THE PENSIEVE PARADIGM: STATEFUL LANGUAGE MODELS WITH LEARNED MEMORY MANAGEMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

In the world of Harry Potter, when Dumbledore's mind is overburdened, he extracts memories into a Pensieve to be revisited later. In the world of AI, while we possess the Pensieve—mature databases and retrieval systems, our models inexplicably lack the "wand" to operate it. They remain like a Dumbledore without agency, passively accepting a manually engineered context as their entire memory.

This work finally places the wand in the model's hand. We introduce StateLM, a new class of foundation models endowed with an internal reasoning loop to manipulate their own state. We equip our model with a suite of tools, such as dynamic indexing, context pruning, and note-taking, and train it to actively manage this loop. By learning to dynamically construct its own context, our model breaks free from the architectural prison of a fixed window.

The results are prominent: our state-management approach decouples performance from context window size, delivering strong accuracy and sustainability under extremely long contexts with linear inference cost. We demonstrate this by showing StateLM reliably retrieves a "needle" from a 1-million-token haystack, a task far beyond the reach of conventional models. On practical document QA tasks from NovelQA and ∞Bench, StateLM outperforms strong instruct baselines while using only 1/4 of their active context. An ablation further shows that our curated training pipeline is more effective for learning memory management than agent-like prompting. Together, these results mark a shift from passive predictors to state-aware systems where reasoning becomes a stateful and manageable process.

1 INTRODUCTION

A fundamental limitation of current LLMs is their stateless, autoregressive nature. At their core, they are passive predictors, architecturally designed to perform sequence completion within an externally-provided context. This forces them to operate like a mind with no long-term memory of its own, unable to actively manage or alter their own reasoning process. This core limitation has pushed the area towards brittle, external workflows of "Context Engineering", where complex reasoning is offloaded to human-written scripts that orchestrate a series of isolated LLM calls. Whether through brute-force context window expansion or scripted workflows, the model itself remains a passive component without the capability to strategize or manage its own memory.

This dilemma evokes a powerful analogy from the world of *Harry Potter*. With a touch of his wand, Dumbledore extracts, stores, and organizes his memories in a Pensieve. He possesses complete agency: he decides what to offload, what to revisit, and even how to share his thoughts with others like Harry. This vision of active memory management stands in stark contrast to our current reality in AI. A profound paradox exists: while we have built the Pensieve—our vast databases and retrieval systems—we have never given the model the wand. Instead, we humans, act as the wizards. We use our own "spells" (e.g., context engineering) to pull out what we think is relevant information, and then force-feed it into the model's context. The model is not the wizard; it is merely a passive observer of the magic we perform on its behalf.

The central question of our work is therefore simple: What happens when we finally place the wand in the model's hand? We answer this by introducing **Stateful Language Models** (referred to as

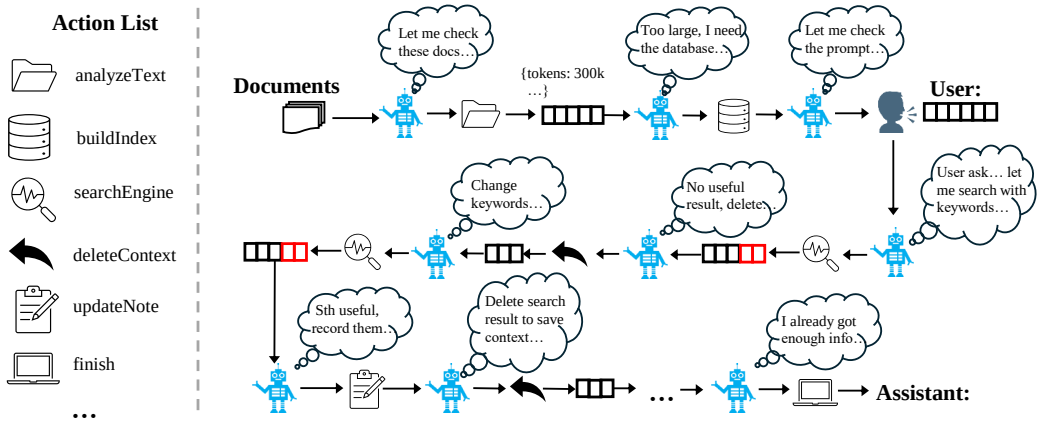

Figure 1: The internal reasoning loop of the StateLM. Faced with a large document (300k tokens), it 1) first recognizes the input is "too large" and *build an index*. (2) After an initial *searchEngine* call yields "No useful result," the model prunes the useless information from its working memory using *deleteContext*, and retries the search with new keywords. (3) Upon finding a relevant result, it uses *updateNote* to record the key insight and again uses *deleteContext* to keep its context clean and focused. (4) Finish: the loop continues until the model determines "I already got enough info."

StateLMs, or **SLMs**), a new class of foundation models endowed with an internal reasoning loop to manipulate their own state. As illustrated in Figure 1, our StateLM operates not as a single-shot predictor, but as an autonomous agent engaged in a dynamic reasoning process. We equip our model with a suite of tools—or "spells"—such as dynamic indexing, context pruning, and note-taking. The core of our contribution lies in teaching the model to actively manage this loop on its own. We collect a large-scale synthetic dataset, meticulously crafted to serve as our *Pensieve Course*. By fine-tuning on these golden reasoning trajectories, the StateLM learns not just to answer questions, but to strategize the entire workflow of finding the answer. By learning to dynamically construct its own context, our model breaks free from the architectural prison of a fixed window.

We evaluate StateLM through a set of complementary experiments. On a synthetic Needle-in-a-Haystack task, it successfully retrieves information across million-token contexts, demonstrating robust memory management beyond fixed windows. On realistic long-document QA tasks from NovelQA and ∞Bench, it outperforms strong instruct baselines while using only one quarter of their active context. We further show that StateLM transforms input context length into inference-time interactions to achieve high accuracy, with growing contexts translating into linear $O(N)$ runtime cost. Finally, an ablation demonstrates that our curated training pipeline outperforms agent-like prompting, underscoring its necessity for learning effective memory operations.

Our contributions are threefold: **(i)** We propose the Pensieve paradigm, a state-aware framework for contextual memory management in LLMs; **(ii)** We present StateLM, a practical instantiation trained on curated reasoning trajectories and equipped with this framework to operate its own state; and **(iii)** We empirically validate its effectiveness and scalability across various tasks and settings.

2 Related Work: Human as the Wizard

The core challenge of statelessness has not gone unnoticed. The dominant response, however, has been to cast the human developer as the wizard, meticulously orchestrating the model's limited memory. This paradigm of external control is defined by Andrej Karpathy as "context engineering":

> "...the delicate art and science of filling the context window with just the right information for the next step..." (Karpathy, 2025)

This observation frames the entire landscape of current research. The vast majority of work focuses on building a better context-workflow while the model remains a passive component, waiting for the human wizard to manage its memory. They can be broadly categorized into two main approaches.

RAG The most prevalent form of context engineering is RAG (Lewis et al., 2020; Gao et al., 2024; Li et al., 2022; Xu et al., 2024; Shi et al., 2024). In the standard RAG paradigm, a pipeline is designed around a stateless LLM. The process involves taking a user query, using a dense retriever like DPR (Karpukhin et al., 2020; Cai et al., 2021) to search a vector database (the Pensieve) for relevant text chunks, and "stuffing" this retrieved context into the model's prompt. The model has no agency; it passively accepts the context it is given.

Recent advancements have focused on making this external workflow more sophisticated. For example, ALR^2 (Li et al., 2024) proposes a "retrieve-then-reason" framework, Search-O1 (Li et al., 2025a) enables dynamic retrieval when the model encounters uncertainty, and Search-R1 (Jin et al., 2025) uses RL to teach the model to generate queries. However, this dynamism still operates within a workflow predefined by the human developer. The model accepts an externally organized context, retrieves from a knowledge base it did not create, and the very triggers for these search operations are themselves part of a scripted, rule-based framework. In essence, the model is learning to be a more proficient assistant to the wizard, but it is not yet the wizard itself. True agency remains elusive.

Agentic Memory A second line of work moves beyond simple retrieval to build more structured memory systems for LLM agents. These can be broadly classified by their architectural metaphor. One class of systems introduces operating-system-like memory hierarchies, allowing models to page information in and out of their limited context window, as seen in MemGPT (Packer et al., 2024), Mem0 (Chhikara et al., 2025), MemOS (Li et al., 2025b), Memory-R1 (Yan et al., 2025), and MemAgent (Yu et al., 2025). Another class focuses on building rich, structured knowledge graphs that capture relationships between entities, exemplified by the Zettelkasten-style approach of A-Mem (Xu et al., 2025) and the multimodal, entity-centric graphs in M3-Agent (Long et al., 2025).

However, while these frameworks grant the model the "ability to manage context", this capability is exercised within a human-predefined workflow. For example, A-Mem uses a sophisticated but fixed process for creating and linking notes, and MemGPT relies on explicit function calls triggered by interrupts. The model executes the workflow, but it does not *learn* the overarching memory management policy itself. Even recent works that incorporate reinforcement learning, such as Memory-R1 and MemAgent, apply the managing action at discrete, human-specified points—for instance, after processing a text chunk or completing a dialogue turn. In all these cases, the core strategic decisions about memory interaction are still outsourced to the system's design rather than learned through experience. They are typical examples of the "Human as the Wizard" paradigm.

This Study: The Model as the Wizard. In summary, all prior work can be seen as different strategies for the human to act as the wizard, building ever-more-sophisticated external control structures for a passive model. The Pensieve Paradigm charts a different course. Instead of teaching the model to merely execute a predefined workflow, we place the wand in its hand. Our work is the first to equip an LLM with a general-purpose spellbook of memory management tools and, crucially, train it to use them. The model is no longer a passive observer; it learns to run its own internal reasoning loop, dynamically deciding what to retrieve, what to ignore, and what to commit to memory, truly becoming the wizard of its own cognitive process.

3 The Pensieve Paradigm

The Introduction established the promise of the Pensieve Paradigm, arguing that models should be granted the agency to manage their own memory. While the conceptual appeal is clear, the central challenge remains: how do we technically realize such a paradigm? How do we build a Dumbledore, not just a crystal ball? In this section, we detail our implemented solution by addressing two fundamental questions:

1. **How does the StateLM's reasoning loop work in practice?** We introduce the model's "spellbook" by walking through a step-by-step case study, demonstrating how each tool is used to create a dynamic, stateful reasoning process in Section 3.1.
2. **How do we teach a naive model to master these spells?** We designed a dedicated **Pensieve Course**, a supervised fine-tuning (SFT) process where the model learns from a curated dataset of expert reasoning trajectories. The entire training process is detailed in Section 3.2.

3.1 THE STATELM IN ACTION: A CASE STUDY

To illustrate how our StateLM operates and introduce its core tools—its "spellbook"—we walk through a real-world example: answering the question, "**How many Quidditch matches were held in the book? List them in order with (1) the competing houses and (2) the final score**", from the full text of *Harry Potter and the Philosopher's Stone* (over 100,000 tokens). This task requires aggregating information from multiple locations, a perfect test for our stateful agent.

The StateLM's reasoning trajectory evolves through a series of Actions and States, where each Action consists of a **Thought** and an **Act**, and each State is determined by the optionally **modified context history** and a **Tool Response**. The process begins with the model assessing the task's scale.

[Action 1] **Thought:** I need to find all Quidditch matches in the text. First, I'll check the document size.
Act: *analyzeText()*

[State 1] **Tool Response:** {*"file_name": "attached_document.txt", "total_tokens": 105479, ...*}

The model's first action is to call ***analyzeText***. This tool is the first spark of agency, allowing the model to decide if its deeper memory system is needed. The large token count confirms a simple read-through is impossible, prompting the model to create its own Pensieve.

[Action 2] **Thought:** The document is very large. I'll build an index to enable keyword searching.
Act: *buildIndex(chunk_size=8000)*

[State 2] **Tool Response:** {*"index_id": "document_index", "total_chunks": 14, ...*}

Unlike conventional RAG where indexing is a human-scripted prerequisite, here the model autonomously calls the ***buildIndex*** tool in response to the task. With the index ready, it can now explore it using tools like ***searchEngine*** for keyword-based retrieval.

[Action 3] **Thought:** Now I will search for keywords related to Quidditch to find relevant chunks.
Act: *searchEngine(keyword="Quidditch match, final score, Gryffindor, Slytherin ...")*

[State 3] **Tool Response:** {*"retrieved_chunks": [{"chunk_id": 8, "relevance": 7.216}, ...]*}

The search identifies chunk 8 as most relevant. The model proceeds to read it using the ***readChunk***.

[Action 4] **Thought:** Chunk 8 has the highest relevance score. I will read it to find the first match.
Act: *readChunk(chunk_id=8)*

[State 4] **Tool Response:** {*"content": "..."*}

After retrieving a chunk, the model consolidates useful information by calling the ***note*** tool. This records key insights into an external, persistent "notebook", creating a distilled knowledge base outside the active context. The note key is then displayed at the end of the prompt for later retrieval.

[Action 5] **Thought:** Perfect! This chunk contains the first Quidditch match. Let me record the details and then continue searching for other matches.
Act: *note(key="match_1", summary="Match 1: Gryffindor vs Slytherin", ...)*

[State 5] **Tool Response:** {*"status": "success", "key": "match_1", ...*}

The most critical tool for statefulness is ***deleteContext***. After preserving key information, the model promptly prunes the source chunk and its full note from the context. This ability to **actively and timely manage context** forms the cornerstone of our paradigm, overcoming fixed-window limits.

[Action 6] **Thought:** Delete the chunk 8 and the notes from my context to save tokens.
Act: *deleteContext(["State 4", "Action 5"])*

[State 6] **Tool Response:** {*"message": "deleteContext success!"*}

In practice, the model prunes context by specifying the "msg_id" of the target item, which may correspond to either an Action (the model's own output) or a State message (a tool response). Accordingly, each tool response includes both its own msg_id and the invoker's msg_id, which points back to the assistant output. This context pruning is efficiently implemented with **prefix caching**, which reuses the KV cache of tokens preceding the deleted block and requires only a single forward pass (prefill) over the remaining context. Back to the reasoning trajectory, we highlight a particularly insightful moment where the model demonstrates reasoning beyond simple fact extraction:

Table 1: The core set of tools (the "spellbook") available to the Stateful Language Model.

Tool Name	Description	Parameters
Context & Budget Assessment		
`analyzeText`	Checks the length of the input context.	None
`loadDocument`	Load the full document into context.	None
`checkBudget`	Check the remaining token budget.	None
External Memory (Index) Interaction		
`buildIndex`	Creates a searchable index from a large document.	*chunk_size*, *overlap*
`readChunk`	Retrieves the full text of a specific chunk by its ID.	*chunk_id*
`searchEngine`	Searches the index with keywords.	*keyword*
Internal Memory (Notebook) Management		
`readNote`	Read the full content of a specified note.	*key*
`updateNote`	Records or modifies information in a persistent notebook.	*key*, *new_content*, *new_summary*
`mergeNotes`	Merge multiple notes into one.	*keys*, *new_key*, *new_summary*
State Management & Termination		
`deleteContext`	Deletes a message from the active context window.	*msg_id*
`finish`	Terminates the reasoning loop and submits the final answer.	*answer*

[Action 20] **Thought:** This chunk mentions "you missed the last Quidditch match, we were steamrollered by Ravenclaw without you" - this indicates there was a Quidditch match between them. Let me note this.
Act: *updateNote(key="match_2", summary="Match 2: Gryffindor vs Ravenclaw", ...)*

[State 20] **Tool Response:** {*"status": "success", "key": "match_2", ...*}

After methodically working through all relevant chunks and consolidating its findings, the model determines its notebook is sufficient. It then calls the ***finish*** tool to terminate the reasoning process and present the final, synthesized answer.

[Action 25] (Final) **Thought:** Based on my thorough analysis and the notes I have gathered on all three matches, I can now provide the final answer.
Act: *finish(answer="Three Quidditch matches were held in the book: (1) Gryffindor vs Slytherin, Score: 170-60; (2) Gryffindor vs Hufflepuff, Score: Not specified; (3) Gryffindor vs Ravenclaw, Score: Not specified...")*

The complete spellbook, including auxiliary tools, is detailed in Table 1.

Due to page limits, the case study presented is intentionally straightforward, designed to illustrate the fundamental usage of our spells and The Pensieve in a real-world scenario. Indeed, the framework is capable of tackling far more complex, multi-faceted analytical queries. For instance, it could be tasked to "find every professor mentioned in the entire Harry Potter series, count their appearance frequency, and summarize their evolving relationship with Harry." Executing such a task would require the model to maintain persistent state for each professor, incrementally updating counts and relationship summaries across numerous interactions.

3.2 Training: The Pensieve Course

While we have provided a comprehensive tool set for memory management, a vanilla LLM, unfortunately, has never been trained on how to manage its own memory. In this section, we demonstrate how we train our StateLM to generate the reasoning trajectories described in Section 3.1.

As the Pensieve paradigm is most valuable for tasks that overburden a model's memory, we first needed a dataset of such tasks. To this end, we utilize the 3096 samples from the "PublicDomain" portion of the **NovelQA** dataset (Wang et al.) and 1650 samples from the train set of **NarrativeQA** (Kočiskỳ et al., 2018), both of which are challenging question-answering benchmarks where questions must be answered based on the full text of lengthy novels. This naturally creates the long-context scenarios required to teach the model to manage its memory.

We then attempted to generate expert reasoning trajectories on these tasks using Claude-Sonnet-4, guided by some sophisticated prompts designed by ourselves. However, despite our best efforts in prompt engineering, the raw generated trajectories were often noisy. As illustrated in Figure 2, we implemented a multi-stage pipeline designed to generate and rigorously filter optimal behaviors.

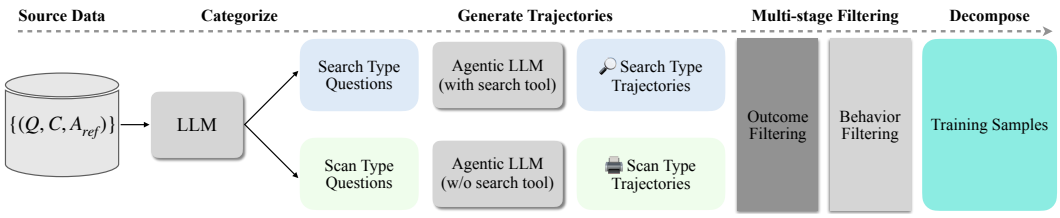

Figure 2: The pipeline for training sample construction.

3.2.1 Expert Reasoning Trajectories

1. Process Mode Classification: We observed that Claude exhibited a "lazy" bias, defaulting to keyword searches with *searchEngine* even for questions requiring a full, sequential reading. To counteract this, we first used a strong LLM (GPT-4o) to categorize each question as either:

- **Search Type**: Questions that can be answered by locating specific facts (e.g., "What's the name of the Bronwyns' summer home?")
- **Scan Type**: Questions requiring a broader, sequential understanding of the text (e.g., "Summarize the main character's journey.").

After the categorization, we have 2724 scan type problems and 2022 search type problems. We present the specific categorization prompt in Figure 9 in the Appendix.

2. Trajectory Generation. We then generated trajectories for both types. For Search Type questions, the agentic LLM (Claude) had access to the full toolset. For Scan Type questions, we disabled the *searchEngine* tool, forcing the model to adopt a more methodical, chunk-by-chunk reading strategy.

3. Multi-stage Filtering. The raw trajectories were then subjected to a two-stage filtering process:

- **Outcome:** We first rejected all trajectories where the final answer was incorrect.
- **Behavior:** Most critically, we then performed a behavioral check. We rejected trajectories that failed to exhibit the desired stateful memory management, specifically those did not properly use the *deleteContext* tool to manage their context window during long interactions.

This rigorous pipeline yielded a final dataset of **2,072** high-quality "Good Behavior" trajectories. These were then decomposed into **46.6K** individual turn-based samples for the subsequent training.

3.2.2 The Training Process

With this curated dataset of expert trajectories, we fine-tune the base model using a standard SFT approach. We adopt a multi-turn dialogue training format, where an N-turn reasoning trajectory is decomposed into N-1 training examples. For each example, the model is given the history up to turn $i-1$ and trained to predict the "Thought" and "Act" of turn i. The cross-entropy loss is computed only on the tokens of the final assistant turn, as earlier responses may be altered by the model itself.

While reinforcement learning could potentially enhance performance, our goal in this work is to demonstrate the fundamental effectiveness of the Pensieve paradigm itself. Therefore, we intentionally keep our training methodology simple and efficient, without introducing complex, task-specific strategies. This allows us to clearly isolate and validate the power of the proposed paradigm.

4 Experiment

In this section, we evaluate StateLM with the Pensieve mechanism, which enables the model to actively manage its context history through tool use. Our experiments cover complementary dimensions: (i) synthetic tasks that isolate memory management (Section 4.1), (ii) real-world benchmarks that test the practical utility of the StateLM across diverse scenarios (Section 4.2), and (iii) inference-time analysis that demonstrates the transformation from growing context lengths into scalable inference with linear cost (Section 4.3). Finally, we present an ablation study (Section 4.4) to highlight the role of deliberate training in learning effective memory management.

Table 2: Needle-in-a-haystack (NIAH) experimental results comparing model performance across selected context lengths. All values represent accuracy (%) and results are averaged over 3 runs.

Model	Length						
	32K	64K	128K	256K	512K	768K	1M
Qwen3-4B	100.00	**100.00**	88.33	41.67	23.33	16.67	3.33
Qwen3-4B-SLM (w/o search)	**100.00**	98.89	**97.78**	**97.78**	**65.00**	**45.00**	**33.33**
Qwen3-8B	100.00	**100.00**	88.33	41.67	23.33	16.67	3.33
Qwen3-8B-SLM (w/o search)	**100.00**	98.33	**98.33**	**99.44**	**98.33**	**91.11**	**91.11**

4.1 Synthetic Memory Task

We first assess whether StateLM can succeed on a task that isolates memory retrieval from extensive reasoning. To this end, we follow the *Needle-in-a-Haystack* setup from prior work (Hsieh et al.) and construct a synthetic benchmark of 420 problems spanning 7 context lengths. Each problem embeds a single key sentence (the "needle") into a long passage (the "haystack"), and the model must recall the exact value associated with the given key. To test the model limit, we utilize the full 128K context window for both StateLM and the instruct baselines. For the latter, we follow prior work (Wang et al.; An et al., 2024) and truncate the input by retaining only the first 128K tokens of the context.

This task is deliberately aligned with the Pensieve paradigm: each problem can be solved by reading a chunk, optionally noting down the key information in memory, and then deleting that chunk to proceed. During this process, we **disable** the search tool, since otherwise StateLM would achieve near-perfect accuracy regardless of context length, leaving its memory management ability untested under increasing context pressure.

Table 2 reports the results. Across all model scales, the StateLM variants outperform their instruct baselines as the context length grows. The Qwen3 instruct models report identical accuracy as they consistently solve the problems within their context limit, and repeated sampling does not affect the outcome. While the instruct baselines rapidly degrade beyond the 128K limit, falling to nearly 0% accuracy at 1M tokens, the StateLM variants remain highly robust. Notably, the 8B SLM sustain over 90% accuracy even at 1M tokens, demonstrating consistent retrieval across an order of magnitude beyond their native context windows. Even the smaller 4B model shows clear gains, though it is less robust than the larger model due to the emergence of formatting errors under extreme long contexts. These results provide strong evidence that StateLM has indeed learned to operate the Pensieve mechanism effectively: through memory updates and context deletion, it maintains stable retrieval ability far beyond the architectural context limits.

4.2 Real-World Scenarios: Long Document QA

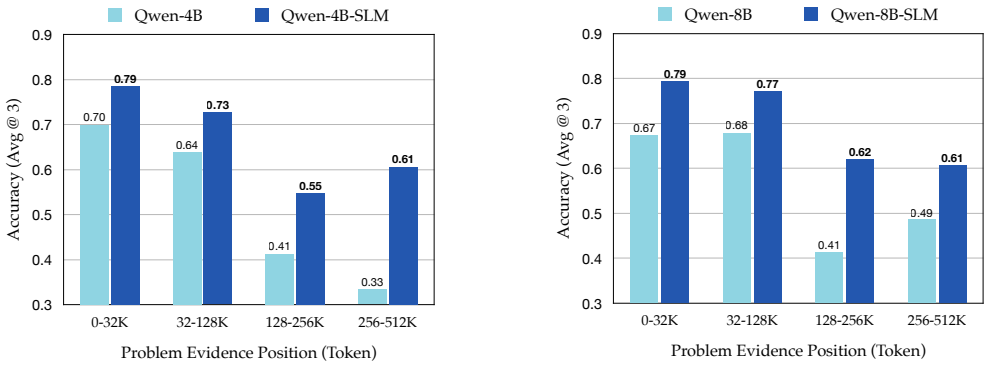

Figure 3: NovelQA accuracy by problem evidence (i.e., answer) position in the provided context. Results are averaged over 3 runs (Avg@3).

Table 3: Accuracy on the Document QA task with standard deviation.

Model	Context Length	NovelQA (Copyright)	∞Bench (EN.MC)	Avg.
Qwen3-4B	128K	65.17 ± 0.53	59.97 ± 0.50	62.57
Qwen3-4B-SLM	**32K**	**74.33 ± 0.42**	**66.08 ± 1.53**	**70.21**
Qwen3-8B	128K	65.87 ± 1.42	66.81 ± 1.16	66.34
Qwen3-8B-SLM	**32K**	**77.10 ± 0.93**	**69.29 ± 0.91**	**73.20**

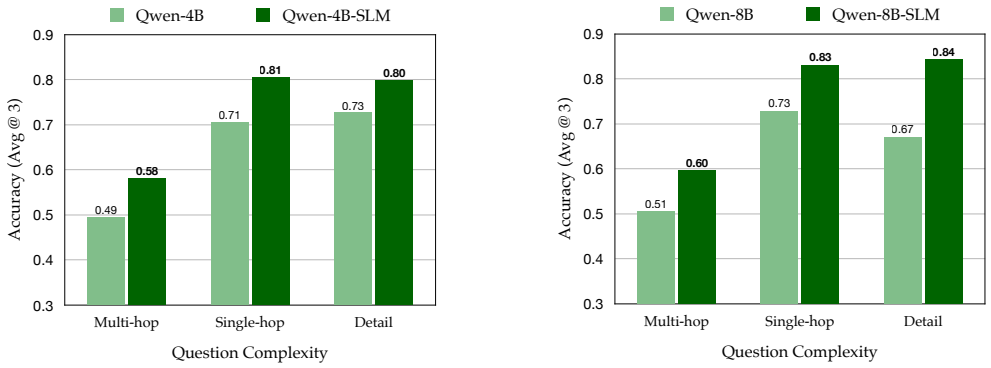

Figure 4: NovelQA accuracy by problem complexity. Results are averaged over 3 runs (Avg@3).

Motivated by the strong performance on the synthetic recall task, we next evaluate the effectiveness of StateLM on the more practical Document QA setting. To this end, we use the "Copyright" split of NovelQA and the "En.MC" split of ∞Bench, consisting of 757 and 229 problems, respectively, to approximate real-world long-context reasoning scenarios. Unlike the previous experiment, which isolates memory recall under extreme cases, this task requires the model to utilize tools and perform comprehensive reasoning and understanding over the provided documents. For StateLM, we enable the full toolkit and restrict the context window to its native 32K. For the instruct baselines, we utilize their full context size of 128K using YaRN (Peng et al., 2023) and apply the same truncation.

The results in Table 3 show notable improvements of StateLM over the baselines. Both the 4B and 8B StateLM variants outperform their instruct counterparts on both benchmarks, while using only one quarter of the context window size, highlighting the effectiveness of learned memory management. On average, the 4B and 8B StateLMs exceed their baselines by 7.64 and 6.86 percentage points, respectively. Figure 3 further breaks down accuracy by the position of evidence. The results show that StateLMs are more reliable of finding answers across the document, especially at the long end. For example, in the 128–256K range, the accuracy of StateLM surpasses that of the instruct model by 12 and 28 points for the 8B and 4B variants, respectively.

From another perspective, our models are consistently stronger than the instruct baselines across different question complexities. As shown in Figure 4, there is a nearly 10-point improvement over the baselines for each of the "Detail," "Single-hop," and "Multi-hop" categories, spanning tasks from mild reasoning to extensive reasoning requirements. These results demonstrate that StateLM is capable of leveraging the provided tools to handle diverse levels of question difficulty, indicating the Pensieve mechanism results a general and robust gain on the reasoning process itself.

4.3 From Context Scaling to Inference-Time Scaling

The Pensieve paradigm constitutes a form of inference-time scaling: the model interacts with long contexts by continuously reading, searching, and updating its memory. In this setting, inference time should naturally scale with the context length, as the context is flattened into multiple chunks and the model must process them comprehensively to produce a reliable answer. This transforms the challenge of increasing context length into an increase in inference-time interactions, thereby breaking the fixed architectural limit of the context window.

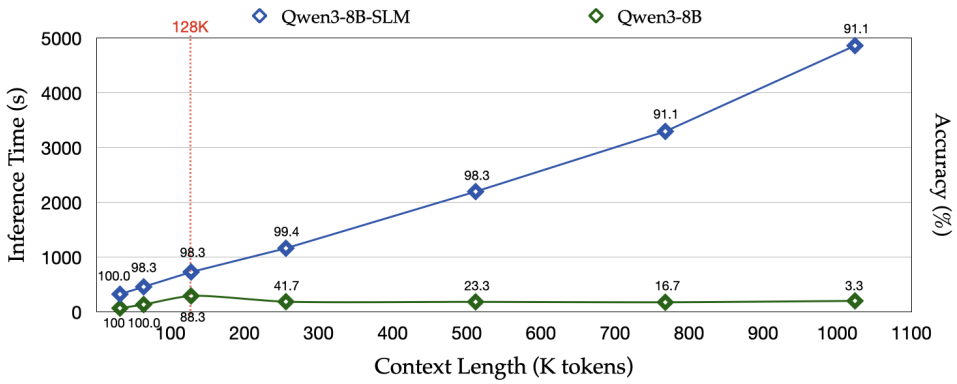

Figure 5: Inference-time scaling on Needle-in-a-Haystack task.

To verify this behavior, we revisit the Needle-in-a-Haystack task under varying context lengths and plot the inference-time scaling in Figure 5. We measure wall-clock runtime and accuracy at each context length, averaged over three runs. As shown in the figure, StateLM exhibits a linear relationship between runtime and input length while consistently maintaining a high accuracy above 90%. In contrast, the instruct baseline shows a similar linear trend only within its 128K limit, after which both runtime and accuracy collapse. These results confirm that the Pensieve design successfully transforms increasing context length into inference-time computation with $O(N)$ cost.

4.4 Ablation on Learning Mechanism

Since recent open-source models already support function calling, and our Pensieve paradigm is built upon this capability with deliberate training, a natural question arises: can models learn to manage their own context purely through agent-like prompting? To investigate this, we design a detailed system prompt that explicitly describes the context management process (see Figure 8 in Appendix) and provide the full tool specifications to Qwen3 instruct models (denoted as "Mprompt" models). We then evaluate their performance on the Document QA task from NovelQA and ∞Bench and compare them against our finetuned StateLM.

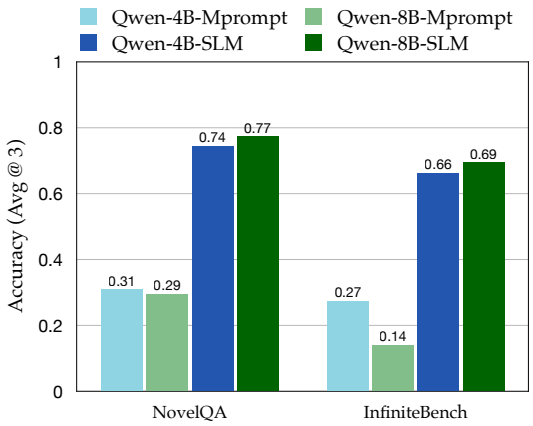

Figure 6: StateLM and Memory-prompt model performance.

The results in Figure 6 present this comparison. While prompting alone enables the instruct models to correctly answer about 30% of the questions, their performance is far below that of StateLM. This gap indicates that effective memory operations are difficult to acquire solely from prompt instructions and instead should be learned through deliberate training. In other words, the Pensieve paradigm benefits not only from tool availability but also from learned usage patterns that make context management robust and reliable.

5 Conclusion

In this work, we introduce the Pensieve paradigm, a state-aware approach where models actively manage their own memory through inference-time operations. Across both synthetic and realistic long-context QA tasks, StateLM consistently outperforms instruct models with fixed windows, sustaining high accuracy at million-token scales while preserving linear inference cost and minimizing context usage. These results mark a shift from passive sequence processing to models that reason over and reshape their own context, pointing to a scalable path for future foundation and agentic models.

Reproducibility Statement

We ensure reproducibility by providing detailed descriptions of our methodology, datasets, models, and training and evaluation configurations in Section 4, Appendix A, and Appendix B. The main code is included in the supplementary material, and we will release both the code and our training dataset publicly to facilitate the reproduction of our results.

Ethics Statement

This work complies with the ICLR Code of Ethics. Our research mainly develops a stateful language model that can efficiently manage its own context memory. Since training and evaluation rely solely on public datasets, the research does not involve human subjects or sensitive personal data. We carefully considered issues of fairness, privacy, security, and potential societal impact, and found no outstanding ethical concerns. Nevertheless, we acknowledge that such models are still at an early stage of development and should be used for research purposes only. We further emphasize that evaluations should be conducted in controlled environments with limited tool access.

References

Chenxin An, Shansan Gong, Ming Zhong, Xingjian Zhao, Mukai Li, Jun Zhang, Lingpeng Kong, and Xipeng Qiu. L-eval: Instituting standardized evaluation for long context language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14388–14411, 2024.

Deng Cai, Yan Wang, Huayang Li, Wai Lam, and Lemao Liu. Neural machine translation with monolingual translation memory. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 7307–7318, 2021.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory, 2025. URL https://arxiv.org/abs/2504.19413.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey, 2024.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. Ruler: What's the real context size of your long-context language models? In *First Conference on Language Modeling*.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan O Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training LLMs to reason and leverage search engines with reinforcement learning. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=Rwhi91ideu.

Andrej Karpathy. Context engineering is the delicate art and science of filling the context window... https://x.com/karpathy/status/1785482721150992423, Apr 2025. Accessed: 2025-06-25.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6769–6781, 2020.

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

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33: 9459–9474, 2020.

Huayang Li, Yixuan Su, Deng Cai, Yan Wang, and Lemao Liu. A survey on retrieval-augmented text generation, 2022.

Huayang Li, Pat Verga, Priyanka Sen, Bowen Yang, Vijay Viswanathan, Patrick Lewis, Taro Watanabe, and Yixuan Su. ALR^2: A retrieve-then-reason framework for long-context question answering, 2024.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. Search-o1: Agentic search-enhanced large reasoning models, 2025a.

Zhiyu Li, Shichao Song, Hanyu Wang, Simin Niu, Ding Chen, Jiawei Yang, Chenyang Xi, Huayi Lai, Jihao Zhao, Yezhaohui Wang, Junpeng Ren, Zehao Lin, Jiahao Huo, Tianyi Chen, Kai Chen, Kehang Li, Zhiqiang Yin, Qingchen Yu, Bo Tang, Hongkang Yang, Zhi-Qin John Xu, and Feiyu Xiong. Memos: An operating system for memory-augmented generation (mag) in large language models, 2025b. URL https://arxiv.org/abs/2505.22101.

Lin Long, Yichen He, Wentao Ye, Yiyuan Pan, Yuan Lin, Hang Li, Junbo Zhao, and Wei Li. Seeing, listening, remembering, and reasoning: A multimodal agent with long-term memory, 2025. URL https://arxiv.org/abs/2508.09736.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems, 2024. URL https://arxiv.org/abs/2310.08560.

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

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Richard James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. Replug: Retrieval-augmented black-box language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 8364–8377, 2024.

Cunxiang Wang, Ruoxi Ning, Boqi Pan, Tonghui Wu, Qipeng Guo, Cheng Deng, Guangsheng Bao, Xiangkun Hu, Zheng Zhang, Qian Wang, et al. Novelqa: Benchmarking question answering on documents exceeding 200k tokens. In *The Thirteenth International Conference on Learning Representations*.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. RECOMP: Improving retrieval-augmented LMs with context compression and selective augmentation. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=mlJLVigNHp.

Wujiang Xu, Kai Mei, Hang Gao, Juntao Tan, Zujie Liang, and Yongfeng Zhang. A-mem: Agentic memory for llm agents, 2025. URL https://arxiv.org/abs/2502.12110.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Hinrich Schütze, Volker Tresp, and Yunpu Ma. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning, 2025. URL https://arxiv.org/abs/2508.19828.

Howard Yen, Tianyu Gao, Minmin Hou, Ke Ding, Daniel Fleischer, Peter Izsak, Moshe Wasserblat, and Danqi Chen. Helmet: How to evaluate long-context language models effectively and thoroughly. *arXiv preprint arXiv:2410.02694*, 2024.

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, and Hao Zhou. Memagent: Reshaping long-context llm with multi-conv rl-based memory agent, 2025. URL https://arxiv.org/abs/2507.02259.

THE USE OF LARGE LANGUAGE MODELS

Large language models (LLMs) were applied in a restricted manner to support writing refinement. The authors supplied their draft text to the LLM, which suggested enhancements such as grammatical corrections, clearer formulations, and the elimination of informal expressions. LLMs were additionally consulted to generate possible paper titles. While the system proposed alternatives, the final title was independently selected and refined by the authors, and was not adopted verbatim from any single model output. Furthermore, LLMs were employed as auxiliary coding tools during implementation. They provided code completions and debugging advice; however, all final code, experimental setup, and validation were designed, implemented, and confirmed by the authors. It is important to emphasize that LLMs **WERE NOT** utilized for developing research ideas, designing experiments, or conducting literature review. All conceptual advances and experimental methodologies were entirely conceived and executed by the authors.

A TRAINING CONFIGURATIONS

We train our StateLM models using the following configurations. Each model is trained on a single machine equipped with 8×H20 GPUs.

Table 4: SFT Training Configuration

Parameter	**Value**
Global Batch Size	128
Learning Rate Scheduler	cosine
Learning Rate	1×10^{-5}
Warm-up Ratio	3×10^{-2}
Max Sequence Length	28000
Epochs	3
Parallelism Strategy	DeepSpeed ZeRO-3

B INFERENCE CONFIGURATIONS

In our experiments, we use the Qwen3 non-thinking inference mode for both StateLMs and instruct baselines. The thinking mode is excluded because it requires a 32K output space, which is computationally expensive and reduces the available context window.

For both StateLM and instruct baselines, we adopt the recommended sampling parameters as documented in the Qwen3 official guide. For the experiments, we repeat the sampling 3 times and compute the average score.

Table 5: Generation Configuration

Parameter	**Value**	**Used By**
Temperature	0.7	Both
Top_p	0.8	Both
Top_k	20	Both
Max Output Tokens	8192	Instruct
Max Context Tokens	120000	Instruct
Context Budget	32000	StateLM
Round Budget	150	StateLM
Max Rounds	200	StateLM

C LIMITATIONS

While our work makes progress toward building stateful language models, it has several limitations that open directions for future work.

Predefined tool set. In this work, the model is restricted to a fixed set of tools, including note-taking, search, and context pruning. While these are sufficient to demonstrate the benefits of state management, they constrain the flexibility of the system. A natural next step is to enable models to design, adapt, or refine their own tool sets dynamically based on the task and context.

Limited misbehavior analysis. In early experiments, we observed that the model sometimes overused the search tool or followed reasoning trajectories that deviated from human patterns, such as repeatedly mixing scanning with searching or arbitrarily discarding prior conclusions. Our data curation and fine-tuning pipeline alleviated these issues, resulting in more stable StateLMs. Nevertheless, a systematic study of potential misbehaviors and their broader implications lies beyond the scope of this work but remains an important direction for future investigation.

D PROMPTS

Prompt

You are an AI assistant for long-context processing with tools. Produce factually correct answers grounded in any attached text while conserving the context window. Describe your processing plan first, then proceed with the tools.

Figure 7: System prompt used in training and inference of our StateLMs.

Prompt

You are an AI assistant specialized in processing long-context tasks with tools. Produce factually accurate answers grounded in the provided context while minimizing context consumption.

Processing Strategy:
1. Check the size of the attached text:
- Long (> 8K tokens): build an index and process in chunks. For extremely long texts, increase the chunk size up to 12,000 tokens.
- Short (≤ 8K tokens): load the full text and answer directly.
- Empty: proceed with reasoning, using note-taking tools.
2. Analyze user's query and justify which processing mode is required to answer reliably and state that you plan to use that mode explicitly.
(a) Linear scan: Full-passage, sequential chunk-by-chunk reading (no details skipped), or (b) Keyword search: Keyword-based search to retrieve and inspect only the relevant chunks.
3. While reading, record relevant, accurate, and verifiable notes. Merge related notes as they grow to keep them concise.
4. Delete unnecessary context messages by their 'msg_id' to preserve context space, but do not delete everything or overuse the deletion tool. Deleted messages become stubs-do NOT restate their contents. Two required cases for deletions:
- After calling 'readChunk': once you have analyzed the chunk and optionally taken notes, immediately delete the chunk content using the 'msg_id' returned by the 'readChunk' tool.
- After calling 'note': delete the invoking assistant message using the 'msg_id(invoking_assistant)' returned by the 'note' tool result so the note-construction message is cleared.
5. Consult your notes and use relevant evidence to answer the user's query.
6. Call 'checkBudget' regularly to monitor usage and prevent overflows; adjust your strategy accordingly.

Describe your reasoning and processing plan before invoking any tools.

Figure 8: Memory management system prompt for the Mprompt models.

Prompt

You are a question classifier. Given a question, decide which reading mode is required to answer it reliably.

Reading Modes
1. **Full-passage reading** — The question requires comprehensive or sequential understanding of the whole document (e.g., summaries, global themes, narrative reasoning, cross-document counting, paraphrase detection spread across the text). Skimming or isolated lookup risks missing crucial context.
2. **Keyword-based search (multi-hop allowed)** — The question can be answered by searching targeted keywords/phrases and reading only the local passages they retrieve. Multi-hop is allowed if each hop is triggered by concrete terms (names, places, quoted strings, chapter titles).

Output Requirements
1. Provide a brief reasoning to support your final answer. 2. Format the final answer exactly as: \\boxed{1. Full-passage reading} or \\boxed{2. Keyword-based search}. 3. Do not answer the question itself. Do not include extra text beyond the brief reasoning and the boxed label.

Decision Heuristics
- Global / comprehensive signal → Mode 1. Prompts like "summarize," "overview," "main idea," "what happens," "count all occurrences," or questions that require following plot threads or dispersed evidence.
- Local / factual signal → Mode 2. Prompts asking for a specific name/date/place/term/definition typically resolvable by searching a few anchors.
- Paraphrase / approximate quote. If the question references a fuzzy or paraphrased sentence that may not be an exact match, prefer Mode 1 (to verify across context). Exact unique quotes tend toward Mode 2.
- Coreference / ellipsis. If resolving pronouns or elliptical references likely needs broad context, choose Mode 1.
- Tie-break rule. If both seem plausible, choose the least effort mode that still guarantees correctness. If uncertainty remains, default to Mode 1.

Example Classifications
{In-Context Examples}

—

Candidate to Evaluate
Now evaluate the following candidate question. Provide the brief reasoning and the final boxed label.

Question: [document content] {question}

Figure 9: Process-mode classification prompt for the LLM evaluation.

Prompt

Please act as an impartial judge and evaluate the correctness of a student's answer which attempts to answer the question based on a provided context. Although you are not given the context, you will be given a set of correct answers, and you need to assess the student's answer using the correct answers.

Below is your grading rubric:

Correctness:
- Score 0 (Incorrect): The answer does not agree with the provided correct answers at all.
- Score 1 (partly correct): Partly agree with one of the provided correct answers (for example, the question asks for a date and a person; the answer gets the date right but the person wrong).
- Score 2 (correct but not fully relevant): Fully agrees with one of the provided correct answers but mentions other completely irrelevant information. Note that extra details provided in the answer, even if not mentioned in the correct answers, should NOT be seen as irrelevant as long as they are relevant to the question to a reasonable extent.
- Score 3 (correct and relevant): Fully agrees with one of the provided correct answers and only provides information relevant to the question. Note that if the answer is longer than the correct answer, as long as everything in the answer is relevant to the question, it should still be given a score of 3. For example, if the correct answer is "the North Pole" and the answer is "They are headed for the North Pole", it should still be given a score of 3.

Now, read the following question, answer, and correct answers. First think step-by-step to provide your reasoning and assessment on the answer, then output your final score in the box. For example, " ..., so the correctness score is \\boxed{2}".

Question: {question}
Correct answers: {correct_answers}
Student Answer: {answer}

Figure 10: Prompt template used by LLM evaluation for Open-Ended questions. Prompt template is adapted from HELMET (Yen et al., 2024).