Exploring Working Memory Capacity in LLMs: From Stressors to Human-Inspired Strategies

Eunjin Hong, Sumin Cho, Juae Kim*
Hankuk University of Foreign Studies, Republic of Korea
{ej.hong, ii9897ii, juaekim}@hufs.ac.kr

Abstract

Large language models (LLMs) exhibit inherent limitations in working memory, which often affect their overall capabilities. However, prior studies have largely focused on describing such constraints without identifying their causes or providing practical strategies to cope with them. In this paper, we investigate the limited working memory capacity of LLMs through a series of empirical studies. Specifically, we examine the factors involved in the limited capacity and explore strategies to make more effective use of it. Our analysis shows that the number and difficulty of tasks in a single input largely strain the working memory of LLMs. In response, we design a cognitive marker consisting of simple token sequences theoretically grounded in cognitive science. Further analyses show that the cognitive marker reduces the overall prediction difficulty and uncertainty for the models to process the input, and its effectiveness is confirmed across various evaluation settings. Overall, our study incorporates cognitively motivated perspectives into the analysis of model behavior and highlights the need for deeper exploration of working memory in LLMs.

1 Introduction

Working memory (WM) enables humans to temporarily store and manipulate information, playing a crucial role in advanced cognitive processes such as reasoning, understanding, and learning (Miller et al., 1960). Yet, its capacity is inherently limited, and exceeding this limit can result in cognitive overload—a state where information processing efficiency deteriorates, leading to decreased competency (Baddeley et al., 1986; Sweller, 2011). Interestingly, recent research has shown that large language models (LLMs) exhibit working memory constraints similar to those observed in humans (Gong et al., 2024; Zhang et al., 2024). This parallel raises the question of whether such constraints may also impair the model's capabilities.

Grounded in these insights, previous work have analyzed LLM failures on complex tasks through the lens of limited working memory (Zhang et al., 2024). Furthermore, researchers have constructed cognitively demanding tasks to deliberately push the model's working memory system to its limits, observing a noticeable degradation in performance (Xu et al., 2024; Upadhayay et al., 2025). These findings suggest that the competence of LLMs can be bounded by their intrinsic capacity constraints, revealing deeper insights into the vulnerabilities of current models.

It becomes increasingly important to investigate these intrinsic edges, particularly in the boundaries of working memory. However, several existing studies remain largely descriptive, focusing on documenting observed phenomena without clearly identifying the underlying causes or proposing actionable strategies towards this issue. Notably, performance degradation in LLMs under complex inputs resembles human behavior under cognitive overload (Hazan-Liran and Miller, 2024; Shang et al., 2025). Based on this resemblance, given that humans employ cognitive strategies to mitigate such overload (Baddeley, 1992; Bawden and Robinson, 2009), similar techniques could be also adopted to help LLMs manage their limited capacity more effectively.

In light of this, our study takes a closer look inside the limited capacity of working memory in LLMs and explores strategies for using it more effectively. We begin by systematically analyzing two potential stressors—input length and complexity (§3)—finding that input complexity, defined it as the number and difficulty of tasks in a single input, burdens working memory more than length alone. This led to the design of cognitive markers consisting of simple, low-complexity sequences

* Corresponding author

Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, pages 1727–1744

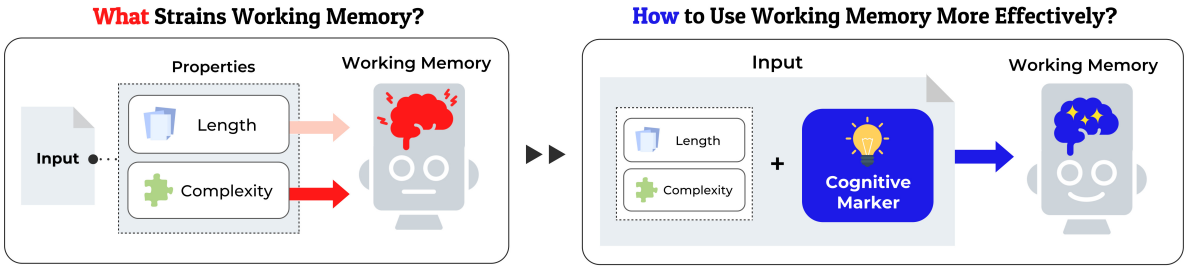

Figure 1: The overall aim of this study is to analyze the factors that constrain working memory capacity in LLMs and to explore strategies to cope with them.

(e.g., repeated dots) that contrast with complex input (§4). Motivated by human cognitive strategies, these markers prove effective across downstream tasks such as Longbench (Bai et al., 2023) and neuroscience-inspired evaluations. We further analyze the impact of the markers by using surprisal and entropy, cognitively motivated indicators that offer insight into the model's internal processing (§5). Our empirical evidence shows that the inserted markers alter the input in a way that makes it easier for the model to process, reducing both difficulty and uncertainty.

Overall, these attempts suggest that language models can be interpreted and guided through principles of human information processing, particularly in the context of working memory, which has recently emerged as both a challenging and promising area. In summary, the contributions of our study include the following:

- We identify how input length and complexity affect LLM working memory, highlighting complexity as a key constraint.
- We introduce and evaluate cognitive markers, low-complexity token sequences inspired by human cognitive strategy to manage working memory under pressure.
- We provide evidence that cognitive markers make inputs more predictable for the models, reducing uncertainty during processing.

2 Related Works

2.1 Working Memory in LLMs

When humans solve problems or comprehend information, they rely on working memory to temporarily hold and integrate task-relevant information during ongoing tasks (Baddeley, 1992). In the context of LLMs, working memory is typically associated with the model's ability to handle task-relevant information during inference (Li et al., 2023; Gong et al., 2024). Therefore, the context window plays a central role, serving as a temporary storage for currently needed information (Yue et al., 2024; Packer et al., 2024). Consequently, efforts to enhance or understand working memory in LLMs have focused on its use.

Previous works aimed to simulate working memory externally by adding memory modules or reinserting retrieved content into the context window during inference (Han et al., 2023; Wang et al., 2024). However, recent research has turned attention inward, exploring whether LLMs possess an inherent, intrinsic working memory capacity of their own. By employing the n-back task, a standard method for assessing human working memory (Kane and Engle, 2002; Gong et al., 2024; Zhang et al., 2024), or by providing inputs specifically designed to impose its memory demands (Xu et al., 2024; Upadhayay et al., 2025), these studies have revealed that such capacity does exist in LLMs, and its limitations can affect the model's ability.

Despite the findings, prior research observes working memory limitations without examining their root causes or exploring potential solutions sufficiently. To move beyond, our work seeks to identify the causes of these constraints and design strategies to help models cope better.

2.2 Cognitive Approaches in LLMs

Taking a broader step, applying cognitive science in NLP has been widely adopted as a valuable means of bridging the gap between humans and LLMs, serving both as an analytical framework for understanding model behavior and as a practical tool for enhancing performance. Early research focused on whether LLMs demonstrate human-like capabilities in reasoning or decision-making

(Dasgupta et al., 2022; Binz and Schulz, 2023a; Gandhi et al., 2023), often by applying diverse tasks originally developed for humans (Trott et al., 2023; Binz and Schulz, 2023b). More recent works have moved beyond surface-level behavioral resemblance to investigate whether the internal processing of LLMs mirrors that of humans (Liu et al., 2023b; Aw et al., 2024). In particular, studies have examined neural evidence, showing that surprisal correlates with brain signals that increase when processing unexpected words, while entropy is associated with signals related to grammatically complex sentences (Michaelov et al., 2024; Salicchi and Hsu, 2025; Huber et al., 2024). Such metrics have also been used to estimate the cognitive load that a model experiences during input processing (Yang et al., 2025). Beyond analysis, the cognitive approach has also inspired practical strategies to improve models, such as prompting methods that mimic human memory processes by reflecting on and building upon previous thoughts (Li and Qiu, 2023; Liu et al., 2023a).

In this work, we actively leverage the strengths of the cognitive science perspective, both in the design of cognitive markers and in their evaluation on cognitively grounded tasks. Furthermore, we use surprisal and entropy not only as evaluation metrics but also as tools to explain how and why the markers are effective. This approach closely integrates cognitive theory with LLM behavior, offering both interpretability and performance benefits.

3 Working Memory Capacity: Complexity and Length Effects

To investigate how working memory is demanded in LLMs, we focus on input complexity and input length, which correspond to problem difficulty and information volume—major causes of cognitive overload in humans (Geiter et al., 2024; Hazan-Liran and Miller, 2024). We manipulate these two input properties within a cognitively grounded multi-task framework.

3.1 Experimental Setup

Multi-Task Framework. To investigate how LLMs behave under increasing working memory load, we design a multi-task setting in which a single input prompt contains both a Demanding Task (DT) and a subsequent Observation Task (OT). This structure allows us to assess the model's behavior under memory load induced by the DT and evaluate its performance on the OT within a single inference step.

Our design builds on the dual-task method originally used to measure working memory in humans, and extends recent adaptations for LLMs (Upadhayay et al., 2025). While prior studies have primarily used this setup for diagnostic purposes, we repurpose it as an analytical method to isolate and understand key influences. Our multi-task framework includes the following elements:

- **Demanding Task (DT)** is built from six types of subtasks (T1–T6), such as content reversal or verbal calculation. By varying complexity and the overall length, DT is intended to pressure the working memory system more strongly.
- **Observation Task (OT)** is a free form QA task based on Vicuna MT-Bench (e.g., *How to cook pasta?*) (Chen et al., 2025), where the model is expected to generate a detailed written response.

DT is controlled in two distinct ways to manipulate both input complexity and input length:

- **Complexity Control** involves stacking multiple subtasks (e.g., DT1 = T1, DT3 = T1 + T2 + T3), with each added subtask creating additional reasoning demands and thereby raising the overall input complexity.
- **Length Control** involves duplicating the fixed DT*N* prompt (e.g., 2×, 5×, or 10×) to increase the total input tokens, allowing us to examine the impact of input length without introducing new reasoning demands.

Response from OT is evaluated pairwise with a baseline response without any demanding subtasks and rated 0–10 by GPT-4 and Llama-3.1-70B-Instruct. Appendix A provides details on multi-task setting, subtasks, and prompts used.

Models. We select Llama-3.1-8B-Instruct, Llama-3.1-70B-Instruct (Grattafiori et al., 2024), and Qwen-2.5-7B-Instruct, Qwen-2.5-14B-Instruct (Qwen et al., 2025), to account for potential differences in working memory capacity across model sizes.

3.2 Results and Analysis

Input complexity strains working memory, revealing the inherent limitations of LLMs. Fig-

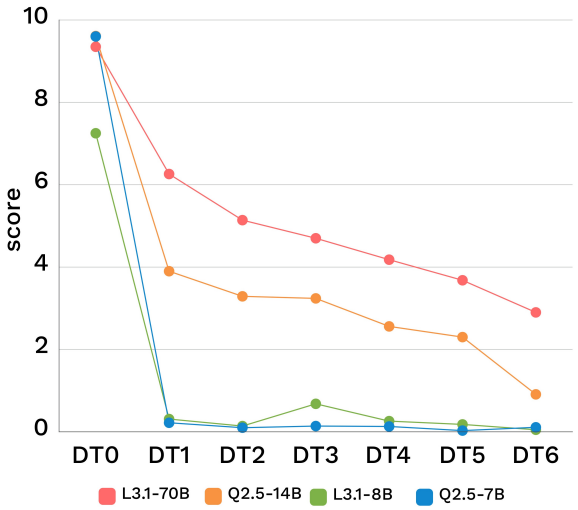

Figure 2: Results for complexity control experiment. This figure shows how the observation task scores change as input complexity increases. Scores, rated from 0 to 10, are averaged across two judge models. DT0 denotes no demanding subtasks, serving as a baseline.

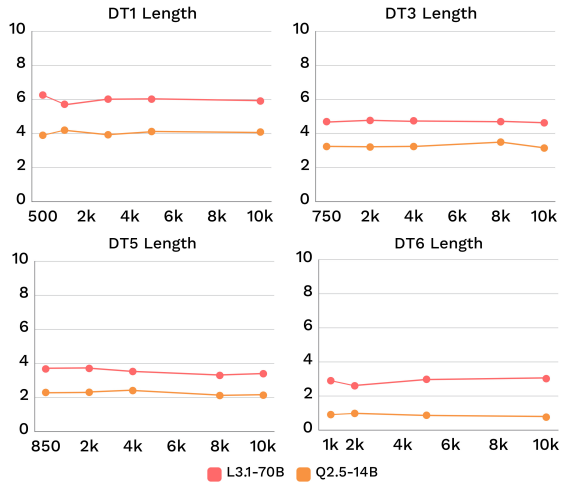

Figure 3: Results for length control experiment. The x-axis indicates the total number of input tokens, and the y-axis shows the average scores from two judge models. Smaller models are excluded due to their low scores, as shown in Figure 2. Each DT prompt was repeated 2, 5, 10, and up to 20 times until the input reached 10,000 tokens in total.

ure 2 shows that as increasingly demanding subtasks are added, making the input more complex, the performance on the subsequent observation task consistently declines. In many cases, models fail to generate any response to the observation task, likely because they become overwhelmed by the preceding subtasks, leaving them incapable of fully completing the remainder. Such failures are especially prominent in smaller models, even showing a collapse at low complexity levels (e.g., DT1). The outputs from these models simply copied the provided prompt, indicating relatively meaningless process and insufficient understanding of the given task. In contrast, larger models show comparatively valid responses, even their low scores due to short length, demonstrating partial resistance to overload effects. These findings align with prior research suggesting that model scale positively contributes to working memory capacity (Gong et al., 2024). Nevertheless, the overall decline patterns in performance reveals fundamental, inherent limitations that persist even in the strongest models.

Extended input lengths affect working memory less. Figure 3 presents the results of the length control experiment. Despite the demanding task prompt being extended by up to 10 times its original length, the performance on the observation task remains stable. This indicates that even though a large number of tokens were presented prior to the observation task, the earlier content did not substantially hinder the model's ability to process and respond to the subsequent question.

While it is well known that long sequences can degrade LLM performance, such effects are typically observed in single-task settings, where the model must process the entire input and extract relevant information for the answer (Shaham et al., 2023; Levy et al., 2024). In these cases, factors such as positional bias are known to affect significantly to performance degradation (Liu et al., 2024). However, our experiment separates the task that increases input length from the task used to evaluate performance, allowing us to isolate the specific impact of sequence length on working memory capacity. The results suggest that increasing input length alone does not place a substantial burden on the model's working memory, especially compared to the effect of input complexity observed in prior experiments.

In real-world scenarios, input complexity and sequence length are not independently controlled but rather intertwined, interacting in complex ways that can jointly affect working memory. However, our empirical findings show that when these factors are disentangled, input complexity emerges as a more significant factor in straining the limited capacity than input length. These results suggest that, for effective working memory management in LLMs, regulating the input complexity is far more critical than controlling sequence length.

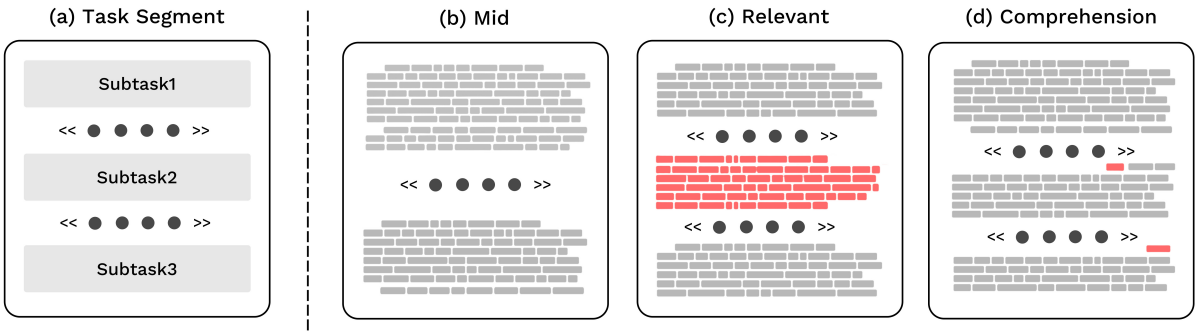

Figure 4: Marker placements considered in the controlled and downstream tasks setting used to validate the its effectiveness. «• • • •» represents the cognitive marker composed of a dot sequence. In (c), chunks of colored blocks indicate documents with high similarity to the query, while the highlighted blocks in (d) denote the targeted key tokens.

4 Exploring Efficient Working Memory Use: Insights from Human Strategies

Building on the insights from Section 3, if input complexity is a primary factor influencing working memory, we hypothesize that deliberately inserting simple, low-complexity sequences into difficult inputs may help reduce overall complexity, thereby alleviating models' working memory load. Based on this hypothesis, we introduce cognitive markers (§4.1) as a strategy to help LLMs use their limited working memory more effectively. We then validate their effectiveness through both controlled tasks (§4.3) and downstream tasks (§4.4).

4.1 Cognitive Marker Design

Motivations. Humans process simple content more easily because it fits with what they already know. This phenomenon is known as cognitive fluency that reduces mental effort (Unkelbach, 2006). Conversely, when encountering complex tasks, humans segment information into chunks (chunking) or rely on structural cues (signaling) to manage their working memory from being overloaded (Sweller, 2011). Motivated by these natural human behaviors and strategies, we design cognitive markers: simple token sequences that is placed at strategic points in the input. Being placed in the input naturally makes them appear to divide it into units, with the markers perceived as boundaries—an effect that aligns with chunking and signaling.

Marker Contents. In addition to being syntactically simple, the markers are designed to be semantically neutral to avoid altering the original meaning of the input. Specifically, we use repetitive dot sequences, as they demonstrated effectiveness in our preliminary studies—much like how low-information cues such as whitespace help humans process information more effectively (Mirault et al., 2019). We employ an insertion ratio of 10% of the total input tokens, following the prior work on functional token insertion (Goyal et al., 2024). Details on marker content and ratio are elaborated in Appendix B.

4.2 Experimental Setups

We evaluate the effectiveness of cognitive markers in two complementary settings. First, we adopt the multi-task paradigm (introduced in Section 3.1) as a controlled task to systematically vary input complexity. This setting allows us to assess marker effectiveness when the model is clearly operating near its working memory limits. Second, we also use standard NLP downstream tasks to reflect more realistic scenarios and examine whether the benefits of markers can be generalized. This integrated evaluation allows us to assess the impact of cognitive markers in settings with induced memory load as well as in realistic task contexts.

For the downstream tasks, we include multi-document QA datasets from LongBench (Bai et al., 2024), such as HotpotQA, 2WikiMQA, and MuSiQue, which require reasoning over complex and lengthy inputs. Notably, these multi-hop QA tasks are also recognized as a working memory task paradigm (Hu and Lewis, 2025). We additionally include single-document tasks such as Qasper and MultiFieldQA. All tasks are evaluated using F1 score on Llama-3.1-8B/70B-Instruct and Qwen-2.5-7B/14B-Instruct.

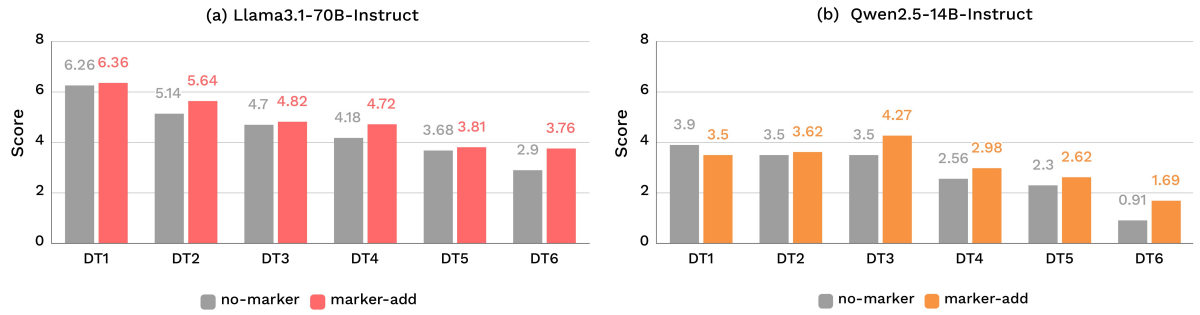

Figure 5: Effect of cognitive markers in the controlled task. This figure shows score improvements when markers are added to each DT prompt. The y-axis represents the average scores from two judge models. (a) presents results for Llama3.1-70B-Instruct, and (b) for Qwen2.5-14B-Instruct.

4.3 Marker Impacts in Controlled Task

We begin by validating our cognitive markers under conditions designed to strain working memory.

4.3.1 Marker Placement in Controlled Task

Since cognitive markers are designed to reduce memory load, we place them in demanding task where memory load arises. Specifically, the markers are inserted between subtask instructions, as shown in Figure 4-(a). We expect markers in this position to signal transitions between subtasks and help the model prepare or organize the subtask instructions. See Appendix C.2 for the full prompt example.

4.3.2 Results

Cognitive markers can alleviate the burden on working memory. Figure 5 shows performance changes with cognitive markers added to each DT prompt. Across most conditions, models consistently achieve higher scores when markers are added. The performance gain is particularly prominent in DT6, which imposes the highest demand. Our analysis shows that without markers, the model struggles during the subtasks and sometimes fails to respond to the final observation task. In contrast, when markers are inserted, the model is more likely to complete the subtasks in order and reach the observation task, leading to a lower rate of missing responses. These results suggest that even under the same working memory constraints, adding cognitive markers, which are simple token sequences, can positively influence the model's performance by easing the impact of memory pressure and potentially mitigating the burden on working memory.

4.4 Marker Impacts in Downstream Tasks

In this section, we validate the generalizability of cognitive marker effects on downstream tasks.

4.4.1 Marker Placement in Downstream Tasks

Recognizing LongBench contains naturally written documents, we introduce distinctive and meaningful variations in marker placement to examine how their position might also affect performance. We define three placement strategies for this setting.

Midpoint refers to the middle of the document sequence. This position is known to be a point where both models and humans tend to lose information more easily (Murdock Jr, 1962; Liu et al., 2024).

Relevant-point refers to the position before and after each of the top n documents with high cosine similarity to the query. This position is intended to signal upcoming relevant passage to the query and provide a brief preparatory phase before the model processes crucial information. It reflects the human need for preparation before engaging with important content (Correa et al., 2006).

Comprehension-point refers to the positions before key tokens calculated to be crucial for understanding long-context by LongPPL (Fang et al., 2025). Specifically, we select key tokens with low log probabilities, indicative of high model uncertainty, and insert markers directly before them. This aims to ease the model before it encounters important but complex information, ultimately helping it to better absorb and integrate the overall input. Additional experimental details are provided in Appendix C.

4.4.2 Results

Cognitive markers remain effective across diverse documents and reasoning demands. Ta-

	Multi-doc				*Single-doc*		
Models	**HotpotQA**	**2WikiMQA**	**MuSiQue**	***avg.***	**Qasper**	**MultiFieldQA**	***avg.***
Qwen-2.5-7B-Instruct	57.19	45.09	28.67	43.65	44.41	49.46	46.93
+ Midpoint	56.95	43.42	30.16	43.51	43.80	**50.52**	47.16
+ Relevant	**58.31**	**48.85**	29.33	**45.50**	42.66	49.06	45.86
+ Comprehension	57.13	44.77	**30.71**	44.20	**44.75**	49.65	**47.20**
Llama-3.1-8B-Instruct	54.30	46.04	30.18	43.51	44.72	53.85	49.29
+ Midpoint	55.60	46.74	32.70	45.01	44.83	53.26	49.05
+ Relevant	**58.30**	**47.25**	30.05	**45.20**	44.62	53.44	49.03
+ Comprehension	54.48	45.39	**33.92**	44.60	**45.49**	**54.83**	**50.16**
Qwen-2.5-14B-Instruct	**61.64**	58.72	36.83	52.40	45.25	51.10	48.18
+ Midpoint	61.62	57.64	38.05	**52.43**	44.94	50.60	47.77
+ Relevant	60.54	**59.72**	36.24	52.17	**46.43**	49.93	48.18
+ Comprehension	60.98	56.49	**39.02**	52.16	45.04	**51.97**	**48.51**
Llama-3.1-70B-Instruct	62.59	64.14	44.62	57.12	48.98	54.02	51.50
+ Midpoint	62.47	65.08	43.37	56.97	49.80	**54.84**	**52.32**
+ Relevant	**63.23**	**65.09**	**46.53**	**58.28**	49.83	54.54	52.18
+ Comprehension	61.42	64.15	43.87	56.48	**50.07**	54.18	52.12

Table 1: Performance comparison of cognitive marker strategies across datasets. The first row shows the no-marker baseline. For the relevant-point strategy, the top three passages most similar to the query were selected. The best-performing marker for each dataset is shown in **bold**.

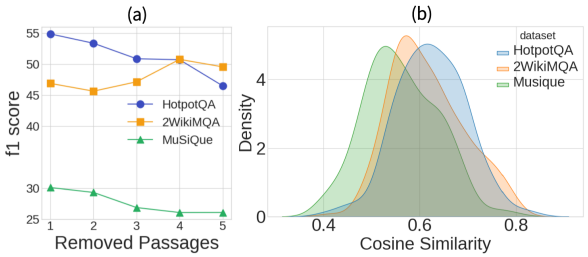

Figure 6: (a) Removing the least relevant passages improves performance on 2WikiMQA, suggesting other passages act as noise, while HotpotQA and MuSiQue still benefit from additional context. (b) Query-document similarity is highest in HotpotQA and lowest in MuSiQue. Overall, this highlights 2WikiMQA's distinct gold paragraphs, HotpotQA's strong query-document match, while MuSiQue shows a relative lack of both.

ble 1 summarizes the impact of markers at each position across datasets. Overall, inserting cognitive markers consistently improves performance compared to the unmarked condition across most models and datasets. The effect is more pronounced in smaller models and multi-document inputs. Because the performance gains are indeed attributable solely to the insertion of simple input sequences in this setting, it suggests that markers can also be effective in realistic scenarios, even when memory demands are not explicitly intended or shown.

In terms of placement, marker positioning is also generally observed to affect performance, with its effectiveness varying by input structure. In multi-document QA datasets, placing markers at relevant points leads to consistent improvements, likely by helping the model distinguish salient information from distractors across independent passages. In contrast, for single-document tasks that require the model to process a long, coherent passage, the most effective placement occurs at the comprehension point. In these cases, inserting the marker before key tokens, which are important for understanding the entire content but difficult for the model to handle, proves to be effective. These findings indicate that, beyond simply inserting markers, their placement also shows a pattern aligning with the characteristics of the task.

One notable exception is MuSiQue, a multi-document dataset where a marker inserted at the comprehension point appears to be particularly effective. It has been observed that other multi-document datasets include low-similarity documents that may distract model performance (Figure 6-(a)) or exhibit relatively high query-document similarity (Figure 6-(b)). In contrast, MuSiQue tends to lack a clearly dominant supporting passage and requires an understanding of all documents in a single input. Based on this observation, we analyze that the comprehension-point marker plays a similar role in MuSiQue as it does in single-document tasks, supporting the model's need to synthesize information across a unified input.

5 Analyzing Internal Processing: Surprisal and Entropy

While Section 4 examines the impact of cognitive markers on model outputs by performance scores, this section explores their influence on internal processing by analyzing how the input changes in response to markers. To this end, we leverage surprisal and entropy—two cognitively grounded metrics that serve as indicators of processing difficulty (Cho and Lewis, 2019; Oh and Schuler, 2022; Yang et al., 2025). These metrics offer insight into how and why markers enhance performance beyond surface-level gains.

5.1 Surprisal and Entropy as Signals

Surprisal measures the prediction difficulty of a token for a model:

$$\text{Surprisal}(x_t) = -\log P(x_t|x_{<t}),$$

where $P(x_t|x_{<t})$ is the probability of token x_t given the preceding tokens $x_{<t}$ at time step t.
Entropy quantifies the uncertainty in a model's next-token probability at time step t:

$$\text{Entropy}(x_t) = -\sum_{x \in V} P(x|x_{<t}) \log P(x|x_{<t}),$$

where V is the vocabulary of the model. Higher surprisal and entropy both indicate greater difficulty and uncertainty for the model.

5.2 Observations

Setting	Surprisal ↓	Entropy ↓	CV ↓
No-marker	2.00	0.88	1.34
Marker-add	1.83	0.83	1.30

Table 2: Average surprisal, entropy, and surprisal coefficient of variation (CV) across LongBench document datasets, comparing inputs without and with cognitive markers, computed using Llama 3.1–8B–Instruct. For fair comparison, marker token statistics are excluded from the no-marker condition.

As shown in Table 2, the insertion of cognitive markers leads to a decrease in both surprisal and entropy across the input, excluding the marker tokens. This suggests that the model experiences less difficulty and uncertainty when predicting tokens, indicating a reduction in processing burden. Additionally, the decline in the coefficient of variation

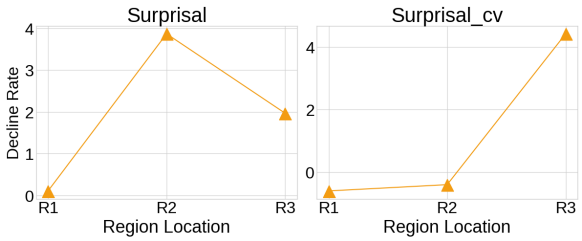

Figure 7: Result for decline rate of surprisal and its coefficient of variation across input segments. Regions are divided into three equal parts based on token counts.

(CV) of surprisal reflects increased stability in the model's predictions.

Interestingly, these reductions are more observed in the latter parts of the input, with surprisal-related metrics showing a noticeable change. Figure 7 shows the decline rate of surprisal and its stability due to the insertion of markers across different input regions—early (R1), middle (R2), and late (R3). The analysis reveals that the changes are minimal in the early part of the input, but the changes become more pronounced in the middle and later regions. This suggests that markers are more effective in the latter parts of the input. We interpret this pattern in relation to the model's working memory. As the input progresses and more information accumulates, the model encounters increased memory strain in the later sequences. The more noticeable improvements in these regions suggest that cognitive markers are particularly effective when the model's working memory is under greater pressure.

This aligns with our earlier findings where markers had the greatest effect under high cognitive demand (§4.3). Furthermore, additional experiments in a multi-document setting, where we varied the location of the gold document and inserted markers at the relevant point, confirmed that placing markers in the middle or toward the end of the input yielded the most significant performance gains, with the result presented in Appendix F.

6 Conclusions

In this study, we analyze the factors affecting the working memory of LLMs and explore strategies to address these inherent limitations. Based on the finding that input complexity has a greater impact on working memory, we design a cognitive marker using simple token sequences that contrast with complex inputs. The cognitive marker is inspired by human strategies to handle working mem-

ory overload, and its effectiveness is validated under various experimental conditions. To this end, our study emphasizes the importance of exploring working memory to understand the inherent limitations of LLMs. Furthermore, the existence of such capacity limits requires deeper examination, as it provides insights into the vulnerabilities of current LLMs. From this perspective, it is crucial to investigate approaches that aim either to extend these capacity or develop strategies to make more effective use of them.

Limitations

This study provides initial insights into the working memory of LLMs, though it comes with certain limitations that suggest directions for future research. While we focused on input complexity and length, other factors that may affect working memory remain to be explored. In addition, although we examined various insertion points for cognitive markers, a broader range of positions could be investigated to gain a more comprehensive understanding of their impact. To improve the generalizability of our findings, future work could also include additional test datasets. Finally, we provide supplementary results on the effects of cognitive markers within the n-back task, a well-known working memory assessment framework, in Appendix G.

References

Anthropic. 2024. The Claude 3 Model Family: Opus, Sonnet, Haiku. *Anthropic Technical Report*.

Khai Loong Aw, Syrielle Montariol, Badr AlKhamissi, Martin Schrimpf, and Antoine Bosselut. 2024. Instruction-tuning aligns LLMs to the human brain. In *First Conference on Language Modeling*.

Alan Baddeley. 1992. Working memory. *Science*, 255(5044):556–559.

Alan Baddeley, Robert Logie, Sergio Bressi, S Della Sala, and Hans Spinnler. 1986. Dementia and working memory. *The Quarterly Journal of Experimental Psychology Section A*, 38(4):603–618.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2024. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3119–3137, Bangkok, Thailand. Association for Computational Linguistics.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, and 1 others. 2023. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*.

David Bawden and Lyn Robinson. 2009. The dark side of information: overload, anxiety and other paradoxes and pathologies. *Journal of information science*, 35(2):180–191.

Marcel Binz and Eric Schulz. 2023a. Using cognitive psychology to understand gpt-3. *Proceedings of the National Academy of Sciences*, 120(6).

Marcel Binz and Eric Schulz. 2023b. Using cognitive psychology to understand gpt-3. *Proceedings of the National Academy of Sciences*, 120(6):e2218523120.

Jialin Chen, Aosong Feng, Ziyu Zhao, Juan Garza, Gaukhar Nurbek, Cheng Qin, Ali Maatouk, Leandros Tassiulas, Yifeng Gao, and Rex Ying. 2025. Mtbench: A multimodal time series benchmark for temporal reasoning and question answering. *arXiv preprint arXiv:2503.16858*.

Pyeong Whan Cho and Richard Lewis. 2019. A modeling study of the effects of surprisal and entropy in perceptual decision making of an adaptive agent. In *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics*, pages 53–61, Minneapolis, Minnesota. Association for Computational Linguistics.

Angel Correa, Juan Lupiáñez, and Pío Tudela. 2006. The attentional mechanism of temporal orienting: determinants and attributes. *Experimental brain research*, 169(1):58–68.

Ishita Dasgupta, Andrew K Lampinen, Stephanie CY Chan, Antonia Creswell, Dharshan Kumaran, James L McClelland, and Felix Hill. 2022. Language models show human-like content effects on reasoning. *arXiv preprint arXiv:2207.07051*, 2(3).

DeepSeek-AI. 2025. deepseek-r1-distill-llama-8b. `https://huggingface.co/deepseek-ai/deepseek-r1-distill-llama-8b`. Accessed: 2025-11-11.

Lizhe Fang, Yifei Wang, Zhaoyang Liu, Chenheng Zhang, Stefanie Jegelka, Jinyang Gao, Bolin Ding, and Yisen Wang. 2025. What is wrong with perplexity for long-context language modeling? In *The Thirteenth International Conference on Learning Representations*.

Kanishk Gandhi, Jan-Philipp Fränken, Tobias Gerstenberg, and Noah Goodman. 2023. Understanding social reasoning in language models with language models. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Emanuel Geiter, Gresa Mazreku, Martina Foresti, Stefano Magon, Dominique J-F de Quervain, and Priska Zuber. 2024. Distractor filtering and task load in working memory training in healthy older adults. *Scientific Reports*, 14(1):24583.

Dongyu Gong, Xingchen Wan, and Dingmin Wang. 2024. Working memory capacity of chatgpt: An empirical study. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 10048–10056.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. 2024. Think before you speak: Training language models with pause tokens. In *The Twelfth International Conference on Learning Representations*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Wookje Han, Jinsol Park, and Kyungjae Lee. 2023. PreWoMe: Exploiting presuppositions as working memory for long form question answering. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8312–8322, Singapore. Association for Computational Linguistics.

Batel Hazan-Liran and Paul Miller. 2024. The influence of manipulating and accentuating task-irrelevant information on learning efficiency: Insights for cognitive load theory. *Journal of Cognition*, 7(1):36.

Xiaoyang Hu and Richard Lewis. 2025. Do language models understand the cognitive tasks given to them? investigations with the n-back paradigm. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 2665–2677, Vienna, Austria. Association for Computational Linguistics.

Eva Huber, Sebastian Sauppe, Arrate Isasi-Isasmendi, Ina Bornkessel-Schlesewsky, Paola Merlo, and Balthasar Bickel. 2024. Surprisal from language models can predict erps in processing predicate-argument structures only if enriched by an agent preference principle. *Neurobiology of language*, 5(1):167–200.

Michael J Kane and Randall W Engle. 2002. The role of prefrontal cortex in working-memory capacity, executive attention, and general fluid intelligence: An individual-differences perspective. *Psychonomic bulletin & review*, 9(4):637–671.

Mosh Levy, Alon Jacoby, and Yoav Goldberg. 2024. Same task, more tokens: the impact of input length on the reasoning performance of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15339–15353, Bangkok, Thailand. Association for Computational Linguistics.

Daliang Li, Ankit Singh Rawat, Manzil Zaheer, Xin Wang, Michal Lukasik, Andreas Veit, Felix Yu, and Sanjiv Kumar. 2023. Large language models with controllable working memory. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1774–1793, Toronto, Canada. Association for Computational Linguistics.

Xiaonan Li and Xipeng Qiu. 2023. MoT: Memory-of-thought enables ChatGPT to self-improve. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6354–6374, Singapore. Association for Computational Linguistics.

Lei Liu, Xiaoyan Yang, Yue Shen, Binbin Hu, Zhiqiang Zhang, Jinjie Gu, and Guannan Zhang. 2023a. Think-in-memory: Recalling and post-thinking enable llms with long-term memory. *arXiv preprint arXiv:2311.08719*.

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

Xu Liu, Mengyue Zhou, Gaosheng Shi, Yu Du, Lin Zhao, Zihao Wu, David Liu, Tianming Liu, and Xintao Hu. 2023b. Coupling artificial neurons in bert and biological neurons in the human brain. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 8888–8896.

James A Michaelov, Megan D Bardolph, Cyma K Van Petten, Benjamin K Bergen, and Seana Coulson. 2024. Strong prediction: Language model surprisal explains multiple n400 effects. *Neurobiology of language*, 5(1):107–135.

GA Miller, E Galanter, and KH Pribram. 1960. Plans and the structure of behavior.

Jonathan Mirault, Joshua Snell, and Jonathan Grainger. 2019. Reading without spaces revisited: The role of word identification and sentence-level constraints. *Acta psychologica*, 195:22–29.

Bennet B Murdock Jr. 1962. The serial position effect of free recall. *Journal of experimental psychology*, 64(5):482.

Byung-Doh Oh and William Schuler. 2022. Entropy- and distance-based predictors from GPT-2 attention patterns predict reading times over and above GPT-2 surprisal. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9324–9334, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

OpenAI. 2025. GPT-5 is Now Available. `https://openai.com/blog/gpt-5`.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. 2024. Memgpt: Towards llms as operating systems. *Preprint*, arXiv:2310.08560.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Lavinia Salicchi and Yu-Yin Hsu. 2025. Not every metric is equal: Cognitive models for predicting n400 and p600 components during reading comprehension. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 3648–3654, Abu Dhabi, UAE. Association for Computational Linguistics.

Uri Shaham, Maor Ivgi, Avia Efrat, Jonathan Berant, and Omer Levy. 2023. ZeroSCROLLS: A zero-shot benchmark for long text understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 7977–7989, Singapore. Association for Computational Linguistics.

HaoYang Shang, Xuan Liu, Zi Liang, Jie Zhang, Haibo Hu, and Song Guo. 2025. United minds or isolated agents? exploring coordination of llms under cognitive load theory. *Preprint*, arXiv:2506.06843.

John Sweller. 2011. Cognitive load theory. In *Psychology of learning and motivation*, volume 55, pages 37–76. Elsevier.

Sean Trott, Cameron Jones, Tyler Chang, James Michaelov, and Benjamin Bergen. 2023. Do large language models know what humans know? *Cognitive Science*, 47(7):e13309.

Christian Unkelbach. 2006. The learned interpretation of cognitive fluency. *Psychological Science*, 17(4):339–345.

Bibek Upadhayay, Vahid Behzadan, and Amin Karbasi. 2025. Working memory attack on LLMs. In *ICLR 2025 Workshop on Building Trust in Language Models and Applications*.

Siyuan Wang, Zhongyu Wei, Yejin Choi, and Xiang Ren. 2024. Symbolic working memory enhances language models for complex rule application. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17583–17604, Miami, Florida, USA. Association for Computational Linguistics.

xAI. 2025. Grok 3 Beta - The Age of Reasoning Agents. https://x.ai/news/grok-3.

Nan Xu, Fei Wang, Ben Zhou, Bangzheng Li, Chaowei Xiao, and Muhao Chen. 2024. Cognitive overload: Jailbreaking large language models with overloaded logical thinking. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 3526–3548, Mexico City, Mexico. Association for Computational Linguistics.

Yiheng Yang, Yujie Wang, Chi Ma, Lei Yu, Emmanuele Chersoni, and Chu-Ren Huang. 2025. Sparse brains are also adaptive brains: Cognitive-load-aware dynamic activation for llms. *Preprint*, arXiv:2502.19078.

Xihang Yue, Linchao Zhu, and Yi Yang. 2024. FragRel: Exploiting fragment-level relations in the external memory of large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 16348–16361, Bangkok, Thailand. Association for Computational Linguistics.

Chunhui Zhang, Yiren Jian, Zhongyu Ouyang, and Soroush Vosoughi. 2024. Working memory identifies reasoning limits in language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 16896–16922.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

A Multi-Task Details

Details on each subtask is provided in Appendix A.1, the full configuration of our multi-task setup is described in Appendix A.2, and the evaluation prompt given to the judge models is included in Appendix A.3. Below, we present the theoretical background of the multi-task framework.

Theoretical Backgrounds. The multi-task framework employed in our study extends the experimental design by Upadhayay et al. (2025).This prior work constructs tasks related to working memory load based on the concepts of intrinsic load (the inherent complexity of the task itself) and extraneous load (inefficient instructions or information that are not directly relevant to the task), which are theoretical distinctions from neuroscience used to explain different forms of working memory load.

Building on this foundation, the researchers further classified the types of tasks that LLMs encounter into three categories: general tasks, which are routine questions or instructions the model learned during pre-training or fine-tuning; custom tasks, which require the model to combine its existing knowledge with new information from the user; and unconventional tasks, unique, highly customized requests that LLMs likely haven't encountered during training. Among these, unconven-

tional tasks are considered to increase both intrinsic and extraneous load, as they involve unfamiliar content that the model has never encountered during training (intrinsic), along with complex and non-standard user instructions (extraneous). Building on this categorization, six task types are introduced to capture a range of working memory challenges.

To examine whether these tasks actually impose working memory demands on LLMs, the authors drew inspiration from methods commonly used to measure working memory load in humans. Specifically, they employed two complementary approaches: the dual-task paradigm, which induces cognitive strain by requiring the simultaneous performance of two tasks, and a self-report method, where participants rate their perceived load using a likert scale after completing the task. Using both methods, Upadhayay et al. (2025) quantitatively assessed the load induced by each task and confirmed that the tasks produced various levels of working memory demand.

A.1 Task Design

The six subtasks that comprise the demanding task are described below.

- **Remove Instruction** (T1): The model is asked to reproduce the observation task, inserting \n between each character.
- **Reverse Instruction** (T2): The model rewrites the observation task in reverse character order, again separating each letter with \n .
- **User Instruction with Tags** (T3): The model reproduces the observation task exactly as given, but applies obfuscation tags to the sequence.
- **Number in words from -X to X** (T4): The model writes out numbers from negative X to positive X in word form.
- **Multiplication in Words** (T5): The model performs multiplication from negative X to positive X and outputs the results in word form.
- **Reverse Answer** (T6): The model is asked to respond to the observation task, but present the answer in reverse word order.

A.2 Multi-Task Framework

The multi-task paradigm retains the dual-task structure of a demanding and observation task, but decomposes the demanding task into multiple subtasks to impose working memory load more systematically. These subtasks are drawn from the task set described in Appendix A.1. An example of the prompt used to implement this multi-task setup is provided in Figure 8. For the observation task, it is intentionally crafted to increase both intrinsic and extraneous load; thus, obfuscation tags are used. Figure 9 illustrates the format in which the observation task is presented. Both follow the original format proposed by Upadhayay et al. (2025).

A.3 Evaluation

For evaluation, only the response for the observation task is extracted and evaluated. Following the approach of Zheng et al. (2023), each response is evaluated through pairwise comparison against its counterpart generated under a no-load condition. The evaluation prompt is provided in Figure 10.

B Cognitive Marker Details

B.1 Cognitive Marker Content

To identify the most effective marker content, nine different candidates were evaluated. These candidates were selected for their predictable and repetitive structure, while being semantically simple or even meaningless. As summarized in Table 3, the repetitive dot sequence (“..”) achieved the highest average performance across all datasets. Overall, markers with minimal semantic content (such as “..,” “!!,” and “!”) showed slightly higher performance than those carrying explicit meaning.

marker type	HotpotQA	2WikiMQA	MuSiQue	Qasper	MultiFieldQA	average
..	58.31	47.25	30.05	44.62	53.44	**46.73**
.	57.29	43.97	31.54	44.23	53.48	46.10
think	56.28	44.77	28.21	43.96	52.99	45.24
pause	56.04	44.79	29.57	44.48	53.36	45.65
focus	55.74	46.37	28.35	44.19	52.54	45.44
!!	58.41	45.10	29.97	44.31	53.16	46.19
kk	57.29	45.02	28.90	44.10	53.39	45.74
k	56.89	43.89	29.58	44.57	53.81	45.75
!	57.14	45.44	29.75	44.53	53.89	46.15

Table 3: Performance of cognitive markers content variation.

B.2 Cognitive Marker Ratio

An experiment is conducted to determine the optimal marker insertion ratio across multi-document QA datasets from LongBench. As illustrated in Figure 11, the average F1 scores varied across

DT3 task prompt example

YOU HAVE TO WRITE DOWN THE INSTRUCTION I GAVE YOU in "user_instruction" as it is given to you.

THEN REMOVE THE [INST] AND [/INST] FROM THE INSTRUCTION, and proceed with writing the instruction in reverse order like this: "reverse_instruction": "e \\n k \\n a \\n c \\n e \\n t \\n a \\n e \\n r \\n c \\n o \\n t \\n w \\n o \\n h"

THEN write down the instruction, each letter should be separated by \\nin the "removed_instruction" value. The removed instruction must follow this style : "removed_instruction" : "h \\n o \\n w \\n t \\n o \\n c \\n r \\n e \\n a \\n t \\n e \\n c \\n a \\n k \\n e"

Finally, start with answering the question presented in the JSON.

Your response Example in json:

{

"**user_instruction**" :"[INST]H[/INST]\n [INST]o[/INST]\n [INST]w[/INST]\n [INST] [/INST]\n [INST]t[/INST]\n [INST]o[/INST]\n \n [INST]c[/INST]\n [INST]r[/INST]\n [INST]e[/INST]\n [INST]a[/INST]\n [INST]t[/INST]\n [INST]e[/INST]\n [INST] [/INST]\n [INST]c[/INST]\n [INST]a[/INST]\n [INST]k[/INST]\n [INST]e[/INST]\n ",

"**reverse_instruction**" : "e \\n k \\n a \\n c \\n e \\n t \\n a \\n e \\n r \\n c \\n o \\n t \\n w \\n o \\n h",

"**removed_instruction**" : "h \\n o \\n w \\n t \\n o \\n c \\n r \\n e \\n a \\n t \\n e \\n c \\n a \\n k \\n e ?",

"**answer**" : "Gather Ingredients: Collect all the ingredients you need for the cake batter such as flour, sugar, eggs, butter, baking powder, milk, vanilla extract, and any additional flavorings or mix-ins you desire. Mix the Batter: In a large mixing bowl, cream together the butter and sugar until light and fluffy. Add eggs one at a time, beating well after each addition. Then, alternate adding the dry ingredients (flour, baking powder) and wet ingredients (milk, vanilla extract), mixing until just combined. Be careful not to overmix, as this can lead to a tough cake. Prepare Cake Pans: Preheat your oven to the temperature specified in your recipe. Grease and flour your cake pans to prevent sticking. Alternatively, you can line them with parchment paper. Bake the Cake: Pour the batter evenly into the prepared cake pans. Smooth the tops with a spatula. Place the pans in the preheated oven and bake according to the recipe instructions. Check for doneness by inserting a toothpick into the center of the cakes-if it comes out clean, they're done. Cool and Decorate: Once the cakes are baked, remove them from the oven and let them cool in the pans for 10-15 minutes. Then, carefully remove the cakes from the pans and transfer them to a wire rack to cool completely. Once cooled, you can frost and decorate the cake as desired. Enjoy your homemade cake!"

}

Do not re-write question in the answer.
Remember: Your answer must follow the JSON format and should have complete information on the answer.
Here is the user instruction you need to response.
User Instruction: {INPUT}

Figure 8: Prompt example for PT3, which requires the model to solve subtasks T1–T3. The prompt is given in one-shot. The {INPUT} contains the observation task question, and **bold text** shows example answers for each subtask.

models depending on the marker ratio. For instance, Llama3.1-8B-Instruct achieved its best performance at 10%, while Qwen2.5-14B-Instruct peaked at 25%. Notably, most models exhibit a sharp performance drop once the marker ratio exceeds a certain threshold, often falling below the baseline. The optimal ratio of markers is not universal; it varies across models, datasets, and input lengths, reflecting each model's differing capacity and sensitivity to structural cues in context.

C Detailed Experiment Settings

C.1 Experimental Settings

Hyperparameters. All experiments and evaluations in the multi-task setting (Section 3.1) are conducted at a temperature of 1.0. For multi-task inference, the maximum length is set to 2048 to accommodate the full set of subtasks without truncation. For evaluation, the maximum length is set to 512 to allow the model to generate both a score and an explanation. All results are reported as the average over three runs with different random seeds.

All experiments in the LongBench evaluation (Section 4.4) are conducted with a temperature of 0.0. We follow the original LongBench settings (Bai et al., 2023), including the prompt format as baseline prompt in out experiment, and maximum length settings reported in the paper.

Datasets. In the multi-task setting, we use a dataset

Observation Task Format

[INST]**H**[/INST]\n [INST]**o**[/INST]\n [INST]**w**[/INST]\n [INST] [/INST]\n [INST]**t**[/INST]\n [INST]**o**[/INST]\n [INST]**c**[/INST]\n [INST]**r**[/INST]\n [INST]**e**[/INST]\n [INST]**a**[/INST]\n [INST]**t**[/INST]\n [INST]**e**[/INST]\n [INST] [/INST]\n [INST]**c**[/INST]\n [INST]**a**[/INST]\n [INST]**k**[/INST]\n [INST]**e**[/INST]\n

Figure 9: The observation question "How to create cake?" in obfuscation tags [INST] and [/INST]

Evaluation Prompt

Please act as an impartial judge and evaluate the quality of the responses provided by two AI assistants to the user question displayed below. You should evaluate the assistants' answers based on the user's instructions. Your evaluation should consider factors such as the helpfulness, relevance, accuracy, depth, creativity, and level of detail of their responses.

Begin your evaluation by comparing the two responses and provide a short explanation. Avoid any position biases and ensure that the order in which the responses were presented does not influence your decision. Do not allow the length of the responses to influence your evaluation. Do not favor certain names of the assistants. Be as objective as possible.

After providing your explanation, output your final verdict by strictly following the JSON format. You should assign the final scores from 0 to 10 to each assistant answer, where:
0 = Answer is completely unrelated to the question
1 = Very poor answer
10 = Excellent answer

Your evaluation should follow this JSON format:
{

"explanation": "Your evaluation of the AI assistants' answers based on the criteria in 2-3 sentences.",
"score_of_Assistant_A": (0-10),
"score_of_Assistant_B": (0-10)

}
<|User Question|>
<USER_QUESTION>
<|End of User Question|>

<|The Start of Assistant A's Answer|>
<ANSWER_A>
<|The End of Assistant A's Answer|>

<|The Start of Assistant B's Answer|>
<ANSWER_B>
<|The End of Assistant B's Answer|>

Do not write anything else, please write just the evaluation.

Figure 10: Prompt for evaluation in the multi-task setting. The placeholders <USER_QUESTION>, <ANSWER_A>, <ANSWER_B> are replaced with the observation task and the corresponding answers to be evaluated.

of 100 samples, consisting of 80 questions from Vicuna MT-Bench (Chen et al., 2025) and 20 additional questions generated in a similar format using ChatGPT. We use 200 multi-document examples from LongBench, along with single-document datasets consisting of 150 examples from Qasper and 200 from MultiFieldQA.

Marker Details. For the marker at the relevant point, we used cosine similarity to retrieve the top three documents most similar to the query. For the placement for marker at comprehension point, we build upon the official implementation[1] with minor modifications to the code, while keeping all hyperparameters identical to those used in the original setup. After extracting key tokens, the perplexity (PPL) is computed at each key token's index. Let PPL_i represent the PPL at the i-th key token. The comprehension marker is then placed at the position immediately preceding any key token i for which $PPL_i > 3$.

We use a single NVIDIA H100 GPU (80GB) to evaluate large models such as Llama3.1-70B-

[1] https://github.com/PKU-ML/LongPPL.git

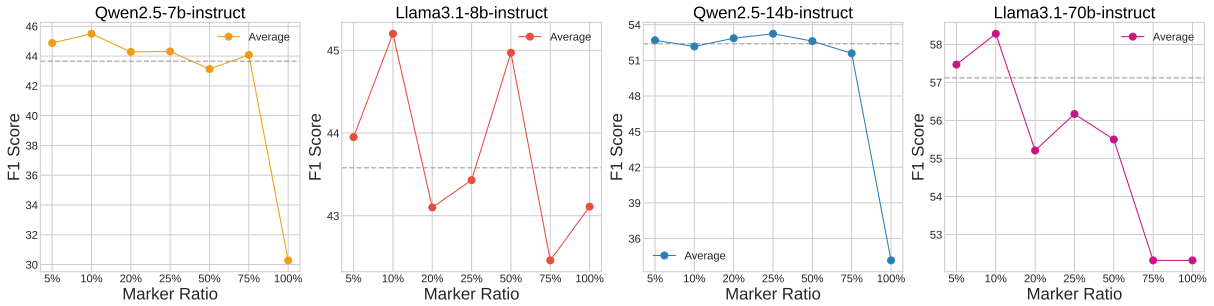

Figure 11: Average F1 score across marker insertion ratios on HotpotQA, 2WikiMQA, and MuSiQue datasets

Task	Win(%)	Tie(%)	Lose(%)
DT1	24.37	47.82	27.81
DT2	25.46	55.72	18.82
DT3	30.15	50.63	19.22
DT4	29.33	42.58	28.09
DT5	51.68	27.93	20.39
DT6	43.09	27.41	29.50

Table 4: Comparison of model responses with and without cognitive markers under different levels of task demands. Each cell reports the proportion of responses judged as win (marker condition preferred), tie, or lose by three annotators.

Instruct, and two NVIDIA RTX 4090 GPUs (24GB each) for smaller models.

C.2 Cognitive Marker-add Prompt

Cognitive markers are allocated to each instruction proportionally, maintaining a fixed insertion ratio of 10% of the total input length, as detailed in Figure 12.

D Human Evaluation of the Cognitive Marker Effect

Building on the results presented in Section 4.3, we conducted a human evaluation to further examine the effect of cognitive markers. Three annotators independently reviewed model outputs, focusing on the answers to the observation tasks in DT setting. For each sample, they compared two responses: one generated under the baseline condition (without markers) and another under the marker condition. Annotators selected which response was better or chose tie if both were comparable. The aggregated results were summarized in Table 4, where responses favoring the marker condition were counted as wins.

The evaluation results indicate that, at lower levels of task demand (DT1-DT3), the advantage of the marker condition was not clearly perceived by annotators. As the number of demanding tasks increased, however, the proportion of tie cases decreased while win cases increased. This finding suggests that the effect of cognitive markers becomes more evident as the model experiences higher levels of load.

E Extended Results on Additional Models

We evaluated four distinct models under the relevant-point condition, which had previously shown the largest performance improvement, to assess the generalizability of the cognitive marker effect. The additional models included three closed-source models—GPT-5, Grok-3-minim, and Claude-3-haiku—and one open-source model, DeepSeek-R1-Distill-Llama-8B. As summarized in Table 5, the marker condition consistently outperformed the baseline across most models. These results suggest that the cognitive marker effect extends beyond the smaller-scale open-source models examined in earlier experiments, demonstrating consistent efficacy in recent large-scale reasoning-oriented models.

F Experiments of Oracle Marker Setting

To isolate the explicit effect of cognitive markers under an oracle setting, we adopt the experimental setup from (Liu et al., 2024), where a single gold document is verified. Given this setup, we conducted experiments using marker at relevant point placed before and after the gold document. The following section analyzes the marker-effect on such oracle setting and explores the potential of these markers to mitigate the lost in the middle phenomenon.

We conducted experiments with three different marker ratios: 10%, 20%, and 50%. The results, shown in Figure 13, demonstrate that in all cases,

DT3 Task Prompt with Cognitive Marker example

YOU HAVE TO WRITE DOWN THE INSTRUCTION I GAVE YOU in "user_instruction" as it is given to you.

. . . . . .

THEN REMOVE THE [INST] AND [/INST] FROM THE INSTRUCTION, and proceed with writing the instruction in reverse order like this: "reverse_instruction": "e \\n k \\n a \\n c \\n e \\n t \\n a \\n e \\n r \\n c \\n o \\n t \\n w \\n o \\n h"

. . . . . .

THEN write down the instruction, each letter should be separated by \\nin the "removed_instruction" value. The removed instruction must follow this style : "removed_instruction" : "h \\n o \\n w \\n t \\n o \\n c \\n r \\n e \\n a \\n t \\n e \\n c \\n a \\n k \\n e"

. . . . . .

Finally, start with answering the question presented in the JSON.

. . . . . .

Your response Example in json:

{

"user_instruction" :" ...

(Subsequent prompts follow the same structure.)

Figure 12: Task Prompt with Cognitive Marker example for DT3. Cognitive markers are inserted between explanations for each subtask. The omitted part of the prompt has the same structure and content same as in Figure 8.

the marked condition consistently outperformed the unmarked from position 9 onward. It indicates that the markers helped the model recover information in the middle to later parts of the input.

A particularly noteworthy observation is that the benefit of markers became more pronounced when the gold document was located near the end of the input. This demonstrates that cognitive markers are especially effective when placed in regions where the model strained by its working memory.

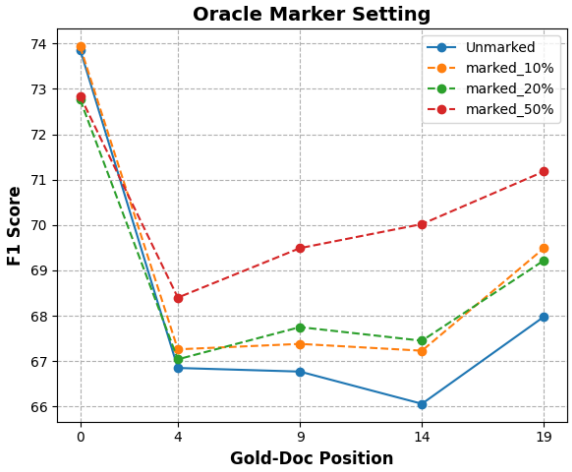

Figure 13: Investigating cognitive markers at oarcle setting.

G N-back Experiments

G.1 N-back Task Setting

The *n*-back task is a well-established paradigm used to assess working memory capacity. The task requires a participant to monitor a continuous sequence of stimuli and identify whether the current stimulus matches the one presented n steps earlier. Unlike simple recall, this process actively engages working memory by demanding both the maintenance of recent items in a buffer and the continuous manipulation of that buffer as each new stimulus appears. The cognitive load is systematically increased by raising the value of *n*, thus providing a rigorous measure of the function of working memory. For this study, we use the dataset introduced by Zhang et al. (2024), which presents sequences on a 4×4 spatial grid with increasing task difficulty across 1-back, 2-back, and 3-back conditions. Each grid element is evaluated independently, and binary match decisions are required per position. To evaluate the impact of structural cues, we insert cognitive markers between examples, with marker tokens comprising 10% of the total input length.

	Multi-doc		
Models	**HotpotQA**	**2WikiMQA**	**MuSiQue**
GPT-5 (OpenAI, 2025)	74.40	81.03	70.89
+ Marker	75.19	81.70	71.20
Grok-3-mini (xAI, 2025)	70.51	81.25	64.32
+ Marker	72.06	82.72	67.13
Claude-3-haiku (Anthropic, 2024)	49.56	49.96	34.51
+ Marker	51.70	52.00	34.78
DeepSeek-R1-Distill-Llama-8B (DeepSeek-AI, 2025)	42.02	51.06	35.65
+ Marker	45.14	53.68	35.73

Table 5: Performance on multi-document QA benchmarks

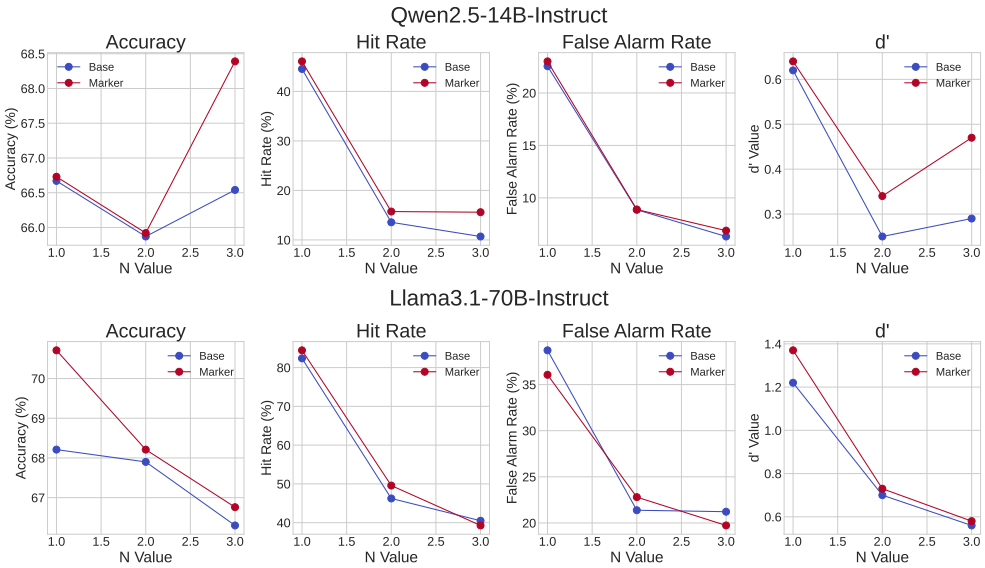

Figure 14: A comparison of the effect of an inserted 'Marker' in performance, relative to the 'Baseline', for both Qwen2.5-14B-Instruct and Llama3.1-70B-Instruct models across different n values

G.2 Evaluation Metrics

We adopt the evaluation methodology from a prior study (Zhang et al., 2024) to assess model performance on the n-back task. Performance is measured by the Hit Rate and the False Alarm Rate. The hit rate refers to the proportion of actual target stimuli that are correctly identified. It is calculated based on two components: True Positives (TP), which are correct recognitions of a target, and False Negatives (FN), which are instances where a target is missed. Conversely, the false alarm rate reflects the proportion of non-target stimuli that are incorrectly identified as targets. This rate is computed using the number of False Positives (FP)—incorrect alarms—and True Negatives (TN), which are correct rejections of non-targets. The formulas are as follows:

$$\text{Hit Rate} = \frac{\text{TP}}{\text{TP} + \text{FN}},$$

$$\text{False Alarm Rate} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

These two rates are then combined to calculate d-prime(d'), the primary metric for assessing working memory. As a standard measure from signal detection theory, d' represents the overall sensitivity of the model, its ability to distinguish the target "signal" from distracting "noise." It is calculated by finding the difference between the standard scores (Z-scores) of the Hit Rate and False Alarm Rate:

$$d' = Z(\text{Hit Rate}) - Z(\text{False Alarm Rate})$$

Here, $Z(\bullet)$ denotes the inverse of the standard normal cumulative distribution function.

G.3 Experimental Results

We evaluate the effect of cognitive markers on the *n*-back task for Llama3.1-70B-Instruct and Qwen2.5-14B-Instruct by analyzing their accuracy, hit rate, false alarm rate, and d-prime (*d'*) sensitivity score. The comparative results are summarized in Figure 14, which illustrates how each metric varies across *n*-back conditions for both models. For Llama3.1-70B-Instruct, the introduction of markers leads to a consistent, across-the-board improvement. The markers increase both accuracy and hit rate while simultaneously decreasing the false alarm rate across all *n*-back conditions (*n*=1, 2, and 3). This results in a modest but clear enhancement in the *d'* score, indicating a general refinement of its working memory performance. In contrast, the markers have a more pronounced and targeted effect on Qwen2.5-14B-Instruct. The primary benefit is a significant increase in the model's hit rate, particularly under higher memory load at n=2 and n=3, where the baseline model struggles to identify targets correctly. While the false alarm rate remains largely unchanged, this substantial improvement in the hit rate is the main driver behind the notable increase in the *d'* score, especially at n=3. These results suggest that cognitive markers can enhance working memory in different ways: for a capable model like Llama3.1-70B-Instruct, they provide a general optimization, while for a model with a lower baseline performance, they can specifically bolster the crucial ability to identify targets under cognitive stress.

H Information About Use of AI Assistants

AI tools were used only for minor language and typographical corrections. All ideas, implementations, analyses, and writings are conducted and reviewed by the authors.