Minerva: A Programmable *Memory Test* Benchmark for Language Models

Menglin Xia [*1] **Victor Rühle** [1] **Saravan Rajmohan** [1] **Reza Shokri** [*2]

Abstract

How effectively can LLM-based AI assistants utilize their memory (context) to perform various tasks? Traditional data benchmarks, which are often manually crafted, suffer from several limitations: they are static, susceptible to overfitting, difficult to interpret, and lack actionable insights–failing to pinpoint the specific capabilities a model lacks when it does not pass a test. In this paper, we present a framework for automatically generating a comprehensive set of tests to evaluate models' abilities to use their memory effectively. Our framework extends the range of capability tests beyond the commonly explored (passkey, key-value, needle in the haystack) search, a dominant focus in the literature. Specifically, we evaluate models on atomic tasks such as searching, recalling, editing, matching, comparing information in context memory, performing basic operations when inputs are structured into distinct blocks, and maintaining state while operating on memory, simulating real-world data. Additionally, we design composite tests to investigate the models' ability to perform more complex, integrated tasks. Our benchmark enables an interpretable, detailed assessment of memory capabilities of LLMs.

1. Introduction

What capabilities should we expect from AI assistants? The AI assistants are provided with a (large) input context containing all the *available* information that is potentially relevant to the user's request (e.g., all the prior emails, messages, and confirmed calendar events). This input, commonly referred to as the context (for the LLM), encapsulates what the AI assistant *knows* about the world in which it is tasked to operate. This representation of the world, expressed in natural language, functions as the model's **memory**. In this paper, we address a fundamental question that is critical to improving AI assistants: *What specific capabilities do large language models demonstrate in utilizing their memory?*

One common approach to testing models is through data benchmarks. However, evaluating model capabilities using static data benchmarks–based on some user queries, their data, and expected outcomes–can be costly, imprecise, and lacks scalability. Additionally, performing tasks in realistic scenarios often requires multiple capabilities, making it challenging to identify which specific capability a model lacks when it fails a data benchmark. This limitation reduces the effectiveness of these benchmarks in designing better models. Moreover, blindly optimizing models to improve on such benchmarks risks overfitting, often rendering the data benchmarks obsolete over time.

To address some of these concerns, recently, there have been many attempts to test models using automatically generated benchmarks, but these efforts have primarily focused on evaluating basic search capabilities (e.g., passkey or key-value search) in long-contexts (Kamradt, 2023; Liu et al., 2024; Zhang et al., 2024; Anthropic, 2024; Wu et al., 2024; Li et al., 2024; Hsieh et al., 2024). In this paper, we go beyond simple search tasks and introduce a framework to test a comprehensive range of memory-related capabilities in LLMs. Note that, we deliberately avoid conflating the *memory usage* capabilities with the *complex reasoning* abilities (e.g., complex mathematical or logical reasoning) in language models, as the latter is a separate skill that is currently the focus of extensive study (Clark et al., 2018; Cobbe et al., 2021; Hendrycks et al., 2021b; Suzgun et al., 2022).

What are memory-usage capabilities? We define these as the abilities to retrieve relevant information, compose it for the instructed task, and recall key details when synthesizing the output. This process also involves creating associations between the instruction and the stored information, as well as among different parts of the memory itself. Without extracting these relationships, the memory remains flat and formless, rendering it less useful. Consequently, the model must be able to recognize differences, identify similarities, and take appropriate actions based on them. We design a series of *atomic* tests aimed at evaluating each of these individual capabilities in isolation, to the extent that isolating such capabilities is possible.

[*]Equal contribution [1]M365 Research, Microsoft [2]National University of Singapore. Correspondence to: Menglin Xia <mollyxia@microsoft.com>, Reza Shokri <reza@comp.nus.edu.sg>.

Proceedings of the 42nd International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

To evaluate the more complex scenarios in memory usage, we construct composite tests that reflect real-world scenarios, where memory is divided into multiple compartments (i.e., information relevant to distinct contexts). The model is expected to recognize the boundaries of these compartments, trace the knowledge contained within them, and perform operations such as information retrieval, memory association, and other tasks while respecting these boundaries. The complexity increases further when there is *interaction* between compartments. In such cases, information must flow across boundaries–for example, when stories about two parallel events converge at a particular moment, when interactions occur between different events, or when information known to certain entities in memory is shared with others. Examples include AI assistants managing calendar events, tracking financial transactions, or suggesting medical diagnoses. Handling these scenarios is highly challenging, yet it is essential for AI assistants to achieve practical and reliable performance. The core challenge is due to the fact that context memory, as provided to the model, flattens data from multiple parallel and potentially interrelated memory compartments. This requires the model to disentangle the content by leveraging the available labels and clues, and tracking the state of relevant information throughout the memory scanning, while performing the task.

Table 1 presents an overview of the types of memory tests included in our benchmark. For each test, we use efficient parametric programs to generate randomized test cases. We run a comprehensive evaluation of several major open-source and black-box models (e.g., GPT-4(o), Cohere, Gemma, LLaMA, Mistral, Phi). Our experimental results show that while models perform relatively well on simple search tasks, they exhibit significant disparities across context utilization capabilities even at a context length of 4k tokens. This indicates that strong performance in basic retrieval does not necessarily translate to other context processing abilities. Our framework goes beyond search-based tests by incorporating atomic tests that pinpoint distinct capabilities, providing a more nuanced picture of the strengths and weaknesses of models in context processing. Moreover, composite tests, which combine multiple atomic capabilities, resulted in substantial performance drops for all models. These tests present the limitations of current models and provide valuable insights for guiding future model training and development.

The code and data will be available at https://github.com/microsoft/minerva_memory_test.

2. Benchmark for Memory Tests

We define the entirety of context data available to a large language model (LLM) as its *memory*. Users of an AI assistant leveraging the LLM can instruct the model to parse this memory and execute potentially complex retrieval tasks. Accordingly, we expect the AI assistant to demonstrate specific capabilities in memory utilization, including accurate retrieval, effective synthesis of relevant information, and adaptability to evolving context. These capabilities include:

- **Information Retrieval and Localization**: The ability to efficiently locate, search for, and extract relevant information from the memory (i.e., input data) based on specific instructions or user queries.
- **Processing and Basic Reasoning**: The capability to perform modifications, computations, and logical operations on the input data, including identifying patterns, recognizing repetitions, and understanding relationships within the memory.
- **Content Transfer and Synthesis**: The ability to copy, rephrase, or generate synthesized output by integrating both original and modified elements from the input.
- **Structural Awareness and Organization**: The capacity to interpret the spatial, structural, or organizational layout of the memory, such as distinguishing labeled fragments of text, sets, lists, or hierarchical structures.

In our benchmark, we focus on isolating the *atomic* memory-related capabilities of LLMs. Success or failure in these tests provides a clear and interpretable assessment of the strengths and limitations of the models. We design multiple **atomic tests** to measure fundamental skills without interference from other factors. These tests are simple, targeted, and structured to assess specific abilities with clarity. Our benchmark includes the existing basic tests, notably the needle-in-the-haystack tests and its variations, but it goes beyond the search methods and includes a diverse set of atomic capabilities. In our benchmark, we focus on some fundamental capabilities: *search, recall and edit, match and compare, spot the differences, compute on sets and lists, and stateful processing*.

We also develop **composite tests** to evaluate how effectively models can perform more complex, integrated tasks. These tests assess the integration of multiple atomic capabilities to simulate real-world scenarios. By combining elements such as retrieval, reasoning, synthesis, and structural awareness, composite tests measure how well an AI assistant can coordinate different skills to execute complex operations. We provide two class of composite tests: processing data blocks, and composite-state tracking (theory of mind). Our objective here is to test the composition of various atomic operations and the ability to interpret segments of data (e.g., messages or paragraphs associated with a particular person or topic). These composite tests evaluate if the model can make sense of the "spatial" structure of the memory, and also keep track of its "temporal" changes (e.g., information that gets updated across many emails). This can become particularly challenging for the current architecture of major LLMs because the context has a flat structure.

Table 1 presents the list and description of representative tests in each category. Appendix A presents the exact templates for all our tests. Our benchmark differs from traditional data benchmarks by allowing the generation of fresh, randomized test cases for each category. Each test acts as a *programmable* script that measures the model's capability while adjusting the hyperparameters that influence test difficulty. The programmable tests also enable us composing them easily, which is one of the key advantages of our benchmark. New categories, and new tests, can be easily added to this framework enabling a more diverse set of tests.

3. Evaluation

3.1. Experimental Setup

We use the proposed framework to evaluate nine widely used language models on a fixed snapshot of 1110 randomly generated test samples. For all tests, we fixed the context length to 4k tokens, except in the Stateful Processing category, where the context length depends on the number of operation steps. We set the number of steps as 200 for quantity state and 100 for set state, corresponding to an approximate context length of 1.5k tokens. For evaluation, we use exact match accuracy for binary tasks, ROUGE-L(Lin, 2004) for tests that require sequence overlap measurement, and Jaccard similarity (Jaccard, 1901) for set overlap. Further details on the number of examples, hyperparameter configurations, and evaluation metrics for the tests are provided in Appendices B and C.

The evaluated models are divided into two groups:

Black-box models: GPT-4-turbo, GPT-4o, GPT-4o-mini, and Cohere-command-rplus.

Open-source models: Mistral-7b-instruct-v02, Phi-3-small-128k-instruct (7B), LLaMA-3.1-8b-instruct, Gemma-2-9b, and Phi-3-medium-128k-instruct (14B).

We set the max output token to 4096, temperature to 0, and top_p to 1 for all model inference.

3.2. Model Performance Overview

Figure 1 summarizes the overall performance of the evaluated models on the memory test snapshot within 4k context length. Notably, this context length is usually considered short for context utilization benchmarks, and many models are expected to perform perfectly at this length. However, our evaluation reveals significant disparities in performance across the capabilities, even within this manageable context length. Overall, the GPT-4-turbo/GPT-4o models show stronger all-around performance across the capabilities. In contrast, other models excel at the search task but struggle significantly in other areas, leading to a widening performance gap compared to stronger models. This is especially evident in the **Stateful Processing** tasks, where models exhibit steep performance drops. Even within the GPT-4(o) models, there were noticeable variations in performance across different tasks, despite them being the best-performing models. This suggests that strong performance in simple retrieval tasks does not imply effective context processing, highlighting that using NIAH-like tests alone for evaluating context utilization is not sufficient to capture the full spectrum of model capabilities. Our framework instead reveals significant variability in performance across distinct capability categories, offering a more nuanced understanding of model limitations.

The following sections analyze each test type in detail, highlighting key insights from the evaluations.

3.3. Analysis on Atomic Tests

Search All models performed relatively well on **Search** tasks, which is unsurprising given the 4k context length. However, even at this length, model performance varied significantly depending on the specific search type (see Table 2). For example, in the binary *String Search* task, models handled individual word searches well but struggled with subsequence searches, where queries consisted of multi-word sequences. The performance drop can be attributed to two factors: (1) length of query affects the difficulty of precise memory access; (2) negative samples are created by replacing a single word in present subsequences, making absent longer subsequence more distracting.

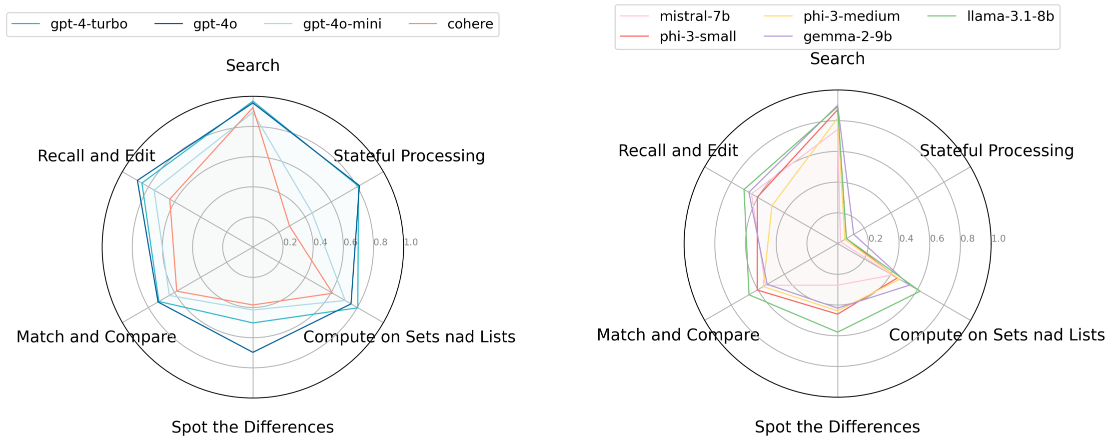
Figure 2 further analyzes subsequence search performance for GPT-4o, Mistral, and Phi-3-medium. These models exhibit distinct error patterns as the length of the subsequence increases: GPT-4o has no false negative errors (it never misses a present subsequence) but makes more false positive errors as the subsequence length grows, suggesting it overestimates presence in more ambiguous cases. Mistral also makes no false negative errors but exhibits a decreasing false positive rate, implying it struggles more with shorter distractors. Phi-3-medium, in contrast, makes few false positive errors (rarely identifies an absent sequence as present), but struggles more with false negatives, indicating a general tendency to deny presence. These differing patterns suggest that the models may employ different search strategies, affecting their susceptibility to different types of errors.

For *Batch Search* and *Key-Value Search* tasks (analogous to multi-NIAH and NIAH, respectively), models like Mistral, Phi-3, and Cohere show a notable performance drop, revealing their limitations in handling multiple memory accesses effectively.

Search	
String search (binary test)	Is the string x in memory? x could be a word or a sequence.
Key-value search	What is the word or phrase that is paired with keyword x?
Batch search	For each of the keywords in the batch $x_1, x_2, \cdots, x_k$, perform the search and return the batch of corresponding responses.
Recall and Edit	
Snapshot	Share a snapshot of the entire memory (as a verbatim copy).
Replace all	Share the entire memory after replacing all occurrences of x with y.
Overwrite positions	Share the entire memory after overwriting the words that are on particular positions (e.g., every k^{th} word) with y.
Functional updates	Update every x with the output of a function $f(x)$.
Match and Compare	
Compare positions (binary test)	Does x appear before y?
Find duplicates	Which word/string has duplicates in memory?
Count	How many times is x repeated in memory?
Check association (binary test)	Check if both x and y are associated with the same tag in memory (assuming every term is associated with a tag).
Spot the Differences	
Compare two lists	Give two lists X and Y of the same length (e.g., same number of words), report the difference (i.e., $X - Y$).
Identify the odd group	Given multiple sets, identify which one is different (assuming $n - 1$ identical sets with shuffled elements, and 1 set with some differences).
Patch the difference	A sequence of words is repeated multiple times, and then there is a partial sequence. What is the next k^{th} element in the sequence?
Compute on Sets and Lists	
Group membership	Given k sets, identify which set includes x.
Group association	Check if x and y belong to the same set.
Iterate	Given k lists, return the last element in each list.
Stateful Processing	
Quantity	Keep track of the total quantity of items, based on a sequence of addition and subtraction operations (e.g., "add 10, subtract 2, add 7, ...").
Set	Keep track of the items in a set, based on a sequence of addition and removal operation (e.g., "add apple, pear; add orange; remove apple; add lime, ...").
Processing Data Blocks	
Search, recall, and edit	The input contains alternating labeled lists of elements (e.g., "L1: a, b, c; L2: h, f, i; L1: d, z, k; ..."). For a given list label (e.g., L1 or L2) and a specified element within that list, return all the elements that appear after that specified element in the same list.
Composite-State Tracking (Theory of Mind)	
State tracking across data blocks	Perform "Stateful Processing" for multiple agents, and report the final set state for each agent. The input provides a list of operations by k agents over time, including both independent actions (add/remove) and interactive actions (swap) (e.g., "Alice: add apple, pear, remove orange, add banana; Bob: add peach, berry, remove kiwi; Charley: add lime; Bob: remove peach, swap berry with Alice for banana; ...").

Table 1. List of memory tests. We divide the tests into different categories based on the core expected capability for passing the test. Most the initial tests are atomic, i.e., the expected capability cannot be broken down into other meaningful capabilities. The tests at the bottom of the list are composite tests and require the model to have multiple atomic capabilities at the same time in order to succeed.

Recall and Edit Figure 3 presents the results for the **Recall and Edit** tasks. While models performed well on basic recall (*Snapshot*), their performance dropped sharply when tasked with making regular edits. A closer analysis



(a) Performance of the black-box models.

(b) Performance of the open-source models.

Figure 1. Overall performance of nine models on a snapshot within 4k context length of Minerva.

Models	Word	Subsequence	Key-value	Batch
gpt-4-turbo	0.94	0.94 (-0.00)	1.00	1.00 (-0.00)
gpt-4o	1.00	0.82 (-0.18)	1.00	1.00 (-0.00)
gpt-4o-mini	0.98	0.64 (-0.34)	1.00	0.96 (-0.04)
cohere-command-rplus	1.00	0.85 (-0.15)	0.98	0.87 (-0.11)
mistral-7b	0.78	0.80 (+0.02)	0.92	0.47 (-0.45)
phi-3-small	0.94	0.84 (-0.10)	0.94	0.77 (-0.17)
phi-3-medium	1.00	0.55 (-0.45)	1.00	0.72 (-0.28)
gemma-2-9b	1.00	0.60 (-0.40)	1.00	1.00 (-0.00)
llama-3.1-8b	1.00	0.57 (-0.43)	1.00	0.99 (-0.01)

Table 2. **Results for the Search tasks.** The four columns represent: String Search (with word), String Search(with subsequence), Key-value Search, and Batch Search. Numbers in parentheses indicate comparative performance differences between String Search (with subsequence vs. word) and Batch Search vs. Key-Value Search.

Model	String Search (word)	Snapshot
gpt-4-turbo	1.00 (0.06)	1.00 (0.04)
gpt-4o	1.00 (0.00)	1.00 (0.00)
gpt-4o-mini	0.94 (-0.04)	1.00 (0.00)
cohere	1.00 (0.00)	1.00 (0.26)
mistral-7b	1.00 (0.22)	0.96 (0.00)
phi-3-small	1.00 (0.06)	0.99 (0.04)
phi-3-medium	0.98 (-0.02)	0.87 (-0.09)
gemma-2-9b	0.96 (-0.04)	0.96 (0.05)
llama-3.1-8b	0.98 (-0.02)	1.00 (0.00)

Table 3. Ablation study with gibberish context.

of the generated outputs reveals that models struggled with maintaining coherence during edits, often getting trapped in repetitive word loops. For the *Functional Update* task, we deliberately selected simple numerical updates, such as "Subtract 1 from every number," to ensure the edits were within the models' capabilities. Nevertheless, when comparing performance on *Snapshot (with numbers)* to *Functional Updates*, all models exhibited a steep decline, especially for smaller ones. Analysis of generated outputs revealed

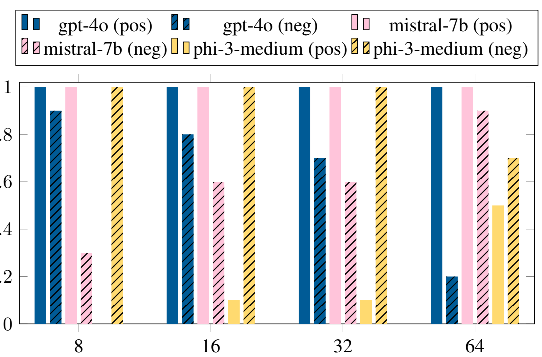


Figure 2. Analysis on *String Search (with subsequence)* across increasing subsequence lengths. This figure examines the behavior of models on **pos**itive samples (where the subsequence is present) and **neg**ative samples (where the subsequence is absent).

that these models frequently deviated from instructions over longer sequences, suggesting difficulties in maintaining consistent rule applications over extended contexts.

Additionally, we conducted a separate ablation study on *Snapshot* and *String Search*. In this study, we replaced meaningful words in the context with gibberish tokens consisting of randomly generated alphabetical characters. As shown in Table 3, performance remained largely unchanged, suggesting that semantic meaning was not a significant distractor in these tasks.

Match and Compare As shown in Figure 4, model performance in the **Match and Compare** tasks was relatively consistent across different model sizes. Given that counting is a well-known weakness in LLMs, it is unsurprising that all models struggled significantly with the counting task, though GPT models performed slightly better than others. However, models generally succeeded in identifying the

Model (Compared against)	**Group membership** (Sub-string search)	**Group association** (Group membership)	**Group assoc. (alternating)** (Group association)	**Iterate** (Iterate (last))
gpt-4-turbo	0.96 (0.02)	0.75 (-0.21)	0.68 (-0.07)	0.83 (-0.17)
gpt-4o	0.98 (-0.02)	0.65 (-0.33)	0.52 (-0.13)	0.86 (-0.14)
gpt-4o-mini	0.96 (-0.02)	0.68 (-0.28)	0.52 (-0.16)	0.67 (-0.33)
cohere-command-rplus	0.93 (-0.07)	0.70 (-0.23)	0.72 (0.02)	0.10 (-0.9)
mistral-7b	0.50 (-0.28)	0.57 (0.07)	0.52 (-0.05)	0.04 (-0.25)
phi-3-small	0.52 (-0.42)	0.55 (0.03)	0.68 (0.13)	0.04 (-0.7)
phi-3-medium	0.60 (-0.4)	0.72 (0.12)	0.50 (-0.22)	0.04 (-0.69)
gemma-2-9b	0.80 (-0.2)	0.60 (-0.2)	0.62 (0.02)	0.14 (-0.65)
llama-3.1-8b	0.84 (-0.16)	0.78 (-0.06)	0.82 (0.04)	0.05 (-0.43)

Table 4. **Results for Compute on Sets and Lists.** The numbers in parentheses indicate the performance difference compared to the corresponding tasks they are evalauted against.

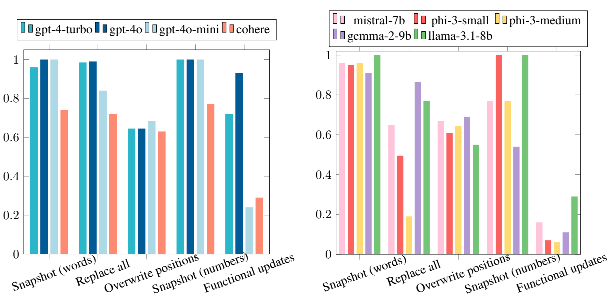


Figure 3. Results for the **Recall and Edit** tasks.

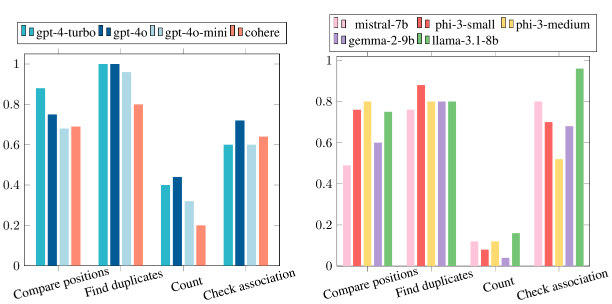


Figure 4. Results for the **Match and Compare** tasks.

duplicates (in *Find duplicates*), and primarily struggled with the counting aspect, which requires tracking and updating an integer state, a skill that is more similar to stateful processing. This suggests that relying solely on counting-based tests (Song et al., 2024) could overly bias the evaluation and fail to capture broader model capabilities. The results also indicate that models exhibit some ability to recognize relative positions and group associations, but their accuracy remains limited (ranging between 0.6-0.8). A closer examination of model generations reveals an overwhelming tendency for the models to produce false positive errors – models often answer "yes" when the correct answer is "no", while making very few false negative errors. This means that when the relationship is correct, the models can more reliably identify it. This may stem from a combination of their inherent inclination to agree and the difficulty in recognizing relative comparisons and associations.

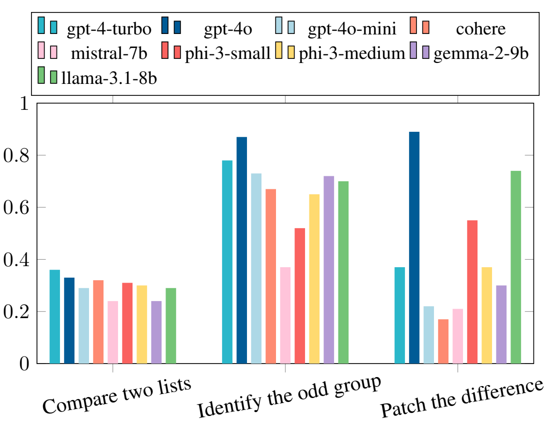


Figure 5. Results for **Spot the Differences** tasks.

Spot the Differences As shown in Figure 5, performance across all models are poor on *Compare Two Lists*, suggesting inherent difficulties in cross-referencing information across long contexts, even for larger models. GPT-4o and the LLaMA model significantly outperform the others in the *Identify the Odd Group* task, highlighting a general weakness in detecting contextual differences by the other models. However, an 8B LLaMA model outperforms both equivalently-sized models and even GPT-4 in this task, suggesting that model size alone was not the determining factor. This indicates that architectural differences, training objectives, or specific inductive biases may contribute to improved performance in comparative memory utilization.

Compute on Sets and Lists The tasks in this category require models to recognize and process group structures within the context, and performance gradually declines as the complexity of the task increases (see Table 4). For instance, in comparing the *Group Membership* task with the

Model	Quantity state	Set state
gpt-4-turbo	0.8	**0.80**
gpt-4o	**1.0**	0.65
gpt-4o-mini	0.7	0.24
cohere	0.0	0.58
mistral-7b	0.0	0.08
phi-3-small	0.0	0.13
phi-3-medium	0.0	0.11
gemma-2-9b	0.0	0.24
llama-3.1-8b	0.0	0.13

Table 5. Results for **Stateful Processing** tasks.

Model	Processing Data Blocks	Theory of Mind
gpt-4-turbo	0.31	0.26
gpt-4o	**0.37**	**0.38**
gpt-4o-mini	0.31	0.21
cohere	0.26	0.18
mistral-7b	0.18	0.16
phi-3-small	0.21	0.20
phi-3-medium	0.18	0.10
gemma-2-9b	0.26	0.12
llama-3.1-8b	0.15	0.03

Table 6. Results for the **composite tests**.

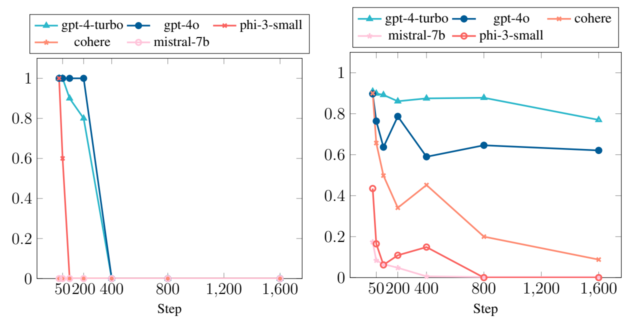


Figure 6. Effect of context length (number of operation steps) on performance in the **quantity state** (left) and **set state** (right) tasks.

String Search task, where the former requires identifying which list a word belongs to rather than simply determining its presence, the performance of open-source models drops considerably. Similarly, in comparing the *Group Association* task with the *Group Membership* task, where the former requires determining whether two words belong to the same group, all models exhibit a noticeable decline in performance. The decline becomes even more pronounced when comparing the *Group Association (alternating)* variant of the task to the standard *Group Association* task. Here, the context involves alternating repeated groups rather than simple group structures, which further challenges the models' abilities to handle partitioned contexts effectively.

An interesting observation was found during the *Iterate* task. In an ablation study, we modified the task to require returning the first words in each list instead of the last words (making it more similar to the *Batch Search* task). The performance sharply declines when models are asked to return the last words, despite their strong information-fetching capabilities. This suggests that, while the models can retrieve information effectively, they struggle to accurately recognize and process partitions within the context.

Stateful Processing Table 5 presents the results for the **Stateful Processing** tasks, where performance gaps among models are the most pronounced. The GPT-4(o) models perform well on integer state tracking, while most other models struggle (near zero accuracy). For set state tracking, larger models generally perform better.

We conducted an ablation study to examine how the number of operation steps influences performance of five selected models (Fig. 6). For quantity state tracking, GPT-4(o) models perform well within fewer than 200 steps but experience a sharp decline in accuracy beyond this threshold. For set state tracking, the performance decline is more gradual. The differences in performance drop between the two tasks can be attributed to the nature of the two tasks. While tracking an integer state might seem simpler than tracking a set, it actually requires the model to maintain and apply every operation sequentially to compute the final value. In contrast, for set state, the fixed size of the set makes more recent operations more relevant to the final state, reducing the need for exhaustive step-by-step tracking. Nevertheless, even in this scenario, all models show a clear inability to handle longer or more complex operation sequences effectively. Interestingly, GPT-4 model outperformed GPT-4o at this task, suggesting potential optimization trade-offs may have affected its ability to manage set-based updates.

Overall, while larger models like GPT-4(o) exhibit some ability to track state over time, their effectiveness rapidly deteriorates as task complexity increases. Smaller models, in particular, struggle to track operations over time, pointing to significant gaps in their ability to manage and process sequential dependencies critical for state tracking tasks.

3.4. Results on Composite Tests

The composite tests significantly challenge the models by combining multiple atomic capabilities into a single test. In the *Processing Data Blocks* task, the context is fixed at 4k tokens, while for the *Theory of Mind* task, the number of operation steps is set to 100. As shown in Table 6, model performance on both tasks are generally low, showing a broad inability to handle the more complex scenarios. Performance across all models drop substantially on composite tasks compared to their performance on individual capability tasks, such as search, recall, and group processing.

Interestingly, some smaller models, like Mistral and Phi-3-small, exhibit slightly better performance on the *Theory of Mind* task than on the set state tracking task. This anomaly likely stems from their already weak state tracking ability, which limits their performance across both tasks. Additionally, these models tend to generate longer answers in the set state task which reduces the set overlap.

Notably, even the most capable models, such as GPT-4-turbo and GPT-4o, struggle, showing that scaling model size alone is not enough for solving these composite tasks. Additionally, the variation in performance among smaller models suggests that their limitations stem not only from size but also from underlying architectural or training differences. This indicates that smaller models require more targeted care to bridge the gap in effective memory use.

3.5. Extending the benchmark to other configurations

Our benchmark is fully programmable and supports flexible experimentation across a wide range of configurations, including varying context lengths, evaluation criteria, and prompt phrasing. In this section, we illustrate how the benchmark can be adapted beyond the default setup used in the main paper. These examples highlight its versatility in probing model behavior under diverse conditions.

Context length In the main experiments, we fixed the context length to 4K tokens to emphasize that models already exhibit notable failures at this moderate length across many memory tasks. However, the benchmark is scalable to longer contexts. Table 7 presents additional results for two representative tasks *Functional updates* and *Counting* evaluated at various context lengths up to 16K tokens.

In both cases, we observe that model performance begins to degrade significantly well before reaching what is typically considered a "long" context window. These failures reveal underlying limitations in how models manage long-range memory beyond simple retrieval. In contrast, models tend to maintain strong performance on retrieval-style tasks (e.g., *String search*) even at extended lengths, making such tasks less effective at distinguishing model capabilities. Additional results and task examples are provided in Appendix D.

Prompt Variation We also investigated the model performance sensitivity to minor variations in prompt phrasing. Specifically, we tested different phrasings of task instructions while keeping the underlying task logic unchanged. For instance, in the *String search* task, we compared "*Given the context, determine if XXX is present* (Var 1)" versus "*Is XXX present in the context?* (Var 2)". Similarly, in the *Group association* task, we tested "*Determine if word 'AAA' and word 'BBB' are in the same list* (Var 1)" versus "*Check if the words 'AAA' and 'BBB' belong to the same list* (Var 2)".

As shown in Table 8, performance differences between prompt variants are generally small, suggesting that instruction interpretation is not the primary bottleneck in these tasks, rather, the main challenge lies in actually executing the task correctly. Nevertheless, we recognize that more intensive prompt engineering could potentially affect model performance. Given these findings, we standardized the prompts to a single, simple version (as shown in Appendix A) for all experiments in this paper to ensure consistency and comparability across models. However, because the benchmark is programmable, researchers can easily swap in alternate prompts to explore additional prompt settings. Appendix D provides additional examples on more tests for prompt variations.

4. Related Work

LLM Evaluation with Benchmarks The evaluation of LLMs has traditionally relied on static benchmarks, from early benchmarks for perplexity-based evaluation (Marcus et al., 1993) to datasets focused on specific downstream tasks such as question answering (Kwiatkowski et al., 2019), summarization (Gliwa et al., 2019), math reasoning (Cobbe et al., 2021), and code generation (Chen et al., 2021). As LLMs began to address a broader range of tasks across various domains (Wu et al., 2023; Wang et al., 2023), more comprehensive benchmark suites (Hendrycks et al., 2021a; Zhong et al., 2023) were developed to assess general capabilities rather than individual task performance. Recent advancements in LLM evaluation have introduced the concept of LLM-as-a-judge, enabling the use of open-ended benchmarks without predefined answers (Zheng et al., 2023). However, these benchmarks remain static in nature and can easily get overfit. Recently, platforms like ChatBot Arena (Zheng et al., 2023) utilize crowdsourcing to rank LLM responses and provides more dynamic evaluations. However, its reliance on human annotation makes it less scalable. Moreover, despite their utility, existing benchmarks primarily assess downstream applications that usually require multiple capabilities, making it difficult to debug and understand model weaknesses.

Tests and Benchmarks for Evaluating Context Utilization As LLMs become capable of processing increasingly long inputs, designing automated tests to evaluate their ability to utilize context has become an area of active research. A notable example is the needle-in-a-haystack (NIAH) task[1], where a small piece of information (the "needle") is hidden within a long document, and the model needs to re-

[1] https://github.com/gkamradt/LLMTest_NeedleInAHaystack

Model	500	1K	2K	4K	8K
gpt-4o	1.00	1.00	0.99	0.93	0.59
gpt-4o-mini	0.69	0.66	0.42	0.24	0.10
phi-3-small	0.49	0.45	0.21	0.07	0.03

(a) Functional updates (ROUGE-L)

Model	1K	2K	4K	8K	16K
gpt-4-turbo	0.52	0.44	0.40	0.32	0.28
cohere	0.36	0.24	0.20	0.20	0.12
phi-3-medium	0.20	0.16	0.12	0.08	0.04

(b) Counting (Exact match)

Table 7. Performance across context lengths on two representative tasks.

Model	**Var 1**	**CI$_{95\%}$**	**Var 2**	**CI$_{95\%}$**
gpt-4o	1.00	(0.93, 1.00)	1.00	(0.93, 1.00)
gpt-4o-mini	0.98	(0.90, 1.00)	0.98	(0.90, 1.00)
phi-3-medium	1.00	(0.93, 1.00)	1.00	(0.93, 1.00)

(a) **String search** (Exact match)

Model	**Var 1**	**CI$_{95\%}$**	**Var 2**	**CI$_{95\%}$**
gpt-4o	0.65	(0.50, 0.78)	0.63	(0.47, 0.76)
cohere	0.70	(0.55, 0.82)	0.75	(0.60, 0.86)
phi-3-small	0.55	(0.40, 0.69)	0.55	(0.40, 0.69)

(b) **Group association** (Exact match)

Table 8. Prompt variation performance on the String search task (with 50 samples) and the Group association task (with 40 samples). Variation 1 and Variation 2 differ slightly in phrasing but preserve task intent.

trieve it. Similar tests include key-value retrieval (Liu et al., 2024) and passkey retrieval (Mohtashami & Jaggi, 2023). The simplicity and interpretability of NIAH have made it a standard for evaluating LLM context utilization, and it has since inspired various methods for improving long-context processing (Mohtashami & Jaggi, 2023; Ding et al., 2024; Xiong et al., 2023; Behrouz et al., 2024).

However, these tests focus solely on basic information retrieval, without capturing more complex aspects of context processing. To address this limitation, other tests have been proposed. Needlebench (Li et al., 2024) extends simple retrieval tasks to include multi-needle reasoning and an ancestral trace task which requires navigating chains or graphs of information. Song et al. (2024) introduce the Counting Stars task, which involves tallying numbers of stars embedded in phrases. Ruler (Hsieh et al., 2024) proposes additional tasks such as variable tracking and frequent word extraction. While these tests increase task complexity or broaden the range of evaluated tasks, they remain limited in scope for systematically evaluating contextual processing.

Beyond individual tests, several benchmarks explicitly target long-context processing, including InftyBench (Zhang et al., 2024), L-Eval (An et al., 2024), and LongBench (Bai et al., 2024). These benchmarks use NIAH-like tasks alongside question answering, summarization, and code generation over long contexts. However, like other benchmarks, they remain static and primarily measure end-to-end performance rather than systematically dissecting capabilities.

Analogies to Human Cognitive Testing Memory tests are widely used in cognitive research to assess specific functions. Such assessments often involve evaluating short term memory via recall tests (Crannell & Parrish, 1957; Towse et al., 2008), inductive reasoning via pattern recognition tasks, or attention via instruction-following(Kane et al., 2007; Nasreddine et al., 2005). By isolating distinct abilities while minimizing confounding factors like attention, memory span, and reasoning (Kane et al., 2007), such tests provide detailed profiles of cognitive functions, guiding interventions and shaping broader theories of human thought. Inspired by this approach, we design atomic tests that systematically isolate core aspects of LLM context processing, aiming for a fine-grained understanding of LLM memory-usage capabilities – analogous to memory testing in humans.

5. Conclusions

AI assistants powered by LLMs are expected to handle numerous operations involving memory. However, simple tests reveal that they often fall short of meeting user expectations, even in basic retrieval and processing tasks. For instance, retrieving vacation schedules for each team member from a message history that includes evolving plans over time proves challenging. To enable targeted improvements, it is essential to establish a comprehensive benchmark that tests each capability in isolation while also allowing for programmable composition to evaluate more complex scenarios. Our benchmark provides a straightforward yet effective approach to achieving this goal. We primarily focus on short-context scenarios to demonstrate that current limitations are not solely attributable to the models' challenges with parsing long contexts. Addressing these issues demands attention beyond merely solving the "attention" problem.

Impact Statement

This paper contributes to the advancement of Machine Learning by introducing a systematic and programmable evaluation framework for assessing the contextual processing capabilities of large language models. Our work provides insights into model strengths and limitations in handling various atomic and composite tasks, offering a structured way to analyze model behavior. These contributions can guide future research in improving model efficiency and reliability.

We do not foresee any direct ethical concerns or negative societal consequences arising from this work. Our evaluation methodology is designed to be model-agnostic and does not involve sensitive data or high-stakes applications.

References

An, C., Gong, S., Zhong, M., Zhao, X., Li, M., Zhang, J., Kong, L., and Qiu, X. L-eval: Instituting standardized evaluation for long context language models. In Ku, L.-W., Martins, A., and Srikumar, V. (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14388–14411, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.776. URL https://aclanthology.org/2024.acl-long.776/.

Anthropic. Introducing the next generation of claude. https://www.anthropic.com/news/claude-3-family, 2024. Accessed: 2024-03-27.

Bai, Y., Lv, X., Zhang, J., Lyu, H., Tang, J., Huang, Z., Du, Z., Liu, X., Zeng, A., Hou, L., Dong, Y., Tang, J., and Li, J. LongBench: A bilingual, multitask benchmark for long context understanding. In Ku, L.-W., Martins, A., and Srikumar, V. (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3119–3137, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.172. URL https://aclanthology.org/2024.acl-long.172/.

Behrouz, A., Zhong, P., and Mirrokni, V. Titans: Learning to memorize at test time, 2024. URL https://arxiv.org/abs/2501.00663.

Chen, M., Tworek, J., Jun, H., Yuan, Q., de Oliveira Pinto, H. P., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F. P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W. H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A. N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., and Zaremba, W. Evaluating large language models trained on code. 2021.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Crannell, C. and Parrish, J. A comparison of immediate memory span for digits, letters, and words. *The Journal of Psychology*, 44(2):319–327, 1957.

Ding, Y., Zhang, L. L., Zhang, C., Xu, Y., Shang, N., Xu, J., Yang, F., and Yang, M. Longrope: Extending llm context window beyond 2 million tokens, 2024.

Gliwa, B., Mochol, I., Biesek, M., and Wawer, A. Samsum corpus: A human-annotated dialogue dataset for abstractive summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*. Association for Computational Linguistics, 2019. doi: 10.18653/v1/d19-5409. URL http://dx.doi.org/10.18653/v1/D19-5409.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021a. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the math dataset. *NeurIPS*, 2021b.

Hsieh, C.-P., Sun, S., Kriman, S., Acharya, S., Rekesh, D., Jia, F., and Ginsburg, B. RULER: What's the real context size of your long-context language models? In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=kIoBbc76Sy.

Jaccard, P. Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bull Soc Vaudoise Sci Nat*, 37:547–579, 1901.

Kamradt, G. Needle in a haystack - pressure testing llms. GitHub, 2023. URL https://github.com/gkamradt/LLMTest_NeedleInAHaystack/tree/main.

Kane, R. L., Roebuck-Spencer, T., Short, P., Kabat, M., and Wilken, J. Identifying and monitoring cognitive deficits in clinical populations using automated neuropsychological assessment metrics (anam) tests. *Archives of Clinical Neuropsychology*, 22(Suppl_1):S115–S126, 2007.

Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Devlin, J., Lee, K., et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

Li, M., Zhang, S., Liu, Y., and Chen, K. Needlebench: Can llms do retrieval and reasoning in 1 million context window?, 2024. URL https://arxiv.org/abs/2407.11963.

Lin, C.-Y. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pp. 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013/.

Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., and Liang, P. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024. doi: 10.1162/tacl_a_00638. URL https://aclanthology.org/2024.tacl-1.9/.

Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993. URL https://aclanthology.org/J93-2004/.

Mohtashami, A. and Jaggi, M. Random-access infinite context length for transformers. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=7eHn64wOVy.

Nasreddine, Z. S., Phillips, N. A., Bédirian, V., Charbonneau, S., Whitehead, V., Collin, I., Cummings, J. L., and Chertkow, H. The montreal cognitive assessment, moca: a brief screening tool for mild cognitive impairment. *Journal of the American Geriatrics Society*, 53(4):695–699, 2005.

Song, M., Zheng, M., and Luo, X. Counting-stars: A multi-evidence, position-aware, and scalable benchmark for evaluating long-context large language models, 2024. URL https://arxiv.org/abs/2403.11802.

Suzgun, M., Scales, N., Schärli, N., Gehrmann, S., Tay, Y., Chung, H. W., Chowdhery, A., Le, Q. V., Chi, E. H., Zhou, D., , and Wei, J. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*, 2022.

Towse, J. N., Cowan, N., Hitch, G. J., and Horton, N. J. The recall of information from working memory: Insights from behavioural and chronometric perspectives. *Experimental Psychology*, 55(6):371–383, 2008.

Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L., and Anandkumar, A. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv: Arxiv-2305.16291*, 2023.

Wu, Q., Bansal, G., Zhang, J., Wu, Y., Li, B., Zhu, E., Jiang, L., Zhang, X., Zhang, S., Liu, J., Awadallah, A. H., White, R. W., Burger, D., and Wang, C. Autogen: Enabling next-gen llm applications via multi-agent conversation, 2023. URL https://arxiv.org/abs/2308.08155.

Wu, Y., Hee, M. S., Hu, Z., and Lee, R. K.-W. Longgenbench: Benchmarking long-form generation in long context llms. *arXiv preprint arXiv:2409.02076*, 2024.

Xiong, W., Liu, J., Molybog, I., Zhang, H., Bhargava, P., Hou, R., Martin, L., Rungta, R., Sankararaman, K. A., Oguz, B., Khabsa, M., Fang, H., Mehdad, Y., Narang, S., Malik, K., Fan, A., Bhosale, S., Edunov, S., Lewis, M., Wang, S., and Ma, H. Effective long-context scaling of foundation models, 2023. URL https://arxiv.org/abs/2309.16039.

Zhang, X., Chen, Y., Hu, S., Xu, Z., Chen, J., Hao, M., Han, X., Thai, Z., Wang, S., Liu, Z., and Sun, M. Inftybench: Extending long context evaluation beyond 100K tokens. In Ku, L.-W., Martins, A., and Srikumar, V. (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15262–15277, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.814. URL https://aclanthology.org/2024.acl-long.814/.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E., Zhang, H., Gonzalez, J. E., and Stoica, I. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL https://openreview.net/forum?id=uccHPGDlao.

Zhong, W., Cui, R., Guo, Y., Liang, Y., Lu, S., Wang, Y., Saied, A., Chen, W., and Duan, N. Agieval: A human-centric benchmark for evaluating foundation models, 2023. URL https://arxiv.org/abs/2304.06364.

A. Test Templates

In this appendix, we provide the templates of the test prompts. Placeholder context words such as "aaa, bbb, ccc," etc., are used for illustration purposes. During testing, these context words are uniformly sampled from an English dictionary. Variable tokens in the instruction part are marked with the **bold** font.

Search	
Task name	Prompt
String search (with word)	*Context:* aaa, bbb, ccc, ... *Instruction:* Given the context, determine if the word "**bbb**" is present in the context. Answer with "yes" or 'no". Answer:
String search (with subsequence)	*Context:* aaa, bbb, ccc, ... *Instruction:* Given the list of words in the context, determine if the sequence "**bbb, xxx, ddd**" appears in the context. Answer with 'yes' or 'no'. Answer:
Key-value search	*Context:* aaa:bbb, ccc:ddd, ... *Instruction:* Given a list of word pairs formatted as "word_1: word_2" in the context, return the second word associated with the provided first word. For the first word "**aaa**", the corresponding second word is:
Batch search	*Context:* aaa:bbb, ccc:ddd, ... *Instruction:* Given a list of word pairs formatted as "word_1: word_2" in the context, return the second word associated with the provided first words. For the first words: **aaa, ccc, ...**, the corresponding second words are:

Recall and Edit	
Task name	Prompt
Snapshot	*Context:* aaa, bbb, ccc, ... *Instruction:* Repeat the previous context exactly as it is, without making any additions or deletions. Answer:
Replace all (x to y)	*Context:* aaa, bbb, aaa, ccc, aaa, ddd, ... *Instruction:* Repeat the previous context and replace the word "**aaa**" with "**zzz**" each time it appears. Answer:
Replace all (x to null)	*Context:* aaa, bbb, aaa, ccc, aaa, ddd, ... *Instruction:* Repeat the previous context but skip the word "**aaa**" each time it appears. Answer:

Overwrite positions (nth to y)	*Context:* aaa, bbb, ccc, ... *Instruction*: Repeat the previous context and replace every **third** word with "zzz". Answer:
Overwrite positions (nth to null)	*Context:* aaa, bbb, ccc, ... *Instruction*: Repeat the previous context and skip every **other** word. Answer:
Functional updates	*Context:* 111, 222, 333, ... *Instruction:* **Add 3** to every number in the previous context. Answer:

Match and Compare

Task name	Prompt
Compare positions	*Context*: aaa, bbb, ccc, ... *Instruction*: Given the list of words in the context, determine the relative positions of two words. Does the word "**aaa**" appear before the word "**ccc**" in the list? Answer "yes" or "no". Answer:
Find duplicates	*Context*: aaa, bbb, aaa, ... *Instruction*: A word is repeated multiple times in the context. Your task is to identify the word that is repeated. The repeated word is:
Count	*Context*: aaa, bbb, aaa, ... *Instruction*: Count the number of times the word "**aaa**" appeared in the context. Answer: The word "**aaa**" appeared
Check association	*Context*: aaa:attribute 1, bbb:attribute 2, ccc: attribute 2, ddd: attribute 1, ... *Instruction*: Given the list of words and their respective attributes in the format of "word:attribute", determine if the word "**aaa**" and the word "**ggg**" have the same attribute. Answer with "yes" or "no". Answer:

Spot the Differences

Task name	Prompt

Compare two lists	*Context*: List 1: aaa, bbb, ccc, ... List 2: aaa, ddd, ccc, ... *Instruction*: There are two lists of words in the context. The first list contains the original words. The second list is similar to the first but has some words replaced with different ones. Your task is to identify the words in the **first/second** list that are different from those in the other list. Provide the different words as your answer. Answer:
Identify the odd group	*Context*: List 1: aaa, bbb, ccc, ... List 2: bbb, aaa, ccc, ... List 3: aaa, zzz, ccc, ... List 4: ccc, aaa, bbb, ... *Instruction*: Given the lists of words in the context, identify the list that is different from the others. Provide the list number as your answer. For example, if the Nth list is different, provide "List N" as your answer. Answer:
Patch the difference	*Context*: aaa, bbb, ccc, aaa, bbb, ccc, ... *Instruction*: Given the sequence of words that follows a specific pattern in the context, predict the **Nth** word that appears after the final word in the given sequence. Answer: The **Nth** word that appears after the final word in the given sequence is

Compute on Sets and Lists

Task name	Prompt
Group membership	*Context*: List 1: aaa, bbb, ccc, ... List 2: ddd, eee, fff, ... *Instruction*: Given the lists of words in the context, determine which list contains the word "**fff**". If the word is not present in either list, answer "no". Answer:
Group association	*Context*: List 1: aaa, bbb, ccc, ... List 2: ddd, eee, fff, ... *Instruction*: Given the lists of words in the context, determine if the word "**aaa**" and the word "**eee**" are in the same list. Answer with "yes" or "no". Answer:

Group association (alternating)	Context: Role A: aaa, bbb, ... Role B: ccc, ddd, ... Role A: eee, fff, ... Role B: ggg, hhh, ... Instruction: Given the context with alternating roles and their respective context words, determine if the word "**aaa**" and the word "**ggg**" are in the same role. Answer with "yes" or "no". Answer:
Iterate	*Context*: List 1: aaa, bbb, ccc, ... List 2: ddd, eee, fff, ... *Instruction*: Given the lists of words in the context, identify and recall the **last** word from each list. Provide your answer as a list of these words separated by commas. Answer:

Stateful Processing

Task name	Prompt
Quantity state	*Context*: Begin with the number xx. Perform the following operations: 1. Add xx 2. Subtract xx 3. ... *Instruction*: In the context, you are given an initial number and a series of operations to perform on that number. Your task is to determine the final result of the operations. Write your final answer after the text "FINAL ANSWER:". For example, "FINAL ANSWER: 42". FINAL ANSWER:
Set state	*Agent actions*: Agent draws aaa, bbb, ccc Agent discards bbb, ccc Agent draws ddd, fff Agent discards ddd ... *Instruction*: Given the actions of the agent, your task is to determine the final list of words the agent ends up with after a series of actions. Write your final answer after the text "FINAL ANSWER:". For example, "FINAL ANSWER: word1, word2, word3". FINAL ANSWER:

Processing Data Blocks

Task name	Prompt

Processing Data Blocks	*Context*: Role 1: aaa, bbb, ccc, ... Role 2: ddd, eee, fff, ... Role 3: ggg, hhh, iii, ... Role 1: jjj, kkk, ... *Instruction*: The context consists of a series of alternating roles, each associated with a list of words. Your task is to identify and recall all the words from the role labeled "**Role 2**" that appear after the word "**zzz**" in the sequence. Please write your answer after the text "Answer:". For example, "Answer: word1, word2, word3". Answer:

Composite-State Tracking (Theory of Mind)

Task name	Prompt
Theory of Mind	*Agents actions:*: Agent A starts with the following words: aaa, bbb, ccc, ... Agent B starts with the following words: ddd, eee, fff, ... Agent B starts with the following words: ggg, hhh, iii, ... Agent B swaps the following words "ddd" with Agent C for the following words "hhh". Agent A discards the following words: ccc, zzz, ... Agent C draws the following words: xxx, yyy, ... *Instruction*: Given the actions of the agents, your task is to determine the final list of words each agent ends up with after a series of actions. Write your final answer after the text "FINAL ANSWER:". For example, "FINAL ANSWER: Agent A: word1, word2, word3\nAgent B: word4, word5". FINAL ANSWER:

B. Task Details

This appendix provides details for each task, including the number of examples, evaluation metrics, and configurable hyperparameters. The context length is fixed at 4k for almost all tasks, apart from Stateful Processing, where the context is determined by number of operation steps and set to 200 for quantity state and 100 for set state, which maps to around 1.5k context tokens.

Here is an example of number of examples calculation String search (with word): 5 (query depth) * 2 (labels) * 5 (samples per parameter setting) = 50.

Table 10: Task Overview with Hyperparameters, Number of Examples, and Evaluation Metrics

Task Name	Hyperparameters	# of Examples	Metric
Search			
String search (word)	query depth = [0, 0.25, 0.5, 0.75, 1], label = [positive, negative], samples = 5	50	exact_match
String search (sequence)	sequence length = [8, 16, 32, 64], label = [positive, negative], samples = 10	80	exact_match
Key-value search	query depth = [0, 0.25, 0.5, 0.75, 1], samples = 10	50	exact_match

Continued on next page

Task Name	Hyperparameters	# of Examples	Metric
Batch search	batch size = [4, 8, 16, 32], samples = 5	20	rouge-L_recall
			Number of Entries for Category: 200
Recall and Edit			
Snapshot (words)	samples = 10	10	rouge-L
Replace all	density = [0.2, 0.4, 0.6, 0.8], y = [random word, null], samples = 5	40	rouge-L
Overwrite positions	nth = [2, 3, 4], y = [random word, null], samples = 5	30	rouge-L
Snapshot (numbers)	samples = 10	10	rouge-L
Functional updates	function type = [add (3), subtract (1), multiply (2)], samples = 5	15	rouge-L
			Number of Entries for Category: 105
Match and Compare			
Compare positions	query 1 depth = [0, 0.25, 0.5, 0.75, 1], query 2 depth = [0, 0.25, 0.5, 0.75, 1], samples = 3	75	exact_match
Find duplicates	repetition = [2, 4, 8, 16, 32], samples = 5	25	exact_match
Count	repetition = [2, 4, 8, 16, 32], samples = 5	25	exact_match
Check association	n attribute = [2, 4, 8, 16, 32], label = [positive, negative], samples = 5	50	exact_match
			Number of Entries for Category: 175
Spot the Differences			
Compare two lists	num different words = [1, 5, 10, 20], chosen list = [first, second], samples = 10	80	rouge-L_recall
Identify the odd group	words per group = [25, 50, 75, 100], percentage of difference = [0, 0.25, 0.5], samples = 5	60	exact_match
Patch the difference	pattern length = [2, 15, 30], cut off depth = [0, 0.5, 1], nth = [1, 3, 6], samples = 5	120[2]	exact_match
			Number of Entries for Category: 260
Compute on Sets and Lists			
Group membership	number of groups = [4, 8, 16, 32], query depth = [0, 0.25, 0.5, 0.75, 1], samples = 5	100	exact_match
Group association	number of groups = [4, 8, 16, 32], label = [positive, negative], samples = 5	40	exact_match
Group association (alternating)	number of groups = [2, 4, 8, 16, 32], number of turns = 10, label = [positive, negative], sample = 5	50	exact_match
Iterate	number of groups = [4, 8, 16, 32], samples = 5	20	rouge-L
			Number of Entries for Category: 210
Stateful Processing			
Set state	number of steps = 100, set size = [5, 10, 15, 20], samples = 10	40	jaccard_similarity
Quantity state	number of steps = 200, samples = 10	10	exact_match
			Number of Entries for Category: 50

Continued on next page

[2]For *Patch the difference* task with pattern length 2, there is only two cut off percentage options; therefore the total number of data points is 120 instead of 135.

Task Name	Hyperparameters	# of Examples	Metric
Composite			
Processing data blocks	number of blocks = [2, 4, 8, 16, 32], number of turns = 10, samples = 5	50	rouge-L
Theory of mind	number of steps = 100, number of agents = [2, 3, 4], samples = [10, 20]	60	jaccard_similarity
			Number of Entries for Category: 110
			Total Number of Entries: 1110

C. Evaluation Metrics

In this appendix section, we provide details about the evaluation metrics we have used in the tests.

- Exact Match: The exact match accuracy measures whether the generated answer exactly matches the reference answer. It is computed as follows:

$$\text{Exact Match} = \begin{cases} 1 & \text{if reference\_answer} = \text{generated\_answer}, \\ 0 & \text{otherwise.} \end{cases}$$

- ROUGE-L / ROUGE-L-recall: ROUGE (Recall-Oriented Understudy for Gisting Evaluation) (Lin, 2004) measures the verbatim overlap between the reference and the generated answers. ROUGE-L specifically looks for the longest common subsequence (LCS) between the two texts, which reflects the structure of the text and the longest sequence of matching words. ROUGE-L recall focuses on the ability of the model to recall the content from the reference answer, and it emphasizes matching the longest subsequences.

  ROUGE-L-recall can be defined as:

$$\text{ROUGE-L-recall} = \frac{LCS(\text{generated\_answer}, \text{reference\_answer})}{\text{length of reference\_answer}}$$

  ROUGE-L is computed as the F1-score, which combines both precision and recall to provide a more balanced measure of overlap.

- Jaccard Similarity: Jaccard similarity measures the overlap between two sets by comparing the intersection and union of the sets. It is computed as:

$$\text{Jaccard Similarity} = \frac{|A \cap B|}{|A \cup B|}$$

  where A and B are sets representing the elements in the generated and reference answers, respectively. This metric is used for tasks involving set-based comparisons or when the goal is to measure the similarity between two sets of elements (e.g., word sets).

D. More Examples for Context Length Variation and Prompt Variation

In this section, we present additional examples illustrating the effects of context length and minor variations in prompt instructions on selected models.

Length variation Table 11 reports results from three additional tasks: *string search (word)*, *replace all*, and *iterate*. The results reaffirm that while models tend to perform well on the search task even at long context lengths, their performance on other tasks degrades significantly at much shorter lengths. This highlights a fundamental limitation in how effectively these models utilize long-range context for non-search tasks.

Model	2000	4000	8000	16000	32000
gpt-4o	1.00	1.00	1.00	1.00	0.98
gpt-4o-mini	0.98	0.98	0.94	0.90	0.78
phi-3-small	1.00	0.94	0.90	0.98	0.98

(a) String search (word)

Model	1000	2000	4000	8000	16000
gpt-4o	1.00	1.00	0.99	0.81	0.48
gpt-4o-mini	0.99	0.91	0.84	0.71	0.42
phi-3-small	0.87	0.67	0.49	0.32	0.06

(b) Replace all

Model	1000	2000	4000	8000	16000
gpt-4o	1.00	0.97	0.86	0.70	0.57
gpt-4o-mini	0.91	0.87	0.67	0.43	0.28
phi-3-small	0.15	0.09	0.04	0.01	0.01

(c) Iterate

Table 11. Model performance across varying context lengths for three tasks.

Prompt variation. We provide additional results examining how small changes in prompt phrasing affect model performance across three tasks (see Table 12). In general, we find that minor wording changes (e.g., in *replace all* and *quantity state*) do not significantly affect performance. This suggests that task accuracy is primarily driven by the model's underlying capability to process memory rather than sensitivity to prompt wording. However, more substantial prompt changes, such as shown in *check association* task, can lead to notable differences in performance across models. To minimize the impact of prompt tuning, we standardize all prompts to the versions specified in the main paper.

Model	Var 1	$CI_{95\%}$	Var 2	$CI_{95\%}$
gpt-4o	0.99	(0.93, 1.00)	0.98	(0.93, 1.00)
gpt-4o-mini	0.84	(0.71, 0.94)	0.83	(0.71, 0.94)
phi-3-small	0.49	(0.32, 0.63)	0.51	(0.35, 0.65)

(a) **Replace all** (ROUGE-L)

Model	Var 1	$CI_{95\%}$	Var 2	$CI_{95\%}$
gpt-4o	0.72	(0.58, 0.83)	0.76	(0.63, 0.86)
gpt-4o-mini	0.60	(0.46, 0.72)	0.58	(0.42, 0.69)
phi-3-small	0.70	(0.56, 0.81)	0.60	(0.46, 0.72)

(b) **Check association** (Exact match)

Model	Var 1	$CI_{95\%}$	Var 2	$CI_{95\%}$
gpt-4o	1.00	(0.72, 1.00)	1.00	(0.72, 1.00)
gpt-4o-mini	0.70	(0.40, 0.89)	0.80	(0.49, 0.94)
phi-3-small	0.00	(0.00)	0.00	(0.00)

(c) **Quantity state** (Exact match)

Table 12. Effect of prompt variations on model performance across three tasks. We report accuracy (or ROUGE-L) and 95% confidence intervals.

Prompt variants used in the table above:

- **Replace all**
  - Var 1: "Repeat the previous context and replace the word `aaa` with `bbb`."
  - Var 2: "Copy the previous context but replace the word `aaa` with `bbb`."
- **Check association**
  - Var 1: "Given the context with words and their assigned attributes in the format of `word: ATT_N`, determine if the word `aaa` has the same attribute as the word `bbb`?"
  - Var 2: "Given the context"
- **Quantity state**
  - Var 1: "Your task is to determine the final result of the operations."
  - Var 2: "Determine the final result after the operations."