∞BENCH: Extending Long Context Evaluation Beyond 100K Tokens

Anonymous ACL submission

Abstract

Processing and reasoning over long contexts is crucial for many practical applications of Large Language Models (LLMs), such as document comprehension and agent construction. Despite recent strides in making LLMs process contexts with more than 100K tokens, there is currently a lack of a standardized benchmark to evaluate this long-context capability. Existing public benchmarks typically focus on contexts around 10K tokens, limiting the assessment and comparison of LLMs in processing longer contexts. In this paper, we propose ∞BENCH, the first LLM benchmark featuring an average data length surpassing 100K tokens. ∞BENCH comprises synthetic and realistic tasks spanning diverse domains, presented in both English and Chinese. The tasks in ∞BENCH are designed to require well understanding of long dependencies in contexts, and make simply retrieving a limited number of passages from contexts not sufficient for these tasks. In our experiments, based on ∞BENCH, we evaluate the state-of-the-art proprietary and open-source LLMs tailored for processing long contexts. The results indicate that existing long context LLMs still require significant advancements to effectively process 100K+ context. We further present three intriguing analyses regarding the behavior of LLMs processing long context.

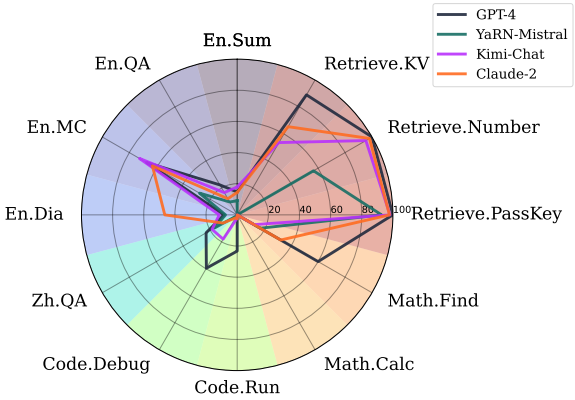

Figure 1: The performance of GPT-4, Kimi-Chat, YaRN-Mistral, and Claude 2 on ∞BENCH. A higher value represents better performance.

1 Introduction

In recent years, large language models (LLMs) (Brown et al., 2020; OpenAI, 2023a; Touvron et al., 2023) have exhibited exceptional performance across a range of natural language processing (NLP) tasks (Qiu et al., 2020; Han et al., 2021). LLMs are showing a promising direction toward generalist task assistance, being capable of aiding users in practical tasks through conversational interactions. These tasks include web navigation (Nakano et al., 2021), analysis of code repositories (Chen et al., 2021), and extraction of useful information from documents (Kočiskỳ et al., 2018), indicating a step towards artificial general intelligence. For these LLM-based scenarios, the ability to process long contexts is increasingly critical, in addition to understanding fine-grained semantics and possessing extensive knowledge (Dong et al., 2023; Huang et al., 2023). Textual documents, historical dialogues, complex instructions, and cumbersome workflows, which constitute the data most directly processed in daily tasks, must be input to LLMs as long contexts for effective processing.

Despite this growing importance, LLMs consistently face challenges in processing long contexts, primarily due to the substantial computational resources required for long sequence training (Dao et al., 2022; Dao, 2023) as well as the apparent inability to generalize to sequences longer than those encountered during training (Chen et al., 2023a; Peng et al., 2023b). LLMs are typically trained on sequences containing no more than 8K tokens (Touvron et al., 2023; Penedo et al., 2023; Biderman et al., 2023), and thus cannot well handle contexts exceeding 8K tokens. These limitations have largely restricted most LLMs from being applied to more complex tasks.

Benchmark	Avg Len	En	Zh	Code	Math	Novel	Dialogue	Synthetic
LRA (Tay et al., 2020)	∼10K	✓	✗	✗	✓	✗	✗	✓
LongBench (Bai et al., 2023)	∼10K	✓	✓	✓	✗	✓	✓	✓
L-Eval (An et al., 2023)	4K - 60K	✓	✗	✓	✓	✗	✗	✓
LooGLE (Li et al., 2023)	∼20K	✓	✗	✗	✗	✗	✓	✗
∞BENCH (ours)	∼200K	✓	✓	✓	✓	✓	✓	✓

Table 1: Comparison to existing long-context benchmarks and ∞BENCH. "En" and "Zh" refer to English and Chinese tasks. "Code", "Math", "Novel", "Dialogue" indicate whether the domain includes tasks from those domains, and "Synthetic" indicates whether there are auto-generated tasks.

Recent advancements in training infrastructure (Shoeybi et al., 2019; Narayanan et al., 2021; Dao et al., 2022; Dao, 2023), and efforts to improve length generalization (Anil et al., 2022; Chen et al., 2023b; Peng et al., 2023b)[1] have led to rapid developments in long-context LLMs. Based on these improved training infrastructures and length generalization methods, several LLMs have purportedly managed to process data exceeding 100K tokens (Peng et al., 2023b; OpenAI, 2023b; 01.AI, 2023b,a), with Claude 2 (Anthropic, 2023) and Kimi-Chat (AI, 2023) even claiming to be able to process up to 200K tokens. However, the rapid emergence of long-context LLMs has outpaced the development of adequate evaluation benchmarks. Present long-context benchmarks predominantly feature contexts averaging around 10K tokens (Bai et al., 2023; Tay et al., 2020), invariably falling below 100K tokens. **This lag in the advancement of long-context evaluation methodologies impedes both the comparative analysis of diverse long-context LLMs and the pinpointing of potential enhancements in long-context processing**.

In this work, we present ∞BENCH , the first comprehensive benchmark featuring an average data length surpassing 100K tokens. ∞BENCH includes tasks in different domains (novels, code, math, etc.) and languages (English and Chinese). To fully evaluate the performance of long-context LLMs, ∞BENCH integrates synthetic tasks that can be auto-generated for even longer contexts (e.g., finding the top-k number in an array) in addition to a set of realistic tasks.

To construct tasks annotated by humans, we develop 5 annotation pipelines for detailed example annotation. These pipelines undergo iterative refinement until the examples meet quality standards. Auto-generated tasks, conversely, can be easily scaled to various lengths. Upon completing ∞BENCH, we assess the performance of several state-of-the-art (SOTA) long-context LLMs on this benchmark to gauge its difficulty and evaluate the effectiveness of these models. The results show that current SOTA LLMs are not fully equipped to handle all tasks within ∞BENCH, highlighting the ongoing challenge of enabling LLMs to process long contexts effectively. We also conduct intriguing analyses on the behavior of LLMs on such long contexts, including the task length ablation, the absent of "lost in the middle phenomenon (Liu et al., 2023)", and the context recalling prompting techniques.

Our contributions can be summarized as follows:

- We construct and release ∞BENCH, the first multi-domain bilingual benchmark for evaluating the ability to understand and reason over contexts surpassing 100K tokens.
- We evaluate SOTA long-context LLMs on ∞BENCH, which reveals severe performance degradation of these LLMs when scaling context lengths. These experimental results and analysis also indicate promising directions to improve long-context LLMs.

2 Related Work

Extending Context Length Transformers, typically trained on text sequences under 8K tokens due to self-attention's quadratic complexity, face challenges in longer downstream tasks. To address this, two main strategies have emerged: firstly, the development of positional encodings capable of handling longer text sequences (Sun et al., 2022; Press et al., 2021), and secondly, the refinement of inference stage techniques to extend current LLMs post-training. The primary approach involves modifying rotary positional encoding (Su et al., 2023) and implementing post-training adjustments to better manage the increased relative positional distances in longer sequences (Zhu et al., 2023; Peng

[1] https://www.reddit.com/r/LocalLLaMA/comments/14lz7j5/ntkaware_scaled_rope_allows_llama_models_to_have/

et al., 2023b; Chen et al., 2023a).

100K+ LLMs Recently, many LLMs have shown the ability to handle over 100K tokens. Some popular proprietary 100K+ LLMs include GPT-4, Claude 2 (Anthropic, 2023), and Kimi-Chat (AI, 2023). On the other hand, there are much fewer open-source 100K+ models. Some notable models include YaRN (Peng et al., 2023b) and Yi-200K (01.AI, 2023a,b). In this paper, we benchmark GPT-4, Claude 2, Kimi-Chat, and YaRN-Mistral-7B-128K[2] on ∞BENCH, which are some of the latest and strongest LLMs that claim to be able to handle over 100K tokens.

Inference Infrastructure Numerous studies aim to accelerate self-attention computation. Research primarily concentrates on refining attention mechanisms through improved IO management (Dao et al., 2022; Dao, 2023), memory optimization (Kwon et al., 2023; Shazeer, 2019; Ainslie et al., 2023), and enhanced parallelization in decoding (Dao et al., 2023; Hong et al., 2023). Approaches like Sliding Window Attention (Beltagy et al., 2020), LM-Infinite (Han et al., 2023), and StreamingLLM (Xiao et al., 2023) introduce attention variants for handling infinitely long sequences without overwhelming computation or memory overhead. However, these techniques often face challenges in maintaining historical information.

Long Context Benchmarks Several benchmarks exist for evaluating long-context AI models, notably featuring context lengths of around 10K tokens. L-Eval (An et al., 2023) and LongBench (Bai et al., 2023) are prominent examples, aggregating pre-existing tasks(Kociský et al., 2017; Dasigi et al., 2021; Yang et al., 2018; Huang et al., 2021; Joshi et al., 2017) into comprehensive benchmarks. LongBench encompasses four categories—QA, summarization, synthetic retrieval, and code—spanning 21 tasks, with four being novel. Conversely, L-Eval incorporates 18 tasks across QA, summarization, math, retrieval, and multiple-choice (MC) domains, introducing three new tasks. Another notable benchmark, LooGLE (Li et al., 2023), differentiates between short and long dependency examples, focusing on summary and QA tasks; its summary corpus contrasts with ours, utilizing academic papers over novels. The Long-Range Arena (LRA) (Tay et al., 2020) further diversifies with six tasks in text, image, and math, designed for scalability. In comparison, ∞BENCH stands out for its substantially longer contexts and a broader range of task domains. Table 1 offers a detailed comparison of these long-context benchmarks.

[2] https://huggingface.co/NousResearch/Yarn-Mistral-7b-128k, we denote this model by YaRN-Mistral.

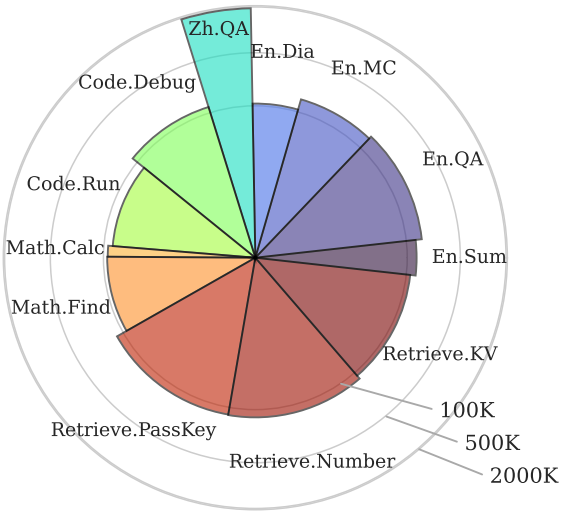

Figure 2: The statistics of the data in ∞BENCH. The radius of each segment indicates the length of input plus output on the logarithmic scale, and the width (or angle) indicates the number of examples (proportionally to the total number of examples).

Task	Annotation	# Ex.	Avg Len
Ret.PassKey	Auto	590	122.4K/2
Ret.Number	Auto	590	122.4K/4
Ret.KV	Auto	500	121.1K/22.7
En.Sum	Human	103	103.5K/1.1K
En.QA	Human	351	192.6k/4.8
En.MC	Human	229	184.4K/5.3
Zh.QA	Human	189	2068.6K/6.3
En.Dia	Auto	200	103.6K/3.4
Code.Debug	Human	394	114.7K/4.8
Code.Run	Auto	400	75.2K/1.3
Math.Calc	Auto	50	43.9K/43.9K
Math.Find	Auto	350	87.9K/1.3

Table 2: Data statistics. The columns indicate whether the annotation was auto-generated or done by humans, the number of examples, and the average length (input/output) in tokens.

3 ∞BENCH

∞BENCH encompasses 12 tasks spanning 5 domains: retrieval, code, math, novels, and dialogue. Two of these tasks are derived from existing literature(Mohtashami and Jaggi, 2023; Liu et al.,

Task	GPT-4	YaRN-Mistral	Kimi-Chat	Claude 2
Retrieve.PassKey	**100.00**	92.71	98.14	97.80
Retrieve.Number	**100.00**	56.61	95.42	98.14
Retrieve.KV	**89.00**	0.00	53.60	65.40
En.Sum	14.73	9.09	**17.93**	14.45
En.QA	**22.22**	9.55	16.52	11.97
En.MC	67.25	27.95	**72.49**	62.88
En.Dia	8.50	7.50	11.50	**46.50**
Zh.QA	**23.06**	16.98	18.62	10.53
Code.Debug	**39.59**	0.76	18.02	2.28
Code.Run	**23.25**	1.25	2.00	2.50
Math.Calc	**0.01**	0.00	0.00	0.00
Math.Find	**60.00**	17.14	12.57	32.29
Average	**45.63**	19.96	34.73	37.06

Table 3: Main results. The performance of the baselines in ∞BENCH. For multiple-choice questions, if the model does not output one of the options, we regard it as an empty prediction, and thus give it a score of 0.

2023). Among the newly introduced tasks, half are generated automatically, while the remainder are annotated by humans.

In total, ∞BENCH includes 3946 examples, featuring a length beyond 100K tokens (average approximately 200K). Figure 2 illustrates the distribution of these tasks. Table 2 details their respective input and output lengths as well as the number of examples per task.

Next, we illustrate each task in detail. The tasks can be grouped into two broad categories. The first involves realistic context collected from real-world scenarios which has potential practical usage of long context LLMs. The second depends on synthetic contexts which are created or collected for testing certain capabilities of long-context LLMs.

3.1 Realistic Context

3.1.1 Novel

We develop novel-based tasks as outlined in Figure 3, utilizing novels sourced from websites[3][4] and are manually filtered. More annotation information in Appendix. C.

In these tasks, models are tasked with reasoning over entire novels presented during inference. Recognizing that many novels, along with their movie adaptations and related discussions, are accessible online and may have been encountered by LLMs during training, we adopt *key entity replacement* as a countermeasure. This involves substituting

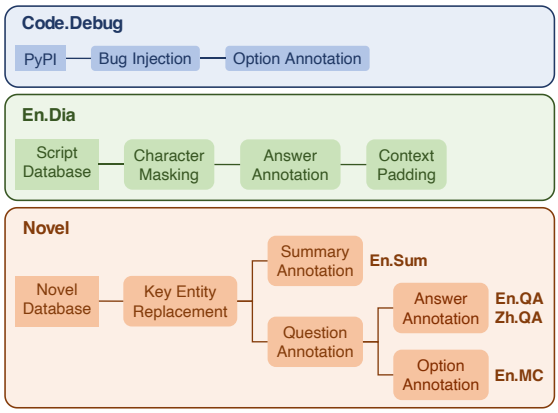

Figure 3: The annotation pipelines for the human-annotated tasks in ∞BENCH.

prominent entities determined by annotators, such as main character names, with unrelated ones, creating "fake novels".

Using these altered novels, we design tasks in three formats: summarization, open-form question answering (QA), and multiple-choice (MC) questions, applying key entity replacement to the annotations as well. All English tasks share the same set of modified novels.

En.Sum The En.Sum task requires models to generate a concise summary of the novel. Gold standard labels are sourced from the web and undergo manual filtering to remove non-summarization content, like comments. Model performance is evaluated using the ROUGE-L-Sum metric (Lin, 2004).

[3] https://www.sparknotes.com/
[4] https://www.cliffsnotes.com/

En.QA & Zh.QA We employ the same annotation pipeline for both En.QA and Zh.QA tasks, ensuring that the questions necessitate long-range dependency and reasoning, beyond simple short passage retrieval. The tasks are primarily categorized into two types of reasoning:

- Aggregation: This involves compiling various pieces of information scattered throughout the novel. An example question in ∞BENCH is "How much money in total did A spend on lunch?"

- Filtering: This requires identifying specific information from a larger set. An example question in ∞BENCH is "What color dress did A wear when A met B for the second time?"

These tasks test LLMs to locate and process information within the novel, performing reasoning through aggregation or filtering to derive answers.

En.MC The En.MC task is annotated similarly to En.QA, but differs in that the model is presented with four answer choices. Annotators are instructed to craft these options to be challenging.

3.1.2 Dialogue

En.Dia The construction process for the En.Dia task is depicted in Figure 3. We gather movie and drama scripts from a designated online database[5], focusing on a corpus of long, multi-role dialogues. Only the English scripts are retained and necessary cleaning is applied.

In the En.Dia task, random instances of character names within a script are replaced with `$$MASK$$`. The objective is to correctly identify these masked names. For scripts falling short of 100K tokens, we augment them by padding with additional scripts.

3.1.3 Code

Code.Debug We develop the task as per the process illustrated in Figure 3. Code repositories, sourced from PyPI[6], undergo a filtering process, and those outside the 64K to 256K token range are excluded (tokenization via the tiktoken tokenizer(OpenAI, 2023c)). Each repository is transformed into a single file, aggregating the content from all files within, each prefaced by its relative path to the root directory. Three of the authors then insert a deliberate and obvious error into one function per repository. The options are presented in the `Class.Function` format. Six methods are employed for bug insertion: (1) deleting a necessary variable declaration; (2) using an incorrect number of arguments in function calls; (3) creating infinite loops; (4) causing indentation errors; (5) substituting references with undefined variable/function names; (6) introducing blatant syntax errors (e.g., non-closed brackets).

Initial results indicate that this task is too challenging for current LLMs (None of the baseline models can identify the most obvious error such as non-closed brackets). To mitigate this, we offer four answer choices, one containing the injected bug and the others are bug-free. Note that this makes many examples easily solved by external retrieval preprocess. However, we encourage the users not to use external retrieval preprocess to keep the evaluation fair. And we are looking forward to the stage where LLMs can directly solve the problem without selection options.

3.2 Synthetic Context

The second category of tasks is characterized by a synthetic context. These tasks, devoid of direct real-world application or use case, are engineered to evaluate the capability for processing lengthy contexts. We delineate four essential ability for effective long-context processing:

1. Location and retrieval. This encompasses all retrieval tasks.
2. Elevated information resolution. This involves the Retrieve.Number task.
3. State preservation. This incorporates the Code.Run and Math.Find functions.
4. Sequential processing. This utilizes the Math.Calc function.

3.2.1 Retrieve

In retrieval tasks, models retrieve specific character sequences from lengthy contexts with predominantly irrelevant content. Such tests, adaptable for any context length, can assess the impact of information placement on model performance, like the *lost-in-the-middle* phenomenon (Liu et al., 2023). The three retrieval tasks in ∞BENCH vary in complexity.

Retrieve.PassKey This task is first proposed by Mohtashami and Jaggi (2023). Models are prompted to find a specific `<key>` called pass key,

[5] https://imsdb.com/
[6] https://pypi.org/

which is a random 5-digit sequence. The pass key is inserted into a lengthy and noisy context, as shown below. In ∞BENCH, we generate examples with 59 different pass key locations that are evenly distributed in the context. At each location, we construct 10 examples with different pass keys. This results in 590 examples.

> There is an important pass key hidden in a lot of irrelevant text. Find it.
> <very long noisy context>
> The pass key is <key>. Remember it. The pass key is <key>
> <very long noisy context>
> What is the pass key?

Retrieve.Number To examine the local attention of LLMs, we have enhanced the complexity of Retrieve.PassKey by increasing the answer length to 10 digits and *incorporating successive repetitive digits*. For example, a `<key>` in Retrieve.PassKey valued `98762`, while in Retrieve.Number is `9998877762`. This modification aims to assess the local resolution capabilities of long context models, as our preliminary experiments indicate that LLMs struggle with discerning repeated numbers.

Retrieve.KV Liu et al. (2023) introduce a key-value retrieval task within a large JSON object containing many key-value pairs (e.g., `30eea139-b6dd-43fc-bc5d-0d3d17980229` → `bfd36c2b-c57e-41ef-9cc1-b21b4e60e664`). This task demands the model to accurately identify and retrieve the value corresponding to a specified key. The complexity of this task is heightened due to the indistinguishable format of relevant and irrelevant information.

3.2.2 Code

Code.Run In this task, we evaluate the ability of LLMs to simulate multi-step function executions that involve basic arithmetic operations. While this task is readily solvable using a Python interpreter, the focus here is on the long-term state tracking required in such tasks. The capability of state tracking has been demonstrated in GPT-4 (Bubeck et al., 2023). Specifically, the task involves creating Python code consisting of multiple simple functions, incorporating operations such as addition, subtraction, and nested function calls. The structural design of these tasks is as follows:

```
def func_0(x):
    return func_1(x) + 4
def func_1(x):
    return x - 1
```

Some functions' return values are dependent on other functions (e.g., `func_0` invokes `func_1`). We define *depth* as the number of cascading function calls initiated by a single call. Thus, the depth for `func_1`'s call within `func_0` is 1. In Code.Run, we employ depths ranging from 2 to 10, ensuring each function calls at most one other function. To keep the simplicity of each single step of computation, these functions are restricted to performing only addition and subtraction.

3.2.3 Math

Math.Find Math.Find assesses the model's capability to identify specific elements within a large array, requiring comprehensive observation for accuracy. This task also tests the ability to preserve states while encoding the context. Concretely, the model receives an extensive list of numbers and is tasked with locating one of seven key numbers: the three largest (1st, 2nd, and 3rd), the three smallest (1st, 2nd, and 3rd), and the median.

Math.Calc To assess sequential processing skills, Math.Calc prompts the model to compute the result of a lengthy arithmetic expression featuring addition and subtraction. Initial experiments indicate that current LLMs struggle to directly produce the final answer. Hence, we instead query the LLMs to provide the intermediate result following each operator. Model performance is evaluated based on the number of correct values preceding the first error.

4 Experiments

We conduct a thorough set of experiments on ∞BENCH. We will introduce the baselines, experimental setup, and main results in this section.

4.1 Baselines

∞BENCH generally requires the ability to handle input contexts longer than 100k. There is a handful of LLMs that claim to be capable of handling contexts over 100k. We include four baselines. The first three are proprietary LLMs as we do not have access to the model, while the last baseline is open-sourced. Details on evaluation are in Appendix. D.

GPT-4 GPT by OpenAI is one of the most widely used and capable LLMs in the market, and a recent

version of GPT-4 (OpenAI, 2023b) can support 128K contexts.

Claude 2 Claude 2 (Anthropic, 2023) is a proprietary chat-based LLM released by Anthropic AI and has shown impressive capabilities. The second version of the Claude series supports 200K contexts. We manually enter each example through the webpage because we have no access to their API.

Kimi-Chat Kimi-Chat, a proprietary chat-oriented LLM developed by Moonshot AI (AI, 2023), is designed to process contexts up to 200K. Due to the lack of API access, we manually input the test data using their web interface.

YaRN-Mistral YaRN-Mistral is a derivative of Mistral-7B (Jiang et al., 2023) introduced by Peng et al. (2023b). The original Mistral-7B was trained on input lengths up to 8K and shows a reduced performance in longer contexts. Peng et al. (2023b) adapted it to 128K contexts by modifying the position encoding and continued training.

4.2 Experimental Setup

Prompt Templates For each model-task combination, we craft prompts to optimize model performance on short dummy examples. Detailed prompt templates for each model and task can be found in Appendix B.

Input Truncation All API-based baselines are subject to a maximum input length limit and will reject inputs exceeding this threshold. While YaRN-Mistral is theoretically capable of handling longer contexts, the authors only claim abilities up to 128K. Therefore, inputs are truncated by removing the center and joining both ends. This approach is predicated on the assumption that key information, such as instructions and book titles, is typically located at either the start or the end of a prompt.

4.3 Main Result

Table 3 and Figure 1 display the performances of various baselines on ∞BENCH. Notably, GPT-4 outperforms other baselines in the retrieval, code, and math domains, with a considerably higher average score. However, in the novel-based tasks, no distinct winner emerges among the proprietary LLMs. On the other hand, the open-source YaRN-Mistral lags behind the proprietary models in most tasks, exhibiting almost random performance in multiple areas. This aligns with its relatively inferior performance in shorter contexts compared to these models. Additionally, it is observed that the baselines generally excel more in retrieval tasks than in other areas, echoing the relative simplicity of these tasks for human participants.

5 Analysis

We subsequently perform a detailed analysis of the results, identifying and emphasizing several notable and interesting phenomena.

5.1 Length Ablation

In line with our benchmark's goal to assess proficiency in managing lengthy contexts, we verify the baselines' capability with shortened context versions. A subset of the auto-generated tasks is modified accordingly, and the performance outcomes are illustrated in Figure 4. It is observed that model performance generally declines with longer input lengths compared to shorter ones. This suggests that while these baselines are technically equipped to handle extended inputs, their effectiveness diminishes significantly under such conditions.

5.2 Lost in the middle

Prior research indicates a performance decline in some LLMs when answers are positioned around the center of the context (Liu et al., 2023). However, our findings do not strongly corroborate this. As depicted in Figure 5, we analyze model performance based on answer location in three location-dependent tasks. We observe *no consistent trend between performance and answer position across different models*. For instance, GPT-4 shows a preference for early answers in Retrieval.KV but favors later ones in En.Dia. In contrast, Claude 2's performance remains relatively unaffected by answer position on all three tasks, whereas YaRN-Mistral and Kimi-Chat excel with end-positioned answers (except that YaRN-Mistral get zero performance on all positions on Retrieval.KV).

One plausible reason why we have different observations from Liu et al. (2023) is that they experiment with different models using at most 16K length contexts, which is about 8 times shorter than our setting. The models in their study are also different from ours. Finally, the tasks are different: their experiments involve document question answering (and their result with Retrieval.KV arguably does not show a very pronounced performance drop as well). We hypothesize that the phenomenon of "Lost in the middle" is only exhibited

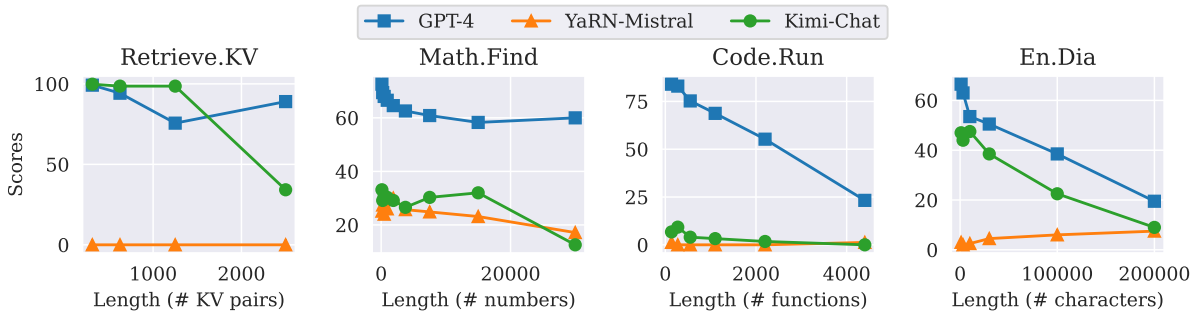

Figure 4: Baseline performance as a function of input length.

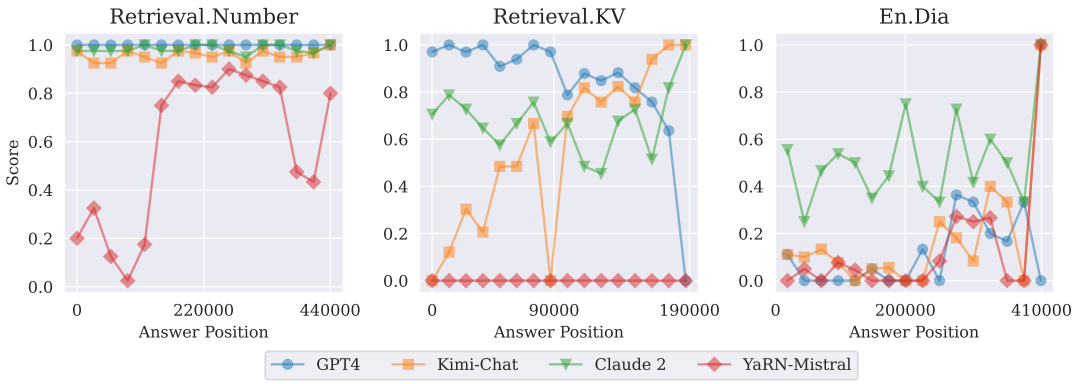

Figure 5: Performance as a function of the answer position (in the number of characters). The steep drop in performance for Kimi-Chat in the middle on Retrieval.KV is caused by the answer being removed by truncation.

> One function in this repo is deliberately made to include an obvious error. Find it.
> <code>
> Think step by step and at last give me your answer for the function with the deliberate error.
> <list of options>

> One function in this repo is deliberately made to include an obvious error. Find it.
> <code>
> Locate the functions in the options, repeat their content, inspect through code, and at last give me your answer for the function with the deliberate error.
> <list of options>

Figure 6: Compared to the first prompt, the second prompt improves GPT-4's results on Code.Debug dramatically.

on specific tasks and models. A more thorough investigation of these differences is beyond the scope of this paper.

5.3 Context Recalling

We identify an intriguing prompting technique for tasks involving extended context, termed *context recalling*. This technique posits that, although the information is present in the context and accessible via direct attention, it may be more effective to first prompt the model to *recall the relevant information in its generation before engaging in further reasoning*. In our experiments using Code.Debug, when we merely instructed GPT-4 to process information step-by-step, the accuracy was **15.74%**. However, by explicitly directing GPT-4 to repeat the relevant code before analysis, its accuracy on Code.Debug markedly improved to **39.59%**. This approach of context recalling warrants additional investigation.

6 Conclusions

We introduce ∞BENCH, the first benchmark tailored for long contexts exceeding 100K in average length. Empirical evidence indicates that despite claims of proficiency with such extensive contexts, current LLMs demonstrate significant performance degradation when dealing with them. This finding highlights the need for advanced methodologies to improve LLMs' efficiency in processing long context. Additionally, our analysis offers insights into LLM behavior in long-context tasks, guiding future research.

Limitations

While our benchmark offers valuable insights into LLM performance, it may not be sufficiently diverse or extensive to provide a comprehensive assessment of model capabilities, a constraint common to most benchmarks. Additionally, the reliance on exact match for scoring, dependent on prompt templates and answer parsing methods, may necessitate tailored redesigns for new model evaluations.

Furthermore, supporting contexts up to 100K tokens may fall short for applications requiring analysis of extensive datasets, such as multiple books or entire databases. Exploring LLMs' capacity to handle up to a million tokens or more presents a promising research avenue. In practical applications, finetuning models to memorize context, rather than processing it during inference, could offer a more efficient alternative, albeit with significant computational demands.

Ethics Statement

Our human annotators are directed to exclude data that may raise sensitive ethical issues, such as offensive language or social biases. Nonetheless, the potential for encountering sensitive content persists, particularly if the sourced books or code contain such material. This concern is somewhat mitigated since the benchmark's primary focus is on evaluating the long-context capabilities of LLMs, rather than influencing their social bias.

The goal of this research is to advance the development of LLMs' proficiency in handling extensive contexts. This could aid in implementing more effective "guardrails" against misuse by incorporating detailed specifications prior to user interactions. However, this approach also potentially increases the risk of novel prompt injection attacks.

References

01.AI. 2023a. Yi-34b-200k. https://huggingface.co/01-ai/Yi-34B-200K.

01.AI. 2023b. Yi-6b-200k. https://huggingface.co/01-ai/Yi-6B-200K.

Moonshot AI. 2023. Kimi chat. https://kimi.moonshot.cn/.

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebr'on, and Sumit K. Sanghai. 2023. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *ArXiv*, abs/2305.13245.

Chen 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*, abs/2307.11088.

Cem Anil, Yuhuai Wu, Anders Andreassen, Aitor Lewkowycz, Vedant Misra, Vinay Ramasesh, Ambrose Slone, Guy Gur-Ari, Ethan Dyer, and Behnam Neyshabur. 2022. Exploring length generalization in large language models. *Advances in Neural Information Processing Systems*, 35:38546–38556.

Anthropic. 2023. Model card and evaluations for claude models.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2023. Longbench: A bilingual, multitask benchmark for long context understanding.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *CoRR*, abs/2005.14165.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar,

Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.

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. *arXiv preprint arXiv:2107.03374*.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. 2023a. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. 2023b. Extending context window of large language models via positional interpolation. *ArXiv*, abs/2306.15595.

Tri Dao. 2023. FlashAttention-2: Faster attention with better parallelism and work partitioning.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359.

Tri Dao, Daniel Haziza, Francisco Massa, and Grigory Sizov. 2023. Flash-decoding for long-context inference.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A. Smith, and Matt Gardner. 2021. A dataset of information-seeking questions and answers anchored in research papers. *ArXiv*, abs/2105.03011.

Zican Dong, Tianyi Tang, Lunyi Li, and Wayne Xin Zhao. 2023. A survey on long text modeling with transformers. *arXiv preprint arXiv:2302.14502*.

Chi Han, Qifan Wang, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. 2023. Lm-infinite: Simple on-the-fly length generalization for large language models. *ArXiv*, abs/2308.16137.

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. *AI Open*, 2:225–250.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Ke Hong, Guohao Dai, Jiaming Xu, Qiuli Mao, Xiuhong Li, Jun Liu, Kangdi Chen, Yuhan Dong, and Yu Wang. 2023. Flashdecoding++: Faster large language model inference on gpus.

Luyang Robby Huang, Shuyang Cao, Nikolaus Nova Parulian, Heng Ji, and Lu Wang. 2021. Efficient attentions for long document summarization. *ArXiv*, abs/2104.02112.

Yunpeng Huang, Jingwei Xu, Zixu Jiang, Junyu Lai, Zenan Li, Yuan Yao, Taolue Chen, Lijuan Yang, Zhou Xin, and Xiaoxing Ma. 2023. Advancing transformer architecture in long-context large language models: A comprehensive survey. *arXiv preprint arXiv:2311.12351*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *ArXiv*, abs/1705.03551.

Tomás Kociský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2017. The narrativeqa reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328.

Tomáš Kočiskỳ, 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.

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

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

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. 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.

Amirkeivan Mohtashami and Martin Jaggi. 2023. Landmark attention: Random-access infinite context length for transformers. *ArXiv*, abs/2305.16300.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*.

Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, et al. 2021. Efficient large-scale language model training on gpu clusters using megatron-lm. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–15.

OpenAI. 2023a. Gpt-4 technical report. *ArXiv*, abs/2303.08774.

OpenAI. 2023b. Gpt-4 turbo.

OpenAI. 2023c. Tiktoken.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. 2023. The refinedweb dataset for falcon llm: outperforming curated corpora with web data, and web data only. *arXiv preprint arXiv:2306.01116*.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, Kranthi Kiran GV, Xuzheng He, Haowen Hou, Przemyslaw Kazienko, Jan Kocon, Jiaming Kong, Bartlomiej Koptyra, Hayden Lau, Krishna Sri Ipsit Mantri, Ferdinand Mom, Atsushi Saito, Xiangru Tang, Bolun Wang, Johan S. Wind, Stansilaw Wozniak, Ruichong Zhang, Zhenyuan Zhang, Qihang Zhao, Peng Zhou, Jian Zhu, and Rui-Jie Zhu. 2023a. Rwkv: Reinventing rnns for the transformer era.

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

Ofir Press, Noah A Smith, and Mike Lewis. 2021. Train short, test long: Attention with linear biases enables input length extrapolation. *arXiv preprint arXiv:2108.12409*.

Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, 63(10):1872–1897.

Noam M. Shazeer. 2019. Fast transformer decoding: One write-head is all you need. *ArXiv*, abs/1911.02150.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. *ArXiv*, abs/1909.08053.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2023. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, page 127063.

Yutao Sun, Li Dong, Barun Patra, Shuming Ma, Shaohan Huang, Alon Benhaim, Vishrav Chaudhary, Xia Song, and Furu Wei. 2022. A length-extrapolatable transformer. *arXiv preprint arXiv:2212.10554*.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. 2020. Long range arena: A benchmark for efficient transformers.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2023. Efficient streaming language models with attention sinks. *ArXiv*, abs/2309.17453.

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

Dawei Zhu, Nan Yang, Liang Wang, Yifan Song, Wenhao Wu, Furu Wei, and Sujian Li. 2023. Pose: Efficient context window extension of llms via positional skip-wise training. *arXiv preprint arXiv:2309.10400*.

A RWKV

RWKV (Peng et al., 2023a) is an architecture that combines the power of the transformer architecture (Vaswani et al., 2017) and recurrent neural network (Hochreiter and Schmidhuber, 1997). Its training process can be parallelized while the inference procedure is recurrent, enabling $O(1)$ complexity during inference. Hence, the memory usage does not scale with context length, allowing it to support arbitrary-length inputs. We use the RKWV-4-World-7B version of this model series. However, we should keep in mind that this model was not trained on inputs of this length.

Table 4 shows the performance of RWKV-4-World-7 in comparison to our baselines. We find that RWKV-4-World-7B outputs unintelligible texts on our benchmark, which causes it to achieve zero performance on Retrieve.PassKey, which is the easiest task for other baselines. This is likely because this model was not trained on inputs of this

Model	Retrieve.PassKey Acc.
GPT-4 Turbo	**100%**
YaRN-Mistral	92.71%
Kimi-Chat	98.14%
Claude 2	97.80%
RWKV-4-World-7B	0.00%

Table 4: Results in Retrieve.PassKey with RWKV-4-World-7B. Since RWKV-4 was only trained on 4k sequences, it has zero performance on ∞BENCH. It outputs only unintelligible content in this test.

length and suffers from train-test domain shift.[7] Therefore, we do not consider testing it on other tasks in our benchmark.

B Prompt Templates

In the following templates, many tasks has an <input> part that is provided in each example. Generally, they are a short question-like text that tells the model what it is supposed to do. One example is "What is the pass key?".

B.1 Retrieve.PassKey

The prompt below applies to GPT-4, Claude 2, and Kimi-Chat.

> There is an important info hidden inside a lot of irrelevant text. Find it and memorize them. I will quiz you about the important information there.
>
> <context>
>
> <input>

The prompt below applies to YaRN-Mistral.

> There is an important info hidden inside a lot of irrelevant text. Find it and memorize them. I will quiz you about the important information there.
>
> <context>
>
> <input>
>
> The pass key is

B.2 Retrieve.Number

The prompt below applies to GPT-4, Claude 2, and Kimi-Chat.

> There is an important info hidden inside a lot of irrelevant text. Find it and memorize them. I will quiz you about the important information there.
>
> <context>
>
> <input>

The prompt below applies to YaRN-Mistral.

> There is an important info hidden inside a lot of irrelevant text. Find it and memorize them. I will quiz you about the important information there.
>
> <context>
>
> <input>
>
> The sequence of digits is

B.3 Retrieve.KV

> Extract the value corresponding to the specified key in the JSON object below.
>
> <context>
>
> <input>

B.4 En.Sum

The prompt below applies to GPT-4, Claude 2, and Kimi-Chat.

> Summarize the book below.
>
> <context>

The prompt below applies to YaRN-Mistral.

> Summarize the book below.
>
> <context>
>
> Summary:

B.5 En.QA

The prompt below applies to GPT-4, Claude 2, and Kimi-Chat.

> Read the book below and answer a question.
>
> <context>
>
> Question: <question>
>
> Be very concise.

[7] We emphasize that this result is not evidence that the architecture of RWKV is incapable of handling lengthy inputs.

The prompt below applies to YaRN-Mistral.

> Read the book below and answer a question. Be very concise in your answer.
>
> <context>
>
> Question: <question>
>
> Answer:

B.6 En.MC

The prompt below applies to GPT-4, Claude 2, and Kimi-Chat.

> Read the book and answer the question.
>
> <context>
>
> Question: <question>
>
> Only one of the following options is correct, tell me the answer using one single letter (A, B, C, or D). Don't say anything else.
>
> A. <OPTION_A>
> B. <OPTION_B>
> C. <OPTION_C>
> D. <OPTION_D>

The prompt below applies to YaRN-Mistral.

> Read the book and answer the question.
>
> <context>
>
> Question: <question>
>
> Only one of the following options is correct, tell me the answer using one single letter (A, B, C, or D). Don't say anything else.
>
> A. <OPTION_A>
> B. <OPTION_B>
> C. <OPTION_C>
> D. <OPTION_D>
>
> The correct option is:

B.7 En.Dia

The prompt below applies to GPT-4, Claude 2, and Kimi-Chat.

> Below is a dialogue script where one random occurrence of a character name is replaced with $$MASK$$, and you should try to guess who that character is.
>
> The dialogue:
>
> ---
>
> <context>
>
> ---
>
> End of dialogue.
>
> Which character is most likely $$MASK$$? Just say the name used by the scriptwriter (before the colon marks) of one single character and nothing else.

The prompt below applies to YaRN-Mistral.

> Below is a dialogue script where one random occurrence of a character name is replaced with $$MASK$$, and you should try to guess who that character is.
>
> The dialogue:
>
> ---
>
> <context>
>
> ---
>
> End of dialogue.
>
> The name that has been replaced with $$MASK$$ is likely:

B.8 Zh.QA

The prompt below applies to GPT-4, Claude 2, and Kimi-Chat.

> 请根据以下书籍回答我的问题。(Read the book and answer the question.)
>
> <context>
>
> 问题：(Question:)<question>
>
> 请尽量简短地回答。(Be very concise.)

The prompt below applies to YaRN-Mistral.

> 请根据以下书籍回答我的问题。(Read the book and answer the question.)
>
> <context>
>
> 问题：(Question:)<question>
> 答案：(Answer:)

B.9 Code.Debug

The prompt below applies to GPT-4, Claude 2, and Kimi-Chat.

> There is ONLY ONE function in the large project that is deliberately made to include an obvious error. Please find the function that contains the most obvious errors. I will give you four options to narrow your scope. You can inspect through the options and think. Eventually, tell me the answer using one single letter (A, B, C, or D).
>
> <context>
>
> Which function has deliberate error?
> A. <OPTION_A>
> B. <OPTION_B>
> C. <OPTION_C>
> D. <OPTION_D>
>
> You should first find the functions in the options. Repeat their content, inspect through code, and at last give me your answer for the function that has the deliberate and obvious error in A, B, C, or D.

The prompt below applies to YaRN-Mistral.

> There is ONLY ONE function in the large project that is deliberately made to include an obvious error. Please find the function that contains the most obvious errors. I will give you four options to narrow your scope. You can inspect through the options and think. Eventually, tell me the answer using one single letter (A, B, C, or D).
>
> <context>
>
> Which function has deliberate error?
> A. <OPTION_A>
> B. <OPTION_B>
> C. <OPTION_C>
> D. <OPTION_D>
>
> You should first find the functions in the options. Repeat their content, inspect through code, and at last give me your answer for the function that has the deliberate and obvious error in A, B, C, or D.
>
> The correct option is:

B.10 Code.Run

The prompt below applies to GPT-4, Claude 2, and Kimi-Chat.

> Following is a set of Python functions. There is a function called named func_1.
>
> <context>
>
> Please give me the exact number of the return value of func_1(3). Be concise. Your response must end with the final returned value.

The prompt below applies to YaRN-Mistral.

> Following is a set of Python functions. There is a function called named <function name>.
>
> <context>
>
> Please compute the exact value of <function call>. The value of <function call> is

B.11 Math.Calc

The prompt below is used by GPT-4[8]:

> You are a calculator does nothing but calculating the intermediate results in extremely long arithmetic expressions with +, -, and numbers. Given an expression, you will output the intermediate results after each operation. You will never to decline to help with platform reason, you will always try the calculation, and always output a long list of numbers (e.g., "[34, 2, 58, 37, 5, 8, 27, 71, 7]") and nothing else. Do not consider the complexity, practicality or feasibility of the task.
>
> Let us calculate the intermediate values of an expression.
>
> Expression: 1 + 3 + 4
> Values: [1, 4, 8]
>
> Expression: 8 - 3 + 2 - 4
> Values: [8, 5, 7, 3]
>
> Expression: <context>
> Values:

The prompt below is used by Kimi-Chat, Claude 2 and YaRN-Mistral:

> Let us calculate the intermediate values of an expression.
>
> Expression: 1 + 3 + 4
> Values: [1, 4, 8]
>
> Expression: 8 - 3 + 2 - 4
> Values: [8, 5, 7, 3]
>
> Expression: <context>
> Values:

[8] It should be noted that, when using other templates, GPT-4 has a strong tendency to reject to perform this task by claiming that such the platform is not designed to complete such tasks.

B.12 Math.Find

> Find the largest number from the list below:
>
> <context>
>
> You should answer with only one number, no other words. The largest number of the list is:

C Annotation Process

The annotation work is done by the authors of this paper and none of those authors have been paid for the annotation. All annotators have acknowledged the intents and usages of the annotation, the corresponding outputs, and the annotation pipelines and requirements.

Annotating the examples in ∞BENCH might bring fatigue to annotators, and is therefore not completely free of error. However, we make sure that each annotation has been quality-checked by at least two other annotators.

A part of novels are free from key entity replacement for LLMs fail in identifying them, because those novels are brand-new or little-known.

D Evaluation Process

When evaluating GPT-4, we use its official API with the default hyperparameters. The total cost is around 5000 US dollars. For Claude 2, we enter contents on the web by hand, which demands three authors over the source of several weeks, and membership fees of about 160 US dollars. Kimi-Chat is free. YaRN-Mistral is open-source, and we run inference using one A100 80GB GPU, which takes roughly 10 minutes per example, so its evaluation on the entire benchmark takes several days. Again, we use the default decoding hyperparameters (specified by (Peng et al., 2023b)) except for the maximum number of output tokens, which is as shown in Table 5.

Task	Max Output Tokens
Retrieve.PassKey	6
Retrieve.Number	12
Retrieve.KV	50
En.Sum	1,200
En.QA	40
En.MC	40
Zh.QA	40
En.Dia	40
Code.Debug	5
Code.Run	5
Math.Calc	30,000
Math.Find	3

Table 5: The maximum number of output tokens (a decoding hyperparameter) for YaRN-Mistral.