


# A Cognitive Writing Perspective for Constrained Long-Form Text Generation

Kaiyang Wan<sup>1</sup>, Honglin Mu<sup>1</sup>, Rui Hao<sup>2</sup>, Haoran Luo<sup>3</sup>, Tianle Gu<sup>1</sup>, Xiuying Chen<sup>1\*</sup>,

<sup>1</sup>MBZUAI, <sup>2</sup>University of Chinese Academy of Sciences, <sup>3</sup>Nanyang Technological University  
{Kaiyang.Wan, Xiuying.Chen}@mbzuai.ac.ae

## Abstract

Like humans, Large Language Models (LLMs) struggle to generate high-quality long-form text that adheres to strict requirements in a single pass. This challenge is unsurprising, as successful human writing, according to the Cognitive Writing Theory, is a complex cognitive process involving iterative *planning*, *translating*, *reviewing*, and *monitoring*. Motivated by these cognitive principles, we aim to equip LLMs with human-like cognitive writing capabilities through CogWriter, a novel training-free framework that transforms LLM constrained long-form text generation into a systematic cognitive writing paradigm. Our framework consists of two key modules: (1) a Planning Agent that performs *hierarchical planning* to decompose the task, and (2) multiple Generation Agents that execute these plans in parallel. The system maintains quality via continuous *monitoring* and *reviewing* mechanisms, which evaluate outputs against specified requirements and trigger necessary revisions. CogWriter demonstrates exceptional performance on LongGenBench, a benchmark for complex constrained long-form text generation. Even when using Qwen-2.5-14B as its backbone, CogWriter surpasses GPT-4o by 22% in complex instruction completion accuracy while reliably generating texts exceeding 10,000 words. We hope this cognitive science-inspired approach provides a paradigm for LLM writing advancements:  CogWriter.

## 1 Introduction

LLMs like ChatGPT (Achiam et al., 2023) have begun to mirror human-like capabilities across diverse natural language processing tasks (Xi et al., 2023; Luo et al., 2024). From crafting concise summaries (Chen et al., 2025b, 2024a) to composing structured reports (Schmidgall et al., 2025; Wang et al., 2024d), these models can generate coherent text in a single pass (Rasheed et al., 2025;

Minaee et al., 2024) with a fluency that often rivals human writers. Recent advances have led to models with expanded context windows of up to 128K tokens (Pawar et al., 2024), theoretically enabling the generation of extensive documents (Bai et al., 2024). However, these models face significant challenges when tasked with generating constrained long-form text under complex constraints, such as following detailed instructions over 10,000 words (Wu et al., 2024a). This limitation poses a crucial barrier for applications requiring extended (Shi et al., 2024), well-structured content, including creative design proposals, technical documentation, and comprehensive research reports.

To understand the disparity between LLMs and human writers, we refer to Cognitive Writing Theory (Flower, 1981), which emphasizes how humans succeed in writing through a recursive activity that dynamically integrates multiple cognitive processes. As shown in the top part of Figure 1, these processes include *planning*, where writers establish high-level goals and develop structural outlines; *translating*, where writers transform abstract ideas into coherent text; and *reviewing*, where writers continuously evaluate and refine their generated content. Crucially, writers control these components through continuous *monitoring*, allowing them to assess and adjust text to better align with evolving objectives throughout the writing process.

Current LLMs excel at generating fluent text, effectively performing the *translating* function of converting internal token vectors into textual content. However, they fundamentally conflict with key cognitive principles in three ways, as shown in the bottom part of Figure 1: 1) They treat long-form text generation merely as an end-to-end task, overlooking the crucial hierarchical *planning* process that should guide content generation; 2) Their autoregressive architecture renders generated tokens as immutable context, preventing the *reviewing* and restructuring capabilities essential to hu-

\*Corresponding author.

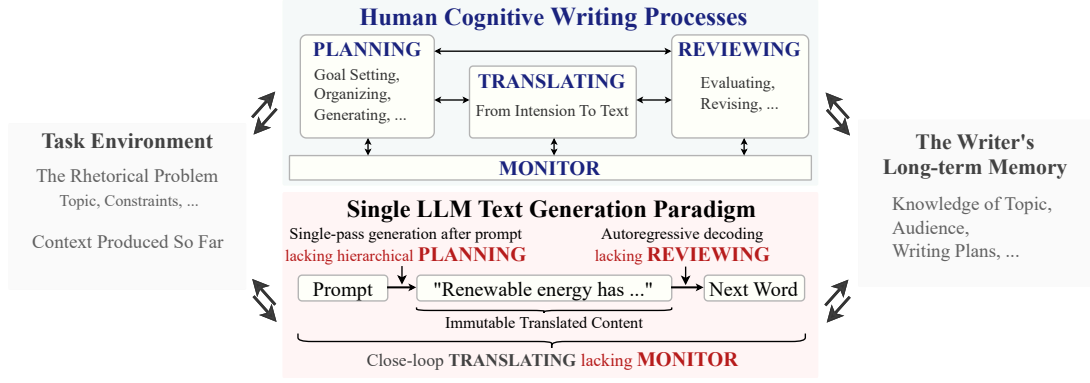


Figure 1: Comparison of human cognitive writing processes and single LLM text generation paradigm.

man writing; and 3) Unlike human writers who actively *monitor* their progress against both local and global objectives, LLMs lack explicit evaluation mechanisms, leading to potential divergence from intended goals in extended generations.

To address the limitations of single-pass generation, we introduce CogWriter, a novel training-free framework that aligns LLM-based text generation with cognitive writing paradigm. At its core, CogWriter employs a Planning Agent that decomposes complex requirements into manageable subtasks, providing explicit guidance for content generation. Based on sub-plans and the initial goal, multiple Generation Agents work in parallel to produce text segments, enabling both efficient generation and quality control that ensures consistent alignment with requirements. Crucially, both the planning and generation processes support iterative reviewing through feedback from external monitoring functions and LLM-based evaluation, thus enabling dynamic plan adjustment and content revision.

We evaluate CogWriter on LongGenBench-16K (Wu et al., 2024a), a benchmark designed to test a language model’s ability to generate instruction-aligned content about 16K tokens. Empirical results demonstrate that our paradigm is effective for both closed-source and open-source LLMs of various sizes. Specifically, even when using Qwen-2.5-14B as its backbone, CogWriter achieves a 22% higher instruction completion accuracy rate compared to GPT-4o, while reliably generating texts exceeding 10,000 words. These results demonstrate the effectiveness of cognitive science-inspired approaches in advancing LLM writing capabilities, particularly for complex constrained long-form text generation. We hope CogWriter’s systematic cognitive writing paradigm will inspire future research in LLM writing advancement.

Our contributions can be summarized as follows:

- We provide a cognitive science perspective on the shortcomings of single-pass LLM generation, highlighting how it diverges from established successful human writing processes.
- We propose CogWriter, a cognitive writing framework that equips LLMs with human writing strategies using multiple LLM-based agents with external monitoring functions.
- We demonstrate that CogWriter remarkably enhances LLMs’ ability to produce long-form, instruction-compliant texts without requiring additional training or reinforcement learning.

## 2 A Cognitive Writing Perspective

The challenge of constrained long-form text generation extends far beyond simply producing more words. Just as a novelist crafts an intricate narrative or an architect designs a towering structure, long text generation requires the coordination of multiple cognitive processes working together. Through the lens of cognitive writing theory, three fundamental processes emerge: hierarchical planning, continuous monitoring, and dynamic reviewing (Flower, 1981), as illustrated in Figure 1.

**Hierarchical Planning** Long-form writing requires a delicate cognitive balance between maintaining local coherence and global structure. Human writers cope with this constraint, as working memory cannot simultaneously retain every detail of a complex narrative (Kellogg, 2013). Skilled writers manage this limitation through hierarchical decomposition, systematically structuring the writing process into multiple levels (e.g., chapters, sections, and paragraphs). This approach enables

them to alternate between top-down thematic planning and bottom-up content development, ensuring alignment with high-level objectives while refining details (Hayes and Flower, 2016).

LLMs encounter a similar limitation: they generate text in a linear, autoregressive manner without an independent planning module to iteratively refine outlines or adapt strategies in real time (Xie et al., 2023). Consequently, their direct prompt-to-text generation process often struggles with complex, multi-threaded narratives. Without structured guidance, LLMs are prone to losing coherence over long spans, as their finite computational capacity quickly becomes overwhelmed (Hu et al., 2024).

**Continuous Monitoring** Effective planning in writing requires continuous oversight. Human writers naturally monitor their work, acting like their own editors. They pay attention to both small details—such as word choice and sentence flow—and the larger structure, ensuring the text maintains a clear theme and purpose (Kellogg, 2013).

In contrast, current mainstream LLMs generate text in a linear, close-loop manner, without the ability to review or refine their output. They lack a built-in system to check their progress against the intended goals, making it difficult to spot and correct issues during generation. Without external monitoring, LLMs struggle to detect when the content drifts off-topic, when the style becomes inconsistent, or when repetition occurs—problems that are especially common in extended long-form writing (Wang et al., 2024c; Ping et al., 2025).

**Dynamic Reviewing** While monitoring continuously tracks the writing process by detecting small errors, inconsistencies, or deviations, reviewing takes this feedback and applies it to make necessary adjustments, such as reorganizing content or improving logical flow. Human writers naturally engage in this iterative reviewing process, refining their work by revisiting earlier content and making adjustments (Bereiter and Scardamalia, 2013).

However, LLMs lack this ability due to their left-to-right, single-pass generation (Yao et al., 2023; Wu et al., 2024b). Without the ability to revisit or reorganize previous content, LLMs struggle with global revisions, such as restructuring sections or ensuring consistency across distant parts of the text (Bae and Kim, 2024; Cheng et al., 2024a,b). This absence of dynamic reviewing often results in long-form outputs with accumulated errors, inconsistencies, or redundant content.

### 3 Problem Formulation

Based on the analysis in Section 2, successfully generating long-form text requires addressing key deficiencies in current LLMs. We propose a new paradigm that equips LLMs with essential abilities to handle long, complex, and instruction-driven text generation. To achieve this, we formally define the constrained long-form text generation task, specifying the types of instructions and requirements the model must meet.

Following Wu et al. (2024a), we formally define constrained long-form generation as the task of generating a sequence of interrelated text segments  $\mathcal{D} = \{D_1, D_2, \dots, D_n\}$ , where each  $D_i$  represents a coherent unit of text that must satisfy certain constraints. Each segment  $D_i$  must achieve a target  $L$  words and adhere to a set of instructions  $\mathcal{T}$ . The instructions  $\mathcal{T}$  guide the generation process and are classified into three types: 1. **Single Instruction (SI)**: This instruction specifies content that must appear at exact, predefined positions. It is denoted as  $\mathcal{T}_S = \{T_{s1}, T_{s2}, \dots\}$ , where each  $T_{si}$  indicates specific content that must be placed in a precise position within the generated descriptions. 2. **Range Instruction (RI)**: This instruction specifies the content that must be included in each description within a designated range. It is represented as  $\mathcal{T}_R = \{T_i, T_{i+1}, \dots, T_{i+j}\}$ , ensuring that the specified content is sequentially assigned within the range  $[i, i + j]$ . 3. **Periodic Instruction (PI)**: This instruction mandates the periodic repetition of specific content at regular intervals. It is defined as  $\mathcal{T}_P = \{T_n, T_{2n}, \dots, T_{m \cdot n}\}$ , where  $n$  is the interval length and  $m$  specifies the number of repetitions. These instructions are unified into a comprehensive Check Set:  $\mathcal{T} = \{\mathcal{T}_S, \mathcal{T}_R, \mathcal{T}_P\}$ .

The versatility of this framework extends to various practical applications. For example, in architectural planning for a 100-floor building, Single Instructions determine specific facilities like a medical center on the 20th floor, Range Instructions define functional zones like corporate offices spanning floors 5-12, and Periodic Instructions maintain consistent amenities such as security checkpoints on every fifth floor. Each floor description must meet a target length of 200 words.

### 4 Methodology

Drawing upon our analysis of cognitive writing processes and the identified limitations of single-pass generation approaches, in this section, we propose

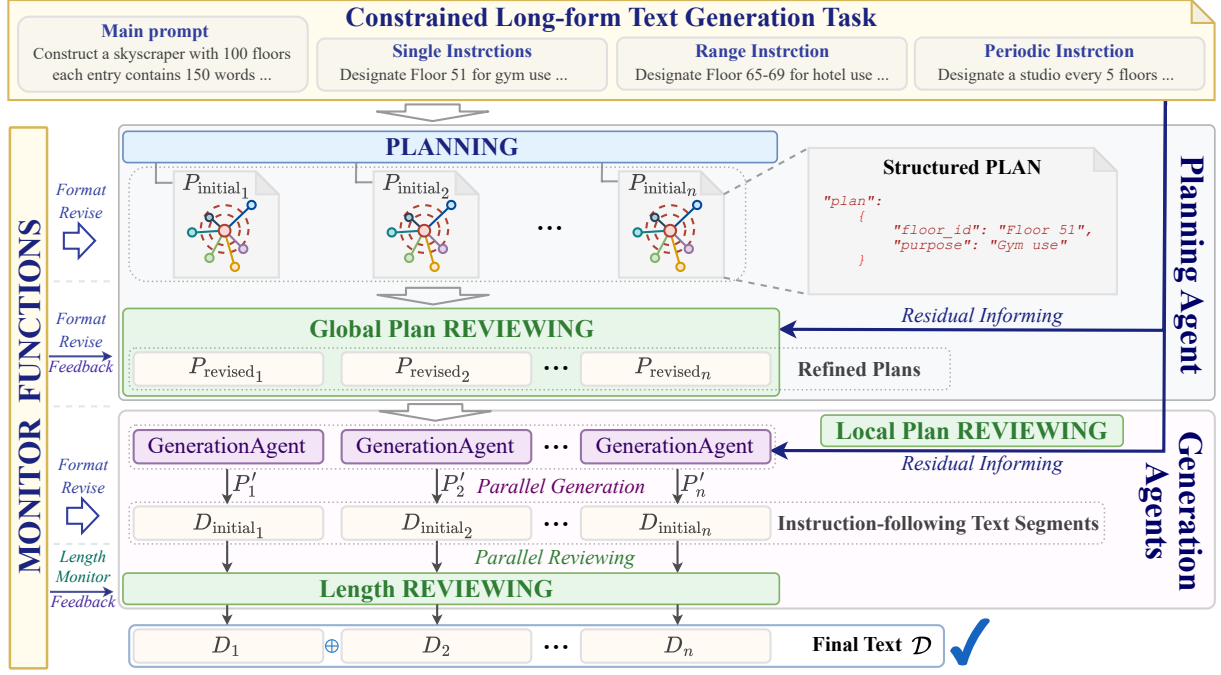


Figure 2: **Overview of the CogWriter Framework.** The framework consists of two key modules: the Planning Agent and the Generation Agents. The Planning Agent generates and refines an initial plan, guiding the structure and flow of the document. The Generation Agents collaborate to generate, revise, and finalize document segments, ensuring consistency in content and narrative coherence across the entire document.

CogWriter, a training-free framework that equips LLM with cognitive writing capabilities and enables LLMs to tackle complex constrained long-form generation with human-like strategic thinking.

#### 4.1 Framework Overview

As shown in Figure 2, CogWriter is designed to bridge the gap between current LLMs and human-like writing processes by integrating planning, monitoring, and reviewing mechanisms into the generation workflow. At its core, CogWriter employs a specialized Planning Agent that hierarchically decomposes the task and create structured plans, breaking down complex writing tasks into manageable components while maintaining their intricate relationships. Generation Agents execute these plans while monitoring mechanisms continuously evaluate the output to detect deviations in content, structure, or requirements. When issues are identified by monitor or LLM, a review process is triggered to revise and refine the output, ensuring overall coherence and adherence to instructions.

#### 4.2 Planning Agent

The Planning Agent serves as the strategic brain of the system. Similar to how an experienced writer begins with a detailed outline, this agent analyzes

task requirements and generates a structured initial plan  $\mathcal{P}_{\text{initial}}$  under strict format constraints:

$$\mathcal{P}_{\text{initial}} \leftarrow \text{GenerateInitialPlan}(p_{\text{plan}}),$$

where  $p_{\text{plan}}$  is the task-specific prompt incorporating instruction descriptions  $\mathcal{T}$ . The target plan is hierarchical, comprising unit plans:  $\mathcal{P}_{\text{initial}} = \{P_{\text{initial}_1}, \dots, P_{\text{initial}_n}\}$ .

After generating the initial plan, the *monitoring* mechanism supervises the process and relays signals to the *reviewing* mechanism for evaluation and validation. The reviewing mechanism evaluates the plan through two key checks: First, it verifies if the generated content satisfies the task-specific constraints  $\mathcal{T}$ . Second, it checks the plan’s structure for any syntax errors and applies necessary corrections. If any issues are detected, a revision process is triggered to refine the plan:

$$\mathcal{P}_{\text{revised}} \leftarrow \text{PlanRevise}(p_{\text{revise}}, \mathcal{P}_{\text{initial}}), \quad (1)$$

$$\mathcal{P} \leftarrow \text{FormatRevise}(\mathcal{P}_{\text{revised}}), \quad (2)$$

where  $p_{\text{revised}}$  includes the revision prompt for the task instructions  $\mathcal{T}$ . This iterative refinement ensures that the final plan is not only of high quality but also optimally structured to guide robust and effective content generation.



---

**Algorithm 1** CogWriter Algorithm

---

**Require:** Prompts  $p_*$  including task instruction  $\mathcal{T}$

**Ensure:** Final text  $\mathcal{D} = \{D_1, \dots, D_n\}$

```
1: function PLANNINGAGENT( $p_*$ )
2:    $\mathcal{P}_{\text{initial}} \leftarrow \text{GenerateInitialPlan}(p_{\text{plan}})$ 
3:    $\mathcal{P}_{\text{revised}} \leftarrow \text{PlanRevise}(p_{\text{revise}}, \mathcal{P}_{\text{initial}})$ 
4:    $\mathcal{P} \leftarrow \text{FormatRevise}(\mathcal{P}_{\text{revised}})$ 
5:   return  $\mathcal{P}$ 
6: end function
7: function GENERATIONAGENTS( $p_*, \mathcal{P}$ )
8:   Initialize empty document collection  $\mathcal{D}$ 
9:   for each  $P_i$  in  $\mathcal{P}$  do
10:     $P'_i \leftarrow \text{PlanAdjust}(p_{\text{adjust}_i}, P_i)$ 
11:     $D_{\text{initial}_i} \leftarrow \text{Generate}(p_{\text{write}}, P'_i)$ 
12:     $D_i \leftarrow \text{LengthRevise}(p_{\text{length}}, D_{\text{initial}_i})$ 
13:   end for
14:    $\mathcal{D} \leftarrow \mathcal{D} \cup D_i$ 
15:   return  $\mathcal{D}$ 
16: end function
```

---

### 4.3 Generation Agents

Once the global plan  $\mathcal{P} = \{P_1, \dots, P_n\}$  is finalized by the Planning Agent, multiple Generation Agents take over, each responsible for generating content for a specific description task  $D_i$ . The process begins with validating and refining the local plan  $P_i$ , through *monitoring* and *reviewing* similar to the Planning Agent to ensure it aligns with the instruction requirements  $\mathcal{T}$ . Concretely, if discrepancies are detected, adjustments are applied to update the plan, as shown in the following equation:

$$P'_i \leftarrow \text{PlanAdjust}(p_{\text{adjust}_i}, P_i), \quad (3)$$

where  $p_{\text{adjust}_i}$  encompasses the specialized prompt designed for reviewing each local plan  $P_i$  against the residual informing from  $\mathcal{T}$ .

Upon validation of  $P'_i$ , the agent generates content by executing the plan:

$$D_{\text{initial}_i} \leftarrow \text{Generate}(p_{\text{write}}, P'_i), \quad (4)$$

where  $p_{\text{write}}$  is the prompt to generate content following the guidance of the plan  $P'_i$ . Based on our preliminary study, this process generally produces content that meets most instruction criteria. However, length constraints may still require further refinement due to the limitations of most current LLMs in controlling output length precisely. To address this, a revision function adjusts the content to meet the specified length  $L$ :

$$D_i \leftarrow \text{LengthRevise}(p_{\text{length}}, D_{\text{initial}_i}), \quad (5)$$

where  $p_{\text{length}}$  is the prompt used to adjust the content length to  $L$  by expanding or compressing the generated text while preserving key details, semantic integrity, and overall coherence.

By following this process, each segment  $D_i$  seamlessly integrates with the overall narrative structure, ensuring both local coherence and global thematic consistency.

## 5 Experiments

### 5.1 Experimental Setup

**Dataset** We evaluated CogWriter using LongGenBench-16K (Wu et al., 2024a), a benchmark specifically designed for assessing a model’s complex constrained long-form text generation capabilities. The dataset features four scenarios, each requiring approximately 16,000 tokens: (1) Diary Writing and (2) Menu Design assess temporal consistency by requiring coherent content organization across weeks of a year, while (3) Skyscraper Design and (4) Urban Planning evaluate spatial reasoning through detailed facility arrangements across floors or city blocks. The benchmark includes 400 test instances, with 100 instances per scenario. Each scenario involves three instruction types (defined in Section 3): single instructions, range instructions, and periodic instructions. For temporal tasks, Diary Writing and Menu Design require at least 200 words per weekly entry, totaling 10,400 words (52 weeks  $\times$  200 words). For spatial tasks, Skyscraper Design and Urban Planning mandate 15,000 words (100 units  $\times$  150 words).

**Evaluation Metrics** We evaluate model performance using three key metrics from LongGenBench. *Main Task Completion Rate* (Comp. Rate) assesses whether all designated subtasks are completed in sequence (e.g., generating entries for every week in a diary without omissions). *Instruction Following Accuracy* measures adherence to single (Acc. Once), range (Acc. Range), and periodic (Acc. Periodic) instructions, with their average reported as Avg. Acc. We utilize the official evaluation scripts to ensure consistency with reported benchmarks. Additionally, we track *Word Count*, ensuring a minimum average threshold of 12,700 words to meet the combined task requirements.

**Experimental Setup** We evaluate our approach across three categories of models and methods. First, we establish baseline performance using several single-pass generation models from the official

Model	Comp. Rate	Acc. Once	Acc. Range	Acc. Periodic	Avg. Acc.	Words (Req. $\geq 12700$ )
LongWriter-Llama3.1-8B	0.46	0.36	0.56	0.17	0.36	11036
Llama-3.1-8B-Instruct	0.94	0.36	0.49	0.17	0.34	8804
Llama-3.1-70B-Instruct	0.79	0.50	0.51	0.18	0.39	8055
Mixtral-8x7B-Instruct-v0.1	0.83	0.42	0.45	0.24	0.37	8113
Qwen-2-72B-Instruct	0.94	0.42	0.44	0.14	0.33	8013
GPT-4o-mini	0.97	0.54	0.48	0.16	0.39	8940
+ SELF-REFINE	0.84	0.57	0.32	0.20	0.36	8154
+ CoT	0.93	0.59	0.48	0.18	0.42	10137
+ CogWriter (Ours)	<b>1.00</b> ( $\uparrow 0.03$ )	<b>0.74</b> ( $\uparrow 0.20$ )	<b>0.61</b> ( $\uparrow 0.13$ )	<b>0.31</b> ( $\uparrow 0.15$ )	<b>0.55</b> ( $\uparrow 0.16$ )	<b>12484</b> ( $\uparrow 3544$ )
Qwen-2.5-14B-Instruct	0.29	0.53	0.54	0.24	0.44	1817
+ SELF-REFINE	0.17	0.45	0.63	0.21	0.43	1122
+ CoT	0.30	0.46	0.20	0.16	0.27	1619
+ CogWriter (Ours)	<b>0.79</b> ( $\uparrow 0.51$ )	<b>0.70</b> ( $\uparrow 0.17$ )	<b>0.65</b> ( $\uparrow 0.11$ )	<b>0.47</b> ( $\uparrow 0.23$ )	<b>0.61</b> ( $\uparrow 0.17$ )	<b>10091</b> ( $\uparrow 8274$ )
Llama-3.3-70B-Instruct	0.99	0.59	0.63	0.21	0.48	9431
+ SELF-REFINE	0.93	0.59	0.64	0.28	0.50	8491
+ CoT	1.00	0.62	0.62	0.21	0.48	9302
+ CogWriter (Ours)	<b>1.00</b> ( $\uparrow 0.01$ )	<b>0.76</b> ( $\uparrow 0.17$ )	<b>0.79</b> ( $\uparrow 0.16$ )	<b>0.55</b> ( $\uparrow 0.34$ )	<b>0.70</b> ( $\uparrow 0.22$ )	<b>12051</b> ( $\uparrow 2620$ )
GPT-4o	0.63	0.63	0.60	0.17	0.47	9055
+ SELF-REFINE	0.66	0.67	0.62	0.33	0.54	4641
+ CoT	0.40	0.58	0.63	0.32	0.51	4482
+ CogWriter (Ours)	<b>0.91</b> ( $\uparrow 0.29$ )	<b>0.80</b> ( $\uparrow 0.17$ )	<b>0.76</b> ( $\uparrow 0.16$ )	<b>0.67</b> ( $\uparrow 0.50$ )	<b>0.74</b> ( $\uparrow 0.27$ )	<b>11618</b> ( $\uparrow 2563$ )

Table 1: Model Performance Comparison and the Improvement Brought by CogWriter (values in parentheses indicate the improvement relative to the base model).

LongGenBench repository, including LongWriter-Llama3.1-8B (Bai et al., 2024), Llama-3.1-8B-Instruct, Mixtral-8x7B-Instruct-v0.1 (Jiang et al., 2023), Llama-3.1-70B (Grattafiori and et al, 2024), Qwen-2-72B-Instruct (Qwen et al., 2025), as well as GPT-4o and GPT-4o-mini. Second, we compare against two prominent enhancement methods: SELF-REFINE (Madaan et al., 2023) and Chain-of-Thought (CoT) prompting (Wei et al., 2022). These methods are applied to four representative foundation models to ensure comprehensive evaluation across different model capabilities and architectures. Finally, to demonstrate the effectiveness of our CogWriter paradigm, we apply it to the same four foundation models: GPT-4o-mini-2024-07-18, GPT-4o-2024-08-06, Qwen-2.5-14B (Team, 2024), and Llama-3.3-70B (Touvron et al., 2024). This selection encompasses closed-source and open-source models with varying parameter scales, enabling us to evaluate CogWriter’s generalizability. For fair comparison, we implement SELF-REFINE and CoT baselines on these same models alongside our proposed framework.

**Implementation Details** We deployed our experiments across local computational resources and cloud-based APIs. For open-source models (Qwen-2.5-14B and Llama-3.3-70B), we leveraged vLLM (Kwon et al., 2023) for its efficient inference

acceleration while maintaining the default temperature and sampling parameters as specified in the official Hugging Face implementations. These experiments were conducted on 4 NVIDIA A100-SXM4-80GB GPUs running CUDA 12.8. For closed-source models (GPT-4o and GPT-4o-mini), we utilized their respective official API.

## 5.2 Main Results

Table 1 highlights the main performance outcomes of our experiments. Firstly, our results reveal that *LongWriter-Llama3.1-8B*, despite being specifically designed and trained from *Llama-3.1-8B-Instruct* for long-form generation, struggles considerably, achieving only a 0.46 completion rate. Similarly, even advanced models with substantial parameter counts, such as Llama-3.1-70B-Instruct and Qwen-2-72B-Instruct, fail to reach the target length of 12,700 tokens in their generated outputs. Secondly, *alternative enhancement methods also exhibit limited effectiveness*. Chain-of-Thought prompting results in a modest improvement in instruction-following accuracy (from 0.39 to 0.42 using GPT-4o-mini), while SELF-REFINE achieves reasonable completion rates. However, both approaches fall short in meeting length requirements and maintaining instruction adherence.

In contrast, CogWriter demonstrates remarkable improvements across all evaluation metrics.

Model	Comp. Rate	Acc. Once	Acc. Range	Acc. Periodic	Avg. Acc.	Words (Req. $\geq 12700$ )
GPT-4o-mini + CogWriter	<b>1.00</b>	<b>0.74</b>	<b>0.61</b>	0.31	<b>0.55</b>	<b>12484</b>
- w/o PlanRevise	0.99	0.73	0.45	0.33	0.50	12472
- w/o PlanAdjust	<b>1.00</b>	0.63	0.46	0.27	0.45	12341
- w/o LengthReview	<b>1.00</b>	0.73	0.61	0.30	0.54	11549

Table 2: Ablation study on the effectiveness of CogWriter’s key components using GPT-4o-mini as the base model.

When using Qwen-2.5-14B-Instruct as its backbone, it boosts the completion rate by 0.51 and improves average accuracy by 0.17. For Llama-3.3-70B-Instruct and GPT-4o, CogWriter achieves near-perfect completion rates while consistently enhancing instruction-following accuracy, excelling at handling complex periodic instructions.

Method	Plan	Decomp.	Monit.	Rev.
Human Writer	✓	✓	✓	✓
CoT	✓	×	×	×
SELF-REFINE	×	×	×	✓
Single-pass LLMs	×	×	×	×
CogWriter	✓	✓	✓	✓

Table 3: Comparison of different writing approaches. *Plan*: planning the writing structure; *Decomp.*: decomposing complex tasks into manageable components; *Monit.*: monitoring progress during generation; *Rev.*: reviewing and refining generated content.

**Advantages of Cognitive Structure** We provide a comparison of the cognitive capabilities of the baselines, our proposed paradigm, and human writers in Table 3, to analyze the strong performance of our approach. It can be seen that human writers naturally employ all four cognitive processes—planning, decomposition, monitoring, and reviewing—while existing computational methods implement only subsets of these capabilities. CoT primarily focuses on planning, and SELF-REFINE incorporates only reviewing. In contrast, CogWriter mirrors the complete human writing process by integrating all four capabilities, which may help explain its superior effectiveness in complex long-form generation tasks.

**Correlation with Model Internal Ability** We next discuss the relationship between performance improvements and the model’s capabilities. When applying our framework to Llama-3.1-8B-Instruct, we observed a clear limitation: the model struggled to generate coherent and structured plans essential for CogWriter’s method. In contrast, for stronger LLMs such as GPT-4o, CogWriter achieved sig-

nificant improvements, including a 0.29 increase in completion rate and a 0.50 increase in periodic instruction accuracy. This suggests that models with more advanced internal cognitive abilities are better at utilizing CogWriter’s coordination of cognitive processes, while weaker models, lacking robust instruction-following skills, fail to fully replicate this process. This limitation shows that CogWriter’s effectiveness depends on the model’s internal abilities, with advancing LLMs enabling more human-like reasoning and problem-solving.

## 6 Discussion

**Ablation Study** We conduct an ablation study to evaluate the impact of different components in our proposed CogWriter framework, as shown in Table 2. Removing the *PlanRevise* module resulted in a noticeable performance drop across key metrics, with the average accuracy decreasing from 0.55 to 0.50. This demonstrates that refining the initial plan through iterative revisions is crucial for maintaining effective task decomposition and alignment with task-specific constraints. Disabling the *PlanAdjust* mechanism further impacted performance, reducing the average accuracy to 0.45, particularly affecting Acc. Once and Acc. Range. Finally, removing the *LengthReview* module led to a drop in content generation quality due to unmet length constraints, highlighting its role in fine-tuning the output to meet requirements. Overall, the results emphasize the importance of each component, with *PlanRevise* and *PlanAdjust* playing key roles in ensuring task decomposition, plan refinement, and overall accuracy of generation.

**Length Control Performance** As specified in Section 3, each description  $D_i$  must achieve a target word count of  $L$ . To evaluate compliance with this requirement, we conducted an analysis of word count distributions across different models. Taking the Diary Writing task as an example, Figure 3 illustrates the performance of Llama-3.3-70B-Instruct and Qwen-2.5-14B-Instruct. The box plot reveals that these base models struggle to meet

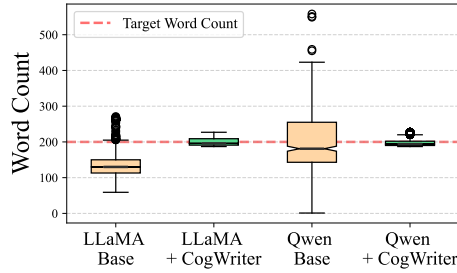


Figure 3: Comparison of Length Control Ability.

the word count requirement, with high variance and frequent deviations from the target length. In contrast, CogWriter achieves superior length control, as shown by its tighter, more stable distribution of word counts. The explicit monitoring mechanism within CogWriter effectively reduces variance and ensures consistent compliance with the length requirement. We provide further analysis results of other models and tasks in Appendix A.1.

### Challenges in Handling Complex Instructions

As shown in Figure 4, our experiments reveal that for all baselines and our model, the average performance follows a consistent ranking: Single Instructions (SI) outperform Range Instructions (RI), while Periodic Instructions (PI) show the lowest success rate. This indicates that, despite task decomposition simplifying the overall process, LLMs still face difficulties in understanding and executing complex instructions. One major issue is instruction overload—as the number of instructions increases, the model’s accuracy drops due to the difficulty in managing multiple constraints simultaneously. Additionally, instruction complexity plays a significant role: Single Instructions are easier as they target fixed positions, Range Instructions involve more positional flexibility, and Periodic Instructions require tracking repetitions across intervals, making them the most challenging to execute correctly. To improve performance in real-world application, it is advisable to limit the number of instructions and manually simplify complex or overlapping instructions where possible.

## 7 Related Work

**Long-form Text Generation** Recent advances in long-form generation have focused on improving models through architectural enhancements and specialized training techniques (Salemi et al., 2025a; Que et al., 2024; Liu et al., 2023; Li et al., 2023). Approaches like Re3 (Yang et al., 2022)

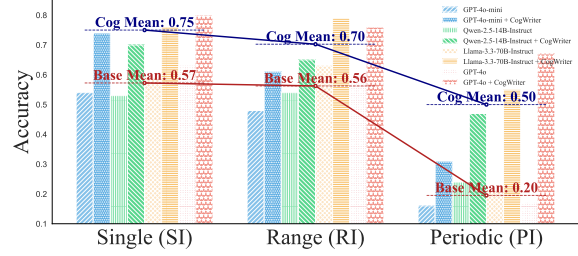


Figure 4: Comparison of Instruction Type Performance.

use recursive reprompting for extended story generation, while DOC (Yang et al., 2023) and hierarchical outlining (Wang et al., 2024c) improve narrative coherence through structured task decomposition. Personalized long-form generation has also gained attention (Salemi et al., 2025a; Wang et al., 2024a), with methods like LongLaMP (Kumar et al., 2024) and reasoning-enhanced techniques (Salemi et al., 2025b) adapting models to meet user-specific needs. Similarly, long-form question answering focuses on producing detailed responses to complex queries (Dasigi et al., 2021; Stelmakh et al., 2022; Lee et al., 2023; Tan et al., 2024). While these methods have improved generation capabilities (Wu et al., 2024a; Que et al., 2024), our work addresses a critical gap by examining long-form generation through the lens of cognitive writing theory.

**Multi-agent Writing** Multi-agent writing has made notable progress in recent years (Guo et al., 2024; Liu et al., 2024; Song et al., 2024), showing how agents can collaborate on diverse writing tasks (Wang et al., 2024b; Hong et al., 2024). Research has explored heterogeneous agent integration (Chen et al., 2025a) and educational applications (Shahzad et al., 2024). In academic writing, frameworks like SciAgents (Ghafarollahi and Buehler, 2024) demonstrate collaboration among specialized agents for complex writing tasks (Wang et al., 2024d; D’Arcy et al., 2024; Su et al., 2024), while the Agents’ Room approach (Huot et al., 2024) highlights the value of task decomposition in narrative writing. Beyond academic contexts, multi-agent methods have been applied to creative and informational writing, such as Wikipedia-style articles (Shao et al., 2024) and poetry (Zhang and Eger, 2024; Chen et al., 2024b). While these methods focus on collaboration, our work applies cognitive writing principles with agents for planning, monitoring, and revisions, enabling flexible adaptation without task-specific training.



## 8 Conclusion and Future Work

In this paper, we analyzed the challenges of constrained long-form text generation from a cognitive writing perspective. Building on these insights and empirical observations, we proposed CogWriter, a novel writing framework that transforms LLM constrained long-form text generation into a systematic cognitive paradigm. CogWriter bridges the gap between human writing cognition and LLM capabilities, leading to substantial and consistent improvements in both instruction completion and generation length across different LLMs, as demonstrated through extensive experiments on LongGenBench. Looking forward, we plan to optimize agent communication cost and develop specialized models that better align with the unique requirements of each cognitive stage in the writing process.

### Limitations

While demonstrating superior performance, CogWriter exhibits two primary limitations. First, while our approach achieves higher quality output, it necessitates more computational resources. As detailed in Appendix A.2, this additional cost stems from multiple rounds of planning, generation, and reviewing. Second, our current implementation utilizes a single LLM across all cognitive writing stages (planning, generation, and reviewing). This uniform approach may not fully leverage the model’s capabilities, as each stage only activates specific aspects of the model’s knowledge and abilities. Future research directions include exploring specialized models for different cognitive stages and investigating Mixture-of-Experts architectures to enhance both domain expertise and parameter efficiency in the cognitive writing process.

### Ethical Considerations

Like other LLMs, our CogWriter framework may inherit biases from training data. It may generate inaccurate content despite its enhanced control mechanisms, emphasizing the need for human oversight in practical applications. While the multi-step cognitive process increases computational costs, the structured planning approach improves efficiency and could be further optimized for sustainability. As with any advanced text generation system, CogWriter could potentially be misused for generating deceptive content, highlighting the importance of responsible deployment and appropriate safeguards in real-world applications.

## References

- Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.
- Minwook Bae and Hyounghun Kim. 2024. Collective critics for creative story generation. In *Proc. of EMNLP*, pages 18784–18819.
- Yushi Bai, Jiajie Zhang, Xin Lv, Linzhi Zheng, Siqi Zhu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2024. [Longwriter: Unleashing 10,000+ word generation from long context llms](#). *Preprint*, arXiv:2408.07055.
- Carl Bereiter and Marlene Scardamalia. 2013. *The psychology of written composition*. Routledge.
- Weize Chen, Ziming You, Ran Li, yitong guan, Chen Qian, Chenyang Zhao, Cheng Yang, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2025a. Internet of agents: Weaving a web of heterogeneous agents for collaborative intelligence. In *Proc. of ICLR*.
- Xiuying Chen, Mingzhe Li, Shen Gao, Xin Cheng, Qingqing Zhu, Rui Yan, Xin Gao, and Xiangliang Zhang. 2024a. Flexible and adaptable summarization via expertise separation. In *Proc. of SIGIR*, pages 2018–2027.
- Xiuying Chen, Tairan Wang, Juexiao Zhou, Zirui Song, Xin Gao, and Xiangliang Zhang. 2025b. Evaluating and mitigating bias in ai-based medical text generation. *Nature Computational Science*, pages 1–9.
- Yanran Chen, Hannes Gröner, Sina Zarrieß, and Steffen Eger. 2024b. Evaluating diversity in automatic poetry generation. In *Proc. of EMNLP*, pages 19671–19692.
- Jiale Cheng, Xiao Liu, Cunxiang Wang, Xiaotao Gu, Yida Lu, Dan Zhang, Yuxiao Dong, Jie Tang, Hongning Wang, and Minlie Huang. 2024a. [Spar: Self-play with tree-search refinement to improve instruction-following in large language models](#). *Preprint*, arXiv:2412.11605.
- Xin Cheng, Di Luo, Xiuying Chen, Lemao Liu, Dongyan Zhao, and Rui Yan. 2024b. Lift yourself up: Retrieval-augmented text generation with self-memory. *Advances in Neural Information Processing Systems*, 36.
- Mike D’Arcy, Tom Hope, Larry Birnbaum, and Doug Downey. 2024. Marg: Multi-agent review generation for scientific papers. *ArXiv*.
- Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A. Smith, and Matt Gardner. 2021. A dataset of information-seeking questions and answers anchored in research papers. In *Proc. of NAACL*, pages 4599–4610.
- L Flower. 1981. A cognitive process theory of writing. *Composition and communication*.

- Alireza Ghafarollahi and Markus J. Buehler. 2024. Scia-gents: Automating scientific discovery through bio-inspired multi-agent intelligent graph reasoning. *Advanced materials*, page e2413523.
- Aaron Grattafiori and et al. 2024. [The llama 3 herd of models](#). *Preprint*, arXiv:2407.21783.
- Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V Chawla, Olaf Wiest, and Xiangliang Zhang. 2024. Large language model based multi-agents: A survey of progress and challenges. *Proc. of IJCAI*.
- John R Hayes and Linda S Flower. 2016. Identifying the organization of writing processes. In *Cognitive processes in writing*, pages 3–30. Routledge.
- Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024. MetaGPT: Meta programming for a multi-agent collaborative framework. In *Proc. of ICLR*.
- Mengkang Hu, Tianxing Chen, Qiguang Chen, Yao Mu, Wenqi Shao, and Ping Luo. 2024. [Hiagent: Hierarchical working memory management for solving long-horizon agent tasks with large language model](#). *Preprint*, arXiv:2408.09559.
- Fantine Huot, Reinald Kim Amplayo, Jennimaria Palomaki, Alice Shoshana Jakobovits, Elizabeth Clark, and Mirella Lapata. 2024. [Agents’ room: Narrative generation through multi-step collaboration](#). *Preprint*, arXiv:2410.02603.
- Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, L  lio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timoth  e Lacroix, and William El Sayed. 2023. [Mistral 7b](#). *Preprint*, arXiv:2310.06825.
- Ronald T Kellogg. 2013. A model of working memory in writing. In *The science of writing*, pages 57–71. Routledge.
- Ishita Kumar, Snigdha Viswanathan, Sushrita Yerra, Alireza Salemi, Ryan A. Rossi, Franck Dernoncourt, Hanieh Deilamsalehy, Xiang Chen, Ruiyi Zhang, Shubham Agarwal, Nedim Lipka, Chien Van Nguyen, Thien Huu Nguyen, and Hamed Zamani. 2024. [Longlamp: A benchmark for personalized long-form text generation](#). *Preprint*, arXiv:2407.11016.
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626.
- Yoonjoo Lee, Kyungjae Lee, Sunghyun Park, Dasol Hwang, Jaehyeon Kim, Hong-In Lee, and Moontae Lee. 2023. QASA: Advanced question answering on scientific articles. In *Proc. of ICML*, pages 19036–19052.
- Cheng Li, Mingyang Zhang, Qiaozhu Mei, Yaqing Wang, Spurthi Amba Hombaiah, Yi Liang, and Michael Bendersky. 2023. Teach llms to personalize - an approach inspired by writing education. *ArXiv*.
- Siyang Liu, Naihao Deng, Sahand Sabour, Yilin Jia, Minlie Huang, and Rada Mihalcea. 2023. Task-adaptive tokenization: Enhancing long-form text generation efficacy in mental health and beyond. In *Proc. of EMNLP*, pages 15264–15281.
- Yuhan Liu, Xiuying Chen, Xiaoqing Zhang, Xing Gao, Ji Zhang, and Rui Yan. 2024. From skepticism to acceptance: Simulating the attitude dynamics toward fake news. *Proc. of IJCAI*.
- Haoran Luo, Yuhao Yang, Tianyu Yao, Yikai Guo, Zichen Tang, Wentai Zhang, Shiyao Peng, Kaiyang Wan, Meina Song, Wei Lin, et al. 2024. Text2nkg: Fine-grained n-ary relation extraction for n-ary relational knowledge graph construction. *Proc. of NeurIPS*, 37:27417–27439.
- Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *Proc. of NeurIPS*, pages 46534–46594.
- Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. 2024. [Large language models: A survey](#). *Preprint*, arXiv:2402.06196.
- Saurav Pawar, S. M Towhidul Islam Tonmoy, S M Mehedi Zaman, Vinija Jain, Aman Chadha, and Amitava Das. 2024. [The what, why, and how of context length extension techniques in large language models – a detailed survey](#). *Preprint*, arXiv:2401.07872.
- Bowen Ping, Jiali Zeng, Fandong Meng, Shuo Wang, Jie Zhou, and Shanghang Zhang. 2025. [Longdpo: Unlock better long-form generation abilities for llms via critique-augmented stepwise information](#). *Preprint*, arXiv:2502.02095.
- Haoran Que, Feiyu Duan, Liqun He, Yutao Mou, Wangchunshu Zhou, Jiaheng Liu, Wenge Rong, Zekun Moore Wang, Jian Yang, Ge Zhang, Junran Peng, Zhaoxiang Zhang, Songyang Zhang, and Kai Chen. 2024. [Hellobench: Evaluating long text generation capabilities of large language models](#). *Preprint*, arXiv:2409.16191.
- 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, Jiayi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2025. [Qwen2.5 technical report](#). *Preprint*, arXiv:2412.15115.
- Zeeshan Rasheed, Muhammad Waseem, Kai Kristian Kemell, Aakash Ahmad, Malik Abdul Sami, Jussi Rasku, Kari Systä, and Pekka Abrahamsson. 2025. [Large language models for code generation: The practitioners perspective](#). *Preprint*, arXiv:2501.16998.
- Alireza Salemi, Julian Killingback, and Hamed Zamani. 2025a. [Expert: Effective and explainable evaluation of personalized long-form text generation](#). *Preprint*, arXiv:2501.14956.
- Alireza Salemi, Cheng Li, Mingyang Zhang, Qiaozhu Mei, Weize Kong, Tao Chen, Zhuowan Li, Michael Bendersky, and Hamed Zamani. 2025b. [Reasoning-enhanced self-training for long-form personalized text generation](#). *Preprint*, arXiv:2501.04167.
- Samuel Schmidgall, Yusheng Su, Ze Wang, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Zicheng Liu, and Emad Barsoum. 2025. [Agent laboratory: Using llm agents as research assistants](#). *Preprint*, arXiv:2501.04227.
- Rimsha Shahzad, Muhammad Aslam, Shaha T. Al-Otaibi, Muhammad Saqib Javed, Amjad Rehman Khan, Saeed Ali Bahaj, and Tanzila Saba. 2024. Multi-agent system for students cognitive assessment in e-learning environment. *IEEE Access*, pages 15458–15467.
- Yijia Shao, Yucheng Jiang, Theodore Kanell, Peter Xu, Omar Khattab, and Monica Lam. 2024. Assisting in writing Wikipedia-like articles from scratch with large language models. In *Proc. of NAACL*, pages 6252–6278.
- Wei Shi, Shuang Li, Kerun Yu, Jinglei Chen, Zujie Liang, Xinhui Wu, Yuxi Qian, Feng Wei, Bo Zheng, Jiaqing Liang, Jiangjie Chen, and Yanghua Xiao. 2024. [Segment+: Long text processing with short-context language models](#). *Preprint*, arXiv:2410.06519.
- Zirui Song, Guangxian Ouyang, Meng Fang, Hongbin Na, Zijiang Shi, Zhenhao Chen, Yujie Fu, Zeyu Zhang, Shiyu Jiang, Miao Fang, et al. 2024. Hazards in daily life? enabling robots to proactively detect and resolve anomalies. *arXiv preprint arXiv:2411.00781*.
- Ivan Stelmakh, Yi Luan, Bhuwan Dhingra, and Ming-Wei Chang. 2022. ASQA: Factoid questions meet long-form answers. In *Proc. of EMNLP*, pages 8273–8288.
- Haoyang Su, Renqi Chen, Shixiang Tang, Xinzhe Zheng, Jingzhe Li, Zhenfei Yin, Wanli Ouyang, and Nanqing Dong. 2024. [Two heads are better than one: A multi-agent system has the potential to improve scientific idea generation](#). *Preprint*, arXiv:2410.09403.
- Haochen Tan, Zhijiang Guo, Zhan Shi, Lu Xu, Zhili Liu, Yunlong Feng, Xiaoguang Li, Yasheng Wang, Lifeng Shang, Qun Liu, and Linqi Song. 2024. ProxyQA: An alternative framework for evaluating long-form text generation with large language models. In *Proc. of ACL*, pages 6806–6827.
- Qwen Team. 2024. [Qwen2.5: A party of foundation models](#).
- Hugo Touvron, Albert Jiang, et al. 2024. [Llama 3: Open and efficient foundation models](#).
- Danqing Wang, Kevin Yang, Hanlin Zhu, Xiaomeng Yang, Andrew Cohen, Lei Li, and Yuandong Tian. 2024a. Learning personalized alignment for evaluating open-ended text generation. In *Proc. of EMNLP*, pages 13274–13292.
- Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. 2024b. A survey on large language model based autonomous agents. *Front. Comput. Sci.*
- Qian Yue Wang, Jinwu Hu, Zhengping Li, Yufeng Wang, daiyuan li, Yu Hu, and Mingkui Tan. 2024c. [Generating long-form story using dynamic hierarchical outlining with memory-enhancement](#). *Preprint*, arXiv:2412.13575.
- Yidong Wang, Qi Guo, Wenjin Yao, Hongbo Zhang, Xin Zhang, Zhen Wu, Meishan Zhang, Xinyu Dai, Min Zhang, Qingsong Wen, Wei Ye, Shikun Zhang, and Yue Zhang. 2024d. Autosurvey: Large language models can automatically write surveys. In *Proc. of NeurIPS*.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proc. of NeurIPS*, pages 24824–24837.
- Yuhao Wu, Ming Shan Hee, Zhiqing Hu, and Roy Ka-Wei Lee. 2024a. Longgenbench: Benchmarking long-form generation in long context llms. *ICLR*.
- Zhenyu Wu, Qingkai Zeng, Zhihan Zhang, Zhaoxuan Tan, Chao Shen, and Meng Jiang. 2024b. Large language models can self-correct with key condition verification. In *Proc. of EMNLP*, pages 12846–12867.
- Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Yuhao Zhou, Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, Zhangyue Yin, Shihan Dou, Rongxiang Weng, Wensen Cheng, Qi Zhang, Wenjuan Qin, Yongyan



Zheng, Xipeng Qiu, Xuanjing Huang, and Tao Gui. 2023. [The rise and potential of large language model based agents: A survey](#). *arXiv preprint*.

Zhuohan Xie, Trevor Cohn, and Jey Han Lau. 2023. The next chapter: A study of large language models in storytelling. In *Proceedings of the 16th International Natural Language Generation Conference*, pages 323–351.

Kevin Yang, Dan Klein, Nanyun Peng, and Yuandong Tian. 2023. DOC: Improving long story coherence with detailed outline control. In *Proc. of ACL*, pages 3378–3465.

Kevin Yang, Yuandong Tian, Nanyun Peng, and Dan Klein. 2022. Re3: Generating longer stories with recursive reprompting and revision. In *Proc. of EMNLP*, pages 4393–4479.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: deliberate problem solving with large language models. In *Proc. of ICONIP*.

Ran Zhang and Steffen Eger. 2024. [Llm-based multi-agent poetry generation in non-cooperative environments](#). *Preprint*, arXiv:2409.03659.

## A Appendix

### A.1 Further Length Control Performance

To comprehensively demonstrate CogWriter’s length control capabilities across different scenarios, we present the generated length distribution of LLaMA-3.3-70B-Instruct, Qwen-2.5-14B-Instruct, GPT-4o, and GPT-4o-mini in Figures 5a-5d. We evaluate two distinct task types: spatial tasks (150 words) and temporal tasks (200 words). Spatial tasks, such as Skyscraper Design and Urban Planning, require detailed facility arrangements across floors or city blocks, with a target length of 150 words per unit. In contrast, temporal tasks, including Diary Writing and Menu Design, emphasize temporal consistency across weeks of a year and require 200 words per weekly entry. Figures 5a and 5c illustrate model performance on spatial tasks, while Figures 5b and 5d present results for temporal tasks, highlighting the models’ ability to adhere to different length constraints across varying task structures.

### A.2 Inference Time and Token Consumption Analysis

To evaluate and analyze the computational efficiency of CogWriter, we conducted comprehensive

experiments examining inference time and token consumption amount.

**Inference Time** For ensure reliable evaluation, we used LLaMA-3.3-70B as our test model, as Qwen exhibited incomplete text generation issues and GPT’s API calls were subject to network latency variations. All experiments were performed on 4 NVIDIA A100 GPUs, with each condition tested three times to ensure reliable results. The experiments were structured as follows: 1) Single text condition: One randomly sampled writing task and 2) 4-example condition: One randomly sampled example from each of the four tasks. We leveraged vLLM for inference acceleration while maintaining default temperature and sampling parameters from official Hugging Face implementations. To ensure a fair comparison, we only considered outputs achieving 100% completion rate. Figure 6 illustrates the inference time comparison between CogWriter and the baseline model across different batch sizes.

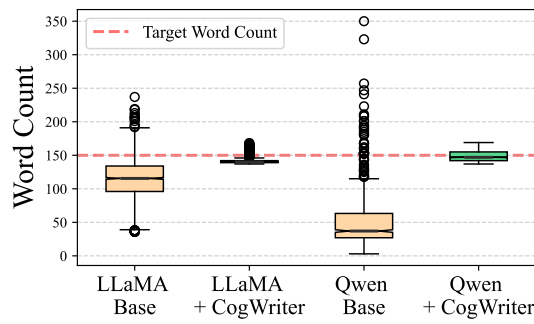
Through the implementation of multi-generation agents for parallel processing, our approach demonstrates a significant reduction in generation time, achieving approximately 50% faster processing compared to the baseline model.

**Token Consumption** Our analysis reveals that CogWriter consumes approximately 2.8 times more output tokens and 10 times more total tokens compared to baseline methods. The observed increase in token utilization can be attributed to two primary factors:

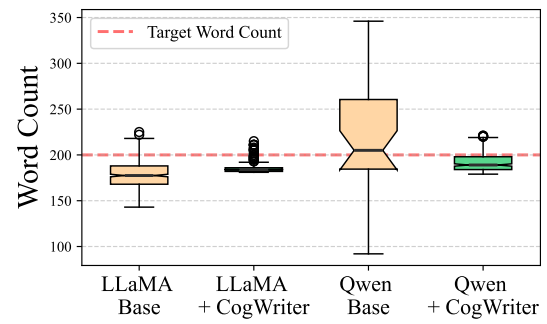
1. While CogWriter ensures comprehensive output generation, baseline models frequently produce responses that are incomplete in quality and length. Notably, baseline models such as GPT-4o often acknowledge their limitations with responses like “I’m sorry, but creating an entire year’s worth of weekly diary entries with detailed narratives is beyond my capabilities in a single response,” resulting in artificially lower token consumption metrics.
2. CogWriter employs an iterative approach involving multiple rounds of plan evaluation against the original prompt, analogous to the human writing process where additional cognitive effort correlates with enhanced document quality and comprehensiveness, thereby increasing token usage.

Despite these considerations, it is noteworthy

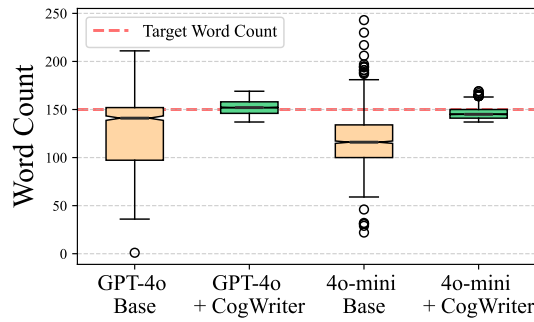




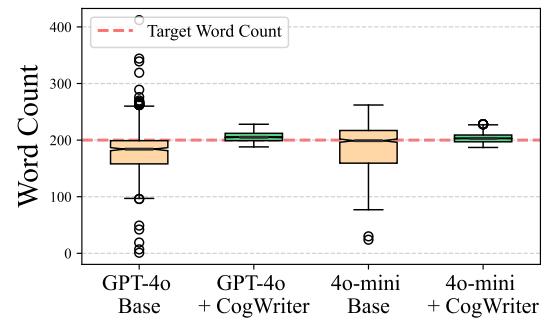
(a) Llama and Qwen on Spatial Tasks



(b) Llama and Qwen on Temporal Tasks



(c) GPT-4o and GPT-4o-mini on Spatial Tasks



(d) GPT-4o and GPT-4o-mini on Temporal Tasks

Figure 5: Length Control Performance Across Different Models and Task Types. (a) and (c) show performance on spatial tasks requiring 150 words per unit, while (b) and (d) present results for temporal tasks with 200-word requirements.

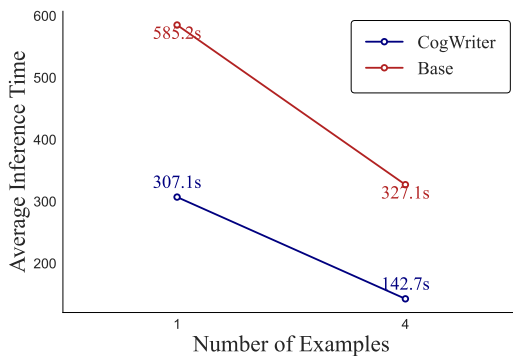


Figure 6: Inference Time Comparison.

that while GPT-4o’s API pricing is 16.67 times higher than GPT-4o-mini<sup>1</sup>, it achieves only a marginal improvement in Average Accuracy (0.08), as demonstrated in Table 1. In contrast, CogWriter demonstrates a more substantial improvement of 0.16 in Average Accuracy over GPT-4o-mini. Furthermore, our framework can be implemented with lightweight closed-source models such as Qwen-2.5-14B-Instruct, enabling local deployment. This capability is particularly valuable for applications prioritizing output quality and data privacy, includ-

<sup>1</sup><https://openai.com/api/pricing/>

ing professional content creation, academic writing, and technical documentation.

Our research primarily focuses on transcending the limitations inherent in conventional single-pass generation approaches, aiming to achieve text quality that surpasses the capabilities of individual LLMs, including advanced models like GPT-4o. Much like professional writing practices, where quality content necessitates extended development time and thinking compared to preliminary drafts, CogWriter’s increased resource utilization reflects the sophistication of its cognitive processing mechanisms.

While acknowledging the additional computational overhead, we identify several promising directions for future research, including the development of memory optimization techniques and the exploration of specialized writing models with enhanced parameter efficiency for specific cognitive processes in the generation pipeline.