arXiv–2307.

Anthropic. 2024. Introducing claude 3.5 sonnet. Accessed: 2024-09-15.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. 2023. Longbench: A Bilingual, Multitask Benchmark for Long Context Understanding. *arXiv preprint arXiv:2308.14508*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. 2023. Longlora: Efficient fine-tuning of long-context large language models. *arXiv preprint arXiv:2309.12307*.

Zican Dong, Tianyi Tang, Junyi Li, Wayne Xin Zhao, and Ji-Rong Wen. 2024. BAMBOO: A comprehensive benchmark for evaluating long text modeling capacities of large language models. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 2086–2099, Torino, Italia. ELRA and ICCL.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Chi Han, Qifan Wang, Hao Peng, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. 2024. LM-infinite: Zero-shot extreme length generalization for large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3991–4008, Mexico City, Mexico. Association for Computational Linguistics.

Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Yuan Yao, Ao Zhang, Liang Zhang, et al. 2021. Pre-trained models: Past, present and future. *Preprint*, arXiv:2106.07139.

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.

Amey Hengle, Prasoon Bajpai, Soham Dan, and Tanmoy Chakraborty. 2024. Multilingual needle in a haystack: Investigating long-context behavior of multilingual large language models. *Preprint*, arXiv:2408.10151.

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. *Preprint*, arXiv:2406.16008.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2024. Swe-bench: Can language models resolve real-world github issues? *Preprint*, arXiv:2310.06770.

Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The NarrativeQA reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328.

Fangyu Lei, Qian Liu, Yiming Huang, Shizhu He, Jun Zhao, and Kang Liu. 2024. S3eval: A synthetic, scalable, systematic evaluation suite for large language models. *Preprint*, arXiv:2310.15147.

Mosh Levy, Alon Jacoby, and Yoav Goldberg. 2024. Same task, more tokens: the impact of input length on the reasoning performance of large language models. *arXiv preprint arXiv:2402.14848*.

Jiaqi Li, Mengmeng Wang, Zilong Zheng, and Muhan Zhang. 2024. LooGLE: Can long-context language models understand long contexts? In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16304–16333, Bangkok, Thailand. Association for Computational Linguistics.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023. Lost in the middle: How language models use long contexts. *Preprint*, arXiv:2307.03172.

Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. 2024. Large language models: A survey. *arXiv preprint arXiv:2402.06196*.

Elliot Nelson, Georgios Kollias, Payel Das, Subhajit Chaudhury, and Soham Dan. 2024. Needle in the haystack for memory based large language models. *Preprint*, arXiv:2407.01437.

OpenAI. 2024. Hello gpt-4o. https://openai.com/index/hello-gpt-4o/. Accessed: 2024-06-08.

Qwen. 2024. Qwen2.5: A party of foundation models.

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

Uri Shaham, Maor Ivgi, Avia Efrat, Jonathan Berant, and Omer Levy. 2023. ZeroSCROLLS: A Zero-Shot Benchmark for Long Text Understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 7977–7989.

Uri Shaham, Elad Segal, Maor Ivgi, Avia Efrat, Ori Yoran, Adi Haviv, Ankit Gupta, Wenhan Xiong, Mor Geva, Jonathan Berant, et al. 2022. SCROLLS: Standardized CompaRison Over Long Language Sequences. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 12007–12021.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. 2021. Long Range Arena: A Benchmark for Efficient Transformers. In *International Conference on Learning Representations*.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *Preprint*, arXiv:2403.05530.

Cunxiang Wang, Ruoxi Ning, Boqi Pan, Tonghui Wu, Qipeng Guo, Cheng Deng, Guangsheng Bao, Xiangkun Hu, Zheng Zhang, Qian Wang, et al. 2024. Novelqa: Benchmarking question answering on documents exceeding 200k tokens. *Preprint*, arXiv:2403.12766.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

Wikipedia. 2024. Wikipedia, The Free Encyclopedia. https://www.wikipedia.org/. [Online; accessed 15-September-2024].

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*.

Tao Yuan, Xuefei Ning, Dong Zhou, Zhijie Yang, Shiyao Li, Minghui Zhuang, Zheyue Tan, Zhuyu Yao, Dahua Lin, Boxun Li, et al. 2024. LV-Eval: A Balanced Long-Context Benchmark with 5 Length Levels Up to 256K. *arXiv preprint arXiv:2402.05136*.

Daoguang Zan, Bei Chen, Zeqi Lin, Bei Guan, Yongji Wang, and Jian-Guang Lou. 2022. When language model meets private library. *arXiv preprint arXiv:2210.17236*.

Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Hao, Xu Han, Zhen Thai, Shuo Wang, Zhiyuan Liu, et al. 2024a. Infinitebench: Extending long context evaluation beyond 100k tokens. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15262–15277.

Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Khai Hao, Xu Han, Zhen Leng Thai, Shuo Wang, Zhiyuan Liu, et al. 2024b. ∞ Bench: Extending Long Context Evaluation Beyond 100K Tokens. *arXiv preprint arXiv:2402.13718*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

A Details of LONGPIBENCH

A.1 Task Definitions

Table SQL This task involves retrieving entries containing specific features from a table with a large number of entries. The prototype of this task is primarily derived from experiments in S3Eval (Lei et al., 2024), specifically those examining information distributions with extreme positional variability.

Code Completion This task involves performing basic programming assignments based on the definitions, signatures, examples, and other information provided in API documentation. The task is considered more challenging than Table SQL tasks because an LLM must not only identify which parts of the API documentation are relevant but also correctly utilize them during coding. The data we use originates from the Private Coding Dataset introduced by Zan et al. (2022). To ensure that the LLM does not rely on internal knowledge about common Python libraries, both the API documentation and task function names have been masked. This privatization process is crucial for evaluating performance on long-text scenarios, as it compels the LLM to extract relevant information directly from the provided context.

Wiki Retrieval This task involves identifying relevant passages from Wikipedia (Wikipedia, 2024) pages based on a given question. It is a common scenario in which LLMs are used to rerank relevant passages retrieved through information retrieval systems (Ajith et al., 2024).

A.2 Task Examples

Here are some examples of the three tasks in LONGPIBENCH. Queries are placed both before and after the context for better query contextualization.

A.2.1 Table SQL

Input You are given a table of entries with the following columns: Country, Name, Birth Year, Birth Month, Blood Type. Your task is to find all the entry with the following Country: China. You should return all the entries that match the query as a python list. For example, ['| China | Hong Liang | 1991 | August | A |', ...]. You should not generate anything else. Here is the table:

Country	Name	Birth Year	...	Blood Type
Italy	Ginevra	2009	February	O
Argentina	Martina	1966	March	B
Egypt	Salma	1985	July	B
...
| China | Zhang Wei | 2006 | November | O |
...
| China | Wang Wei | 1966 | February | AB |
...
| Australia | Emily | 1983 | December | O |
| Italy | Leonardo | 1985 | November | O |

You are given a table of entries with the following columns: Country, Name, Birth Year, Birth Month, Blood Type. Your task is to find all the entry with the following Country: China. You should return all the entries that match the query as a python list. For example, ['| China | Hong Liang | 1991 | August | A |', ...]. You should not generate anything else.

Ground Truth

[
"| China | Zhu Wei | 1992 | September | B |",
"| China | Zhang Wei | 1955 | March | O |",
"| China | Zhang Wei | 2006 | November | O |",
"| China | Wang Wei | 2001 | September | B |",
"| China | Yang Wei | 2016 | November | AB |",
"| China | Li Na | 1974 | January | B |",
"| China | Liu Wei | 1975 | November | O |",
"| China | Gao Wei | 1954 | August | B |",
"| China | Zhu Wei | 1989 | September | AB |",
"| China | Wang Wei | 1966 | February | AB |"
],

A.2.2 Code Completion

Notice that in the *Code Completion* task, the ground truth is provided in its unmasked form, while the LLMs generate code based on the masked API documentation, resulting in masked code as output.

Input Please complete the code snippet above according to the provided code snippet and the api doc.

```
# Text where substitution will
take place
text = 'Thelib_2 alib_2 123 apples
and 456 oranges.'
# Define pattern and replacement
for substitution
sub_pattern = r'
d+' lib_2placement = 'NUM'
# Task 1:  Substitute matching
text using `sub_pattern` and
`lib_2placement`
lib_2sult_1 = print(lib_2sult_1)
# Task 2:  ...
```

The following context is a code snippet with the detailed api doc.

```
{
  "api_path": "lib_2.submodule_26",
  "api_doc": "Returns complex...",
  "api_signature": "",
  "api_parameters": "",
  "api_parameters_number": "=0",
  "api_returns": ""
},
```

...

(more API instances)
Please complete the code snippet above according to the provided code snippet and the api doc.

```
# Text where substitution will
take place
text = 'Thelib_2 alib_2 123 apples
and 456 oranges.'
# Define pattern and replacement
for substitution
sub_pattern = r'
d+' lib_2placement = 'NUM'
# Task 1:  Substitute matching
text using `sub_pattern` and
`lib_2placement`
lib_2sult_1 = print(lib_2sult_1)
# Task 2:  ...
```

Ground Truth

```
import re

text = 'There are 123 apples
and 456 oranges.'
sub_pattern = r'\d+'
replacement = 'NUM'

## task 1
result_1 = re.sub(sub_pattern,
replacement, text)
print(result_1)
...
```

A.2.3 Wiki Retrieval

Input Please find the top-10 most helpful Docs that will help answer the question. (You do not need to answer it.)
What are ten easy eco-friendly practices that individuals can adopt in their daily lives?
Here is the context
Doc 1
Gaetano J̈amesS̈enese (born 6 January 1945) is an Italian saxophonist, composer, and singer-songwriter. Life and career Senese was born in Naples, the son of Anna Senese and James Smith, an American soldier from North Carolina in Italy because of World War II. Senese's father moved back to the US eighteen months after Gaetano's birth and never returned. Senese started playing the saxophone at 12 years old.
Doc 2
He made his professional debut in the 1960s, as a member of the rhythm and blues band The Showmen (later known as Showmen 2), with whom he won the 1968 edition of Cantagiro. In 1974 Senese co-founded and led the critically acclaimed jazz-rock group Napoli Centrale. After the group disbanded in 1978, he started a long collaboration with Pino Daniele, both in studio and on stage. His first solo album was released in 1983 by Polydor Records.
.....
Doc 1128
Release and critical reception Generations in Song was first released on Coldwater Records in 2001. It was originally offered as a compact disc and contained 19 tracks in its original release. On February 10, 2004, the album was re-released on Slewfoot Records in a compact disc format again. However, only 12 tracks were included on the re-release. The album cover was also changed for the re-release of the project.
Please find the top-10 most helpful Docs that will help answer the question. (You do not need to answer it.)
What are ten easy eco-friendly practices that individuals can adopt in their daily lives?
You should output a python list of the Doc Index like "'['Doc 1', ...]"' as your answer

Ground Truth
["Doc 920", "Doc 927", "Doc 935", "Doc 942", "Doc 949", "Doc 957", "Doc 964", "Doc 971", "Doc 979", "Doc 986"]

B Details of Experimental Setup

B.1 Inference Parameters

To ensure consistency and reproducibility in our experiments, we standardized the inference parameters across all models during the inference phase. Specifically, we set the temperature parameter (temp) to 0.1 and the top-p sampling parameter (top_p) to 0.9. This unification of inference settings facilitates the replication of experiments and establishes a consistent evaluation standard across different models.

B.2 Prompt Template

For the three tasks, we used the following prompt templates respectively. Notice that we place queries both before and after the context body for better query contextualization.

B.2.1 Table SQL

Input You are given a table of entries with the following columns: Country, Name, Birth Year, Birth Month, Blood Type. Your task is to find all the entry with the following Country: `{country}`. You should return all the entries that match the query as a python list. For example, ['| China | Hong Liang | 1991 | August | A |', ...]. You should not generate anything else. Here is the table:

> {context}
> You are given a table of entries with the following columns: Country, Name, Birth Year, Birth Month, Blood Type. Your task is to find all the entry with the following Country: {country}. You should return all the entries that match the query as a python list. For example, ['| China | Hong Liang | 1991 | August | A |', ...]. You should not generate anything else.

B.2.2 Code Completion

> **Input** Please complete the code snippet above according to the provided code snippet and the api doc.
> {query}
> The following context is a code snippet with the detailed api doc.
> {context}
> Please complete the code snippet above according to the provided code snippet and the api doc.
> {query}

B.2.3 Wiki Retrieval

> **Input** Please find the top-10 most helpful Docs that will help answer the question. (You do not need to answer it.)
> {query}
> Here is the context
> {context}
> Please find the top-10 most helpful Docs that will help answer the question. (You do not need to answer it.)
> {query}
> You should output a python list of the Doc Index like "'['Doc 1', ...]"' as your answer

C Details of Experimental Results

In the main text, for better readability, we only presented the experimental results of a subset of tested LLMs in the form of line charts. Here we present all the experimental results in both tabular and chart form. This will better facilitate the precise display of the experimental results.

Figure 5 and 6 use line charts to illustrate the performance of all selected closed-source and open-source models across the respective test tasks.

Table 1 and Table 2 summarize the performance of all models on the *Table SQL* task across different absolute and relative positions. Similarly, Table 3 and Table 4 present the results for the *Code Completion* task, while Table 5 and Table 6 correspond to the *Wiki Retrieval* task. Finally, Table 7 and Table 8 show the impact of query contextualization.

Model	Performance / %															
	Lv. 1	Lv. 2	Lv. 3	Lv. 4	Lv. 5	Lv. 6	Lv. 7	Lv. 8	Lv. 9	Lv. 10	Lv. 11	Lv. 12	Lv. 13	Lv. 14	Lv. 15	Lv. 16
Claude	97.5	97.5	98.0	93.0	94.5	96.5	96.0	91.5	96.0	97.0	99.5	97.5	97.0	96.5	99.0	98.0
Deepseek	100.0	99.5	99.5	97.0	98.5	98.5	97.5	99.5	99.0	95.5	97.0	98.0	99.0	99.0	97.5	100.0
Gemini	100.0	100.0	100.0	100.0	100.0	99.5	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
GLM	100.0	99.0	94.0	93.0	91.5	91.0	95.0	96.5	93.5	91.5	92.0	89.0	91.0	91.0	94.0	86.0
GPT	100.0	100.0	100.0	100.0	99.0	99.5	99.5	100.0	99.5	100.0	99.0	98.5	99.5	100.0	98.5	96.0
Llama	96.0	96.0	93.0	96.0	91.0	88.0	92.0	92.0	94.0	94.0	94.0	89.0	94.0	99.0	99.0	98.0
Wizard	85.5	42.5	31.5	20.5	3.0	38.0	36.5	23.0	12.0	13.5	3.5	5.0	1.5	8.5	24.5	90.0
Qwen 7b	85.5	93.5	98.0	99.5	98.5	93.0	98.0	99.5	96.0	70.5	45.0	70.5	64.0	74.0	81.0	87.5
Qwen 14b	93.5	80.5	93.0	93.5	98.5	93.0	93.5	98.0	96.0	94.5	96.0	96.0	97.0	98.5	96.0	98.5
Qwen 32b	98.0	98.0	99.0	99.0	99.5	89.0	98.0	98.0	97.5	97.0	98.5	95.5	93.9	96.0	94.5	93.5
Qwen 72b	99.5	99.5	98.0	96.0	92.5	93.5	96.5	98.0	99.0	99.5	99.5	100.0	99.5	100.0	100.0	100.0

Table 1: Performance of various models across different **absolute position** levels in *Tabel SQL*. The model names are abbreviated for better layout. Full names are listed in Section 3.

Model	Performance / %															
	Lv. 1	Lv. 2	Lv. 3	Lv. 4	Lv. 5	Lv. 6	Lv. 7	Lv. 8	Lv. 9	Lv. 10	Lv. 11	Lv. 12	Lv. 13	Lv. 14	Lv. 15	Lv. 16
Claude	100.0	74.0	68.0	65.5	72.5	63.0	71.0	60.5	63.0	61.0	66.0	64.0	65.0	67.5	66.5	64.0
Deepseek	100.0	78.0	81.0	82.0	82.0	79.5	69.0	81.0	72.0	79.0	70.5	69.0	75.0	78.0	76.0	81.5
Gemini	100.0	97.5	89.0	81.0	78.0	78.5	84.5	79.0	79.5	78.5	79.5	74.0	77.5	74.0	75.0	82.5
GLM	90.0	69.0	68.5	67.5	63.0	58.0	65.0	48.5	62.0	50.5	60.0	57.5	61.5	52.0	51.5	44.0
GPT	100.0	84.5	86.5	82.5	70.5	74.0	86.5	80.0	83.0	76.5	81.5	78.0	78.5	77.0	73.5	80.0
Llama	100.0	77.0	77.0	75.0	88.0	75.0	79.0	74.0	74.0	80.0	68.0	66.0	75.0	72.0	79.0	69.0
Wizard	74.0	28.5	23.5	22.5	47.5	47.0	61.5	51.5	54.0	56.5	65.5	61.0	61.5	60.0	61.0	59.5
Qwen 7b	95.0	39.0	42.5	53.0	61.0	56.0	39.0	48.5	42.0	51.5	36.0	48.0	36.0	40.0	45.5	42.5
Qwen 14b	99.5	59.0	59.0	63.0	68.5	56.0	58.5	55.0	59.5	59.0	58.0	62.5	59.5	62.5	54.0	63.0
Qwen 32b	99.5	72.0	69.0	65.5	75.5	68.0	60.5	71.0	64.0	61.5	66.0	64.0	67.5	64.0	64.0	69.0
Qwen 72b	100.0	81.0	77.0	85.0	87.5	72.5	67.5	63.5	67.0	66.0	67.5	73.5	70.0	75.5	75.5	83.0

Table 2: Performance of various models across different **relative position** levels in *Tabel SQL*. The model names are abbreviated for better layout. Full names are listed in Section 3.

Model	Absolute Performance / %															
	Lv. 1	Lv. 2	Lv. 3	Lv. 4	Lv. 5	Lv. 6	Lv. 7	Lv. 8	Lv. 9	Lv. 10	Lv. 11	Lv. 12	Lv. 13	Lv. 14	Lv. 15	Lv. 16
Claude	54.28	55.19	52.55	51.44	62.81	65.84	56.86	46.59	60.76	65.84	56.86	65.84	60.41	62.81	60.81	56.40
Gemini	81.37	76.37	75.59	73.87	83.34	81.81	83.46	83.95	84.55	83.95	84.15	84.78	82.81	84.55	81.94	80.31
GPT	47.64	57.34	44.99	43.00	44.57	53.65	48.37	50.20	50.78	48.66	47.45	49.87	49.37	54.72	60.61	45.37

Table 3: Performance of various models across different **absolute levels** in *Code Completion*. The data includes **absolute** scores for the Claude, Gemini, and GPT models.

Model	Relative Performance / %															
	Lv. 1	Lv. 2	Lv. 3	Lv. 4	Lv. 5	Lv. 6	Lv. 7	Lv. 8	Lv. 9	Lv. 10	Lv. 11	Lv. 12	Lv. 13	Lv. 14	Lv. 15	Lv. 16
Claude	67.50	57.27	56.64	52.69	53.18	60.16	47.24	43.26	64.58	52.96	47.43	62.41	47.01	51.31	48.79	52.96
Gemini	83.87	83.65	74.71	84.36	84.36	81.37	84.11	74.29	83.09	75.45	83.09	75.59	84.36	77.47	76.45	72.61
GPT	54.57	51.13	56.03	58.11	52.19	48.57	41.59	51.64	68.87	62.38	44.21	42.66	51.66	45.25	66.06	54.83

Table 4: Performance of various models across different **relative levels** in *Code Completion*. The data includes **relative** scores for the Claude, Gemini, and GPT models. Code Completion!

Model	Absolute Performance / %															
	Lv. 1	Lv. 2	Lv. 3	Lv. 4	Lv. 5	Lv. 6	Lv. 7	Lv. 8	Lv. 9	Lv. 10	Lv. 11	Lv. 12	Lv. 13	Lv. 14	Lv. 15	Lv. 16
Claude	96.33	98.33	99.00	95.00	93.33	95.00	96.00	95.67	97.67	97.00	99.00	94.33	93.00	91.33	92.33	92.00
Gemini	98.00	98.00	98.00	98.00	98.00	97.00	98.00	97.67	98.00	98.00	97.33	98.00	98.00	98.00	95.33	98.00
GPT	100.00	99.00	100.00	98.00	98.00	100.00	99.00	100.00	96.00	97.00	98.00	99.00	99.00	98.00	96.00	96.00

Table 5: Performance of various models across different **absolute levels** in *Wiki Retrieval*. The data includes **absolute** scores for the Claude, Gemini, and GPT models.

Model	Relative Performance / %															
	Lv. 1	Lv. 2	Lv. 3	Lv. 4	Lv. 5	Lv. 6	Lv. 7	Lv. 8	Lv. 9	Lv. 10	Lv. 11	Lv. 12	Lv. 13	Lv. 14	Lv. 15	Lv. 16
Claude	100.00	96.00	99.67	96.00	98.00	96.33	95.00	95.67	93.00	92.67	91.00	93.00	95.33	92.00	96.67	96.33
Gemini	99.67	98.00	95.33	92.00	93.67	93.33	94.33	85.67	94.00	91.00	93.00	93.00	94.00	91.67	95.00	96.00
GPT	100.00	98.00	96.00	98.00	100.00	98.00	100.00	95.00	95.00	97.00	96.00	97.00	97.00	100.00	97.00	98.00

Table 6: Performance of various models across different **relative levels** in *Wiki Retrieval*. The data includes **relative** scores for the Claude, Gemini, and GPT models.

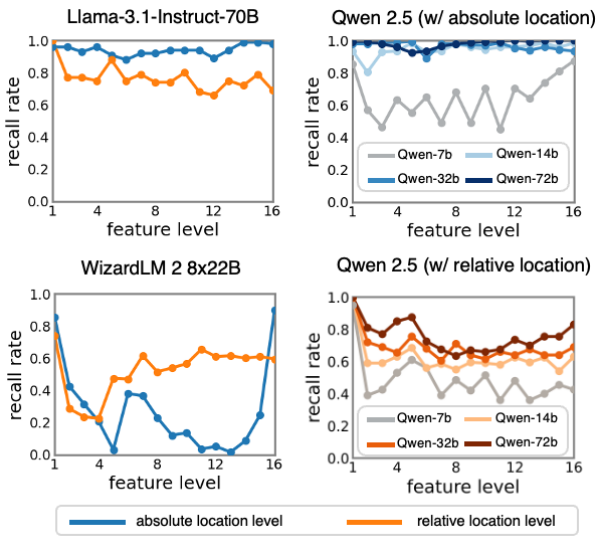

Figure 5: The impact of relevant information's absolute and relative position for all open-source commercial models. A higher absolute position feature level indicates locations closer to the end of input, while a higher relative position feature level indicates a greater distance between relevant pieces of information.

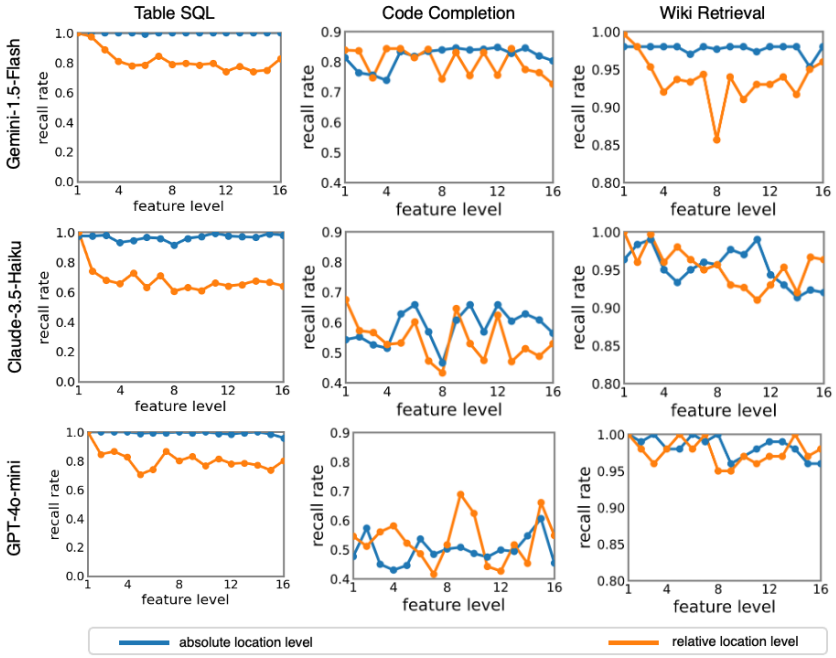

Figure 6: The impact of relevant information's absolute and relative position for all tested commercial models. A higher absolute position feature level indicates locations closer to the end of input, while a higher relative position feature level indicates a greater distance between relevant pieces of information.

Model	Query Position	Performance / %															
		Lv. 1	Lv. 2	Lv. 3	Lv. 4	Lv. 5	Lv. 6	Lv. 7	Lv. 8	Lv. 9	Lv. 10	Lv. 11	Lv. 12	Lv. 13	Lv. 14	Lv. 15	Lv. 16
GPT	Head	100.0	90.5	85.0	89.5	98.0	99.5	95.0	100.0	90.0	95.0	89.0	83.0	100.0	83.5	69.0	86.5
	Tail	100.0	80.0	36.0	47.0	68.0	73.5	81.5	84.5	70.5	81.5	79.5	60.0	68.0	72.5	67.0	83.0
	Both	100.0	100.0	100.0	100.0	99.0	99.5	99.5	100.0	99.5	100.0	99.0	98.5	99.5	100.0	98.5	96.0
Qwen 14B	Head	93.5	84.5	91.0	96.0	97.5	90.5	96.5	97.5	95.0	93.5	94.0	95.0	96.5	98.5	98.5	97.5
	Tail	82.5	57.0	72.5	88.5	88.0	79.0	86.0	77.5	89.5	90.0	88.0	89.5	92.5	96.5	95.0	97.5
	Both	93.5	80.5	93.0	93.5	98.5	93.0	93.5	98.0	96.0	94.5	96.0	96.0	97.0	98.5	96.0	98.5

Table 7: Performance of GPT-4o-mini (OpenAI, 2024) and Qwen-2.5 14B (Qwen, 2024) across different **absolute position** levels with varying placement of the query. The query position can be at the head, tail, or both positions in the input.

Model	Query Position	Performance / %															
		Lv. 1	Lv. 2	Lv. 3	Lv. 4	Lv. 5	Lv. 6	Lv. 7	Lv. 8	Lv. 9	Lv. 10	Lv. 11	Lv. 12	Lv. 13	Lv. 14	Lv. 15	Lv. 16
GPT	Head	95.0	60.0	68.5	69.5	60.5	63.0	64.0	75.5	54.5	63.5	66.0	62.0	67.0	46.5	60.0	79.0
	Tail	94.0	67.5	60.0	55.5	40.5	50.0	69.5	58.0	52.5	49.0	55.0	51.0	52.5	58.0	49.0	68.0
	Both	100.0	84.5	86.5	82.5	70.5	74.0	86.5	80.0	83.0	76.5	81.5	78.0	78.5	77.0	73.5	80.0
Qwen 14b	Head	95.0	60.0	68.5	69.5	60.5	63.0	64.0	75.5	54.5	63.5	66.0	62.0	67.0	46.5	60.0	79.0
	Tail	94.0	67.5	60.0	55.5	40.5	50.0	69.5	58.0	52.5	49.0	55.0	51.0	52.5	58.0	49.0	68.0
	Both	100.0	84.5	86.5	82.5	70.5	74.0	86.5	80.0	83.0	76.5	81.5	78.0	78.5	77.0	73.5	80.0

Table 8: Performance of GPT and Qwen 14b across different **relative levels** with varying placement of the query. The query position can be at the head, tail, or both positions in the input.