ConvergeWriter: Data-Driven Bottom-Up Article Constructions

Anonymous ACL submission

Abstract

Large Language Models (LLMs) excel at text generation, but still struggle to produce accurate, long-form documents grounded in large knowledge bases. Existing "top-down" approaches—which plan before retrieving evidence—often result in fragmented or unfaithful content due to misalignment between the plan and available knowledge.

To overcome this, we introduce a "bottom-up" framework based on a "Retrieval-First for Knowledge, Clustering for Structure" strategy. Our method begins by establishing the "knowledge boundaries" of the source corpus through comprehensive retrieval, followed by unsupervised clustering to group evidence into distinct "knowledge clusters." These clusters objectively inform a hierarchical outline and subsequent text generation, ensuring strict adherence to the source and significantly reducing hallucination.

Experimental results on both 14B and 32B parameter models demonstrate that our method achieves performance comparable to or exceeding state-of-the-art baselines, and is expected to demonstrate unique advantages in knowledge-constrained scenarios that demand high fidelity and structural coherence. Our work presents an effective paradigm for generating reliable, structured, long-form documents, paving the way for more robust LLM applications in high-stakes, knowledge-intensive domains.

1 Introduction

In recent years, Large Language Models (LLMs) have made significant advances in Natural Language Processing (NLP), excelling in tasks like open-domain QA and text generation (Yang et al., 2023; Zhao et al., 2025; Yue, 2025). However, effectively incorporating external knowledge to produce detailed, credible, and hallucination-free long-form documents remains challenging. This task requires not only coherent long-text generation but also integration of reference materials beyond the model's context window, while ensuring accuracy and structural integrity (Xu and Peng, 2025).

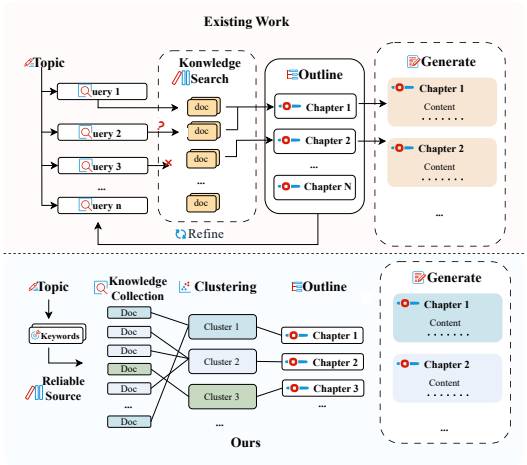

Figure 1: The process differences between our approach and the previous top-down framework.

Specifically, leveraging external knowledge for long-text generation faces three core challenges:

- **Massive Context Processing:** Referencing large volumes of background material demands processing ultra-long texts, posing computational challenges and requiring efficient extraction of salient information (Liu et al., 2024; Wang et al., 2024b).
- **Knowledge Organization and Structuring:** Multi-source knowledge must be structured logically to avoid redundancy and ensure coherence, going beyond mere information listing (Skarlinski et al., 2024).
- **High-Fidelity and Low-Hallucination:** In accuracy-critical domains (e.g., finance, healthcare), generated text must strictly adhere to credible sources under "closed knowledge" constraints to eliminate hallucinations (Ji et al., 2023; Shao et al., 2024).

Existing methods like OmniThink (Xi et al., 2025) and STORM (Shao et al., 2024) typically adopt a top-down, hypothesis-driven strategy: first outlining a structure or raising questions, then retrieving evidence to fit this framework. This approach, however, may misalign with the knowledge base, leading to inefficient retrieval, fragmented content, and reduced credibility (Shi et al., 2024).

To overcome these issues, we propose **ConvergeWriter**, a bottom-up, data-driven framework for long-text generation based on "retrieval-first for knowledge, clustering for structure". It begins by retrieving relevant knowledge, then plans the article's content and structure based on the actual evidence, constraining generation to available knowledge and avoiding LLM hallucinations. Our approach addresses each challenge as follows:

- **For Massive Context Processing:** We use multi-round relevance-expanding retrieval to gather documents, then apply tree-structured hierarchical summarization (inspired by RAPTOR (Sarthi et al., 2024)) to compress text while preserving core information, significantly increasing processable document length and volume.
- **For Knowledge Organization and Structuring:** Retrieved documents are clustered into traceable **Knowledge Clusters**, forming a content-grounded framework. The article outline is derived directly from these clusters, ensuring structural coherence and knowledge-base alignment.
- **For High-Fidelity and Low-Hallucination:** The outline is fully data-driven. During paragraph generation, we use an "Outline–Knowledge Cluster Jointly Guided" retrieval-augmented approach, anchoring each segment to verifiable documents, mitigating hallucinations caused by information gaps or model fabrication.

Experiments using Wikipedia as the sole retrieval source show that ConvergeWriter achieves competitive or superior performance compared to state-of-the-art baselines on both 14B and 32B parameter models, validating its effectiveness in knowledge-bound and high-reliability scenarios.

2 Related Work

Recent advancements in long-text generation with Large Language Models largely follow a hypothesis-driven, top-down paradigm. Systems such as (Shao et al., 2024) decompose writing into iterative stages, including multi-perspective knowledge gathering, outlining, and drafting. Similarly, (Xi et al., 2025) expand knowledge via dynamic "information tree" construction and abstraction, while (Bai et al., 2025) separates generation into planning and sequential paragraph writing. Across these approaches, retrieval mainly validates or supplements a predefined outline constructed before the full knowledge scope is known (Bai et al., 2024).

Multi-agent frameworks further extend this paradigm. (Wang et al., 2024a) employ Planner/Author/Reviewer agents guided by a predefined "PGTree", using Reference-Reviewer-Augmented Generation (RRAG) for local optimization. (Wan et al., 2025) simulate human writing through hierarchical planning and parallel generation with iterative revision. (Li et al., 2025) integrate real-time web exploration in a "Think-Search-Draft" loop with reinforcement learning. Despite improved task decomposition and coherence, these methods still rely on an initial plan formed without verified knowledge availability, risking structural misalignment with retrievable evidence.

Despite their effectiveness, existing approaches face three challenges in knowledge-constrained scenarios:

1. **Knowledge-Structure Mismatch:** Predefined outlines may exceed retrievable evidence, causing fragmented or fabricated sections.
2. **Static Planning:** Outlines cannot adapt to actual knowledge coverage, forcing models to speculate.
3. **Opaque Provenance:** The link between generated content and source documents remains indirect, limiting verifiability.

To address these issues, ConvergeWriter proposes a data-driven, bottom-up framework. It first performs retrieval-first clustering to identify knowledge boundaries, then induces the article structure accordingly. This grounds generation in available evidence, mitigating outline hallucination while improving traceability through cluster-anchored generation.

3 Methodology

The core workflow of ConvergeWriter is illustrated in Fig. 12. The method begins with multiple rounds

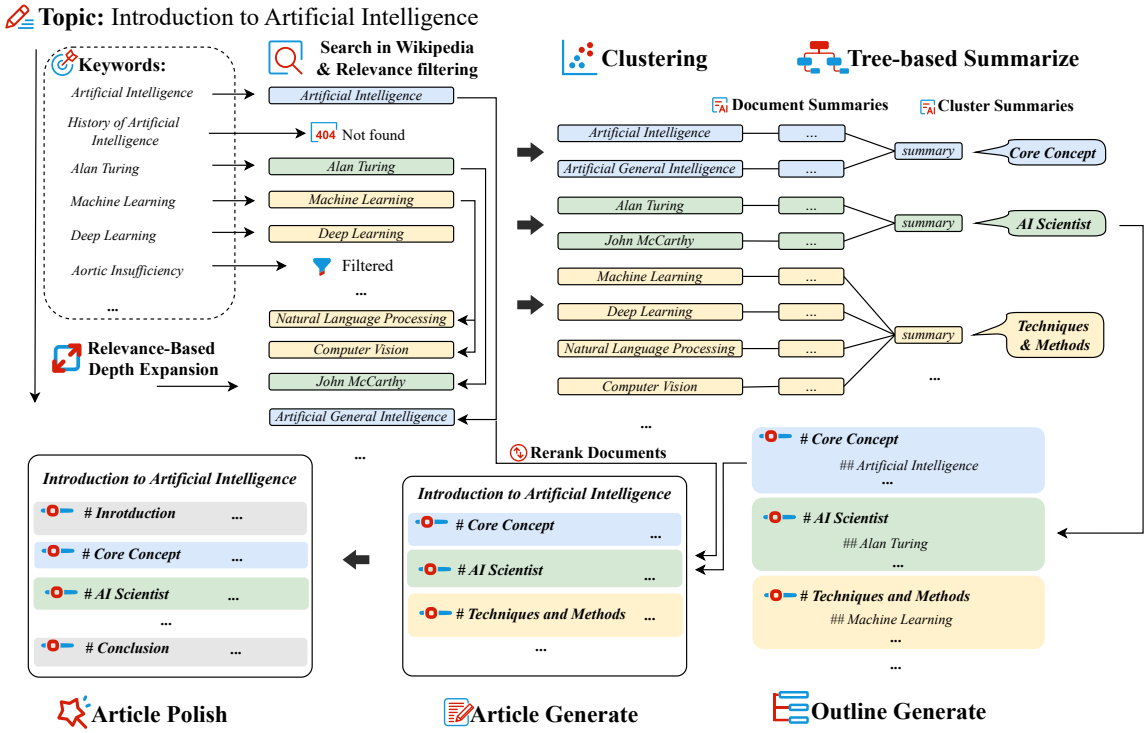

Figure 2: ConvergeWriter's workflow: (1) Iterative relevance-expanding knowledge retrieval and document screening; (2) Unsupervised clustering with tree-structured summarization; (3) LLM-based outline generation; (4) Recall-enhanced chapter-wise content generation, introduction/conclusion supplementation, and polishing.

of relevance-expanding knowledge retrieval and supporting document screening. Subsequently, unsupervised clustering is performed on the documents to form knowledge clusters, combined with tree-structured summarization to construct a hierarchical knowledge structure. Following this, a logically coherent outline (where main chapters correspond to knowledge clusters) is generated by an LLM based on the integrated summaries. Finally, a recall-enhanced generation paradigm jointly driven by the outline and knowledge clusters is employed for chapter-wise content generation and integration. Content for each chapter is modularly generated by the LLM driven by the outline and constructed knowledge clusters, followed by the supplementation of the introduction and conclusion sections, along with comprehensive article polishing.

3.1 Relevance-Expanding Knowledge Retrieval

To enhance the concreteness and depth of the generated content, we design a relevance-expanding knowledge retrieval module that systematically collects high-coverage documents highly relevant to the topic T from external knowledge bases (e.g., Wikipedia). This process is completed through an iterative two-stage approach:

Stage 1: Breadth-First Retrieval
First, leverage a Large Language Model (LLM) $\mathcal{M}$ to generate an initial keyword set based on topic T:

$$\mathcal{K}_0 = \mathcal{M}(T; \mathcal{I}_{\text{keyword-gen}}) \tag{1}$$

Then invoke the retrieval interface $\mathcal{R}$ (e.g., Wikipedia API) using $\mathcal{K}_0$ to obtain the preliminary document set $\mathcal{D}^{(0)} = \mathcal{R}(\mathcal{K}_0)$. Subsequently, perform relevance filtering on $\mathcal{D}^{(0)}$ via the LLM, retaining only documents relevant to T to form a refined set:

$$\mathcal{D}^{(1)} = \{d \in \mathcal{D}^{(0)} \mid \mathcal{M}(T, d; \mathcal{I}_{\text{rel-filter}}) = \text{Relevant}\} \tag{2}$$

Stage 2: Relevance-Based Depth Expansion
To deepen the knowledge collection and mitigate potential omissions of critical subtopics or details, we perform relevance-based depth expansion based on $\mathcal{D}^{(1)}$. For each document d in $\mathcal{D}^{(1)}$, utilize the LLM $\mathcal{M}$ to generate expanded keywords guided by both T and the content of d, prompting it to explore deeper potential information within the context of the current topic, thereby forming an expanded keyword set:

$$\mathcal{K}_{\text{ext}} = \bigcup_{d \in \mathcal{D}^{(1)}} \mathcal{M}(T, d; \mathcal{I}_{\text{depth-exp}}) \tag{3}$$

Use $\mathcal{K}_{\text{ext}}$ for secondary retrieval to obtain $\mathcal{D}_{\text{raw}}^{(2)}$, followed by the same relevance filtering as in Stage 1. Finally, merge the results from both stages to construct the high-quality knowledge document set:

$$\mathcal{D}^* = \mathcal{D}^{(1)} \cup \{d \in \mathcal{D}_{\text{raw}}^{(2)} \mid \mathcal{M}(T, d; \mathcal{I}_{\text{rel-filter}}) = \text{Relevant}\} \quad (4)$$

Through this iterative two-stage retrieval process, we construct a high-quality, highly relevant, and sufficiently deep knowledge document set $\mathcal{D}^*$ to support subsequent article generation. This retrieval-first strategy ensures that subsequent content planning and generation strictly adhere to the boundaries of practically available knowledge evidence.

3.2 Knowledge Structurization

After obtaining the high-quality candidate document set $\mathcal{D}^*$, we organize it structurally to reveal its inherent topical distribution, paving the way for generating a logically coherent article outline in subsequent stages. The core of this process is transforming the unstructured document collection into a set of semantically meaningful knowledge clusters.

Specifically, this step consists of two phases: document clustering and hierarchical summarization:

1. Document Clustering via Silhouette Coefficient Optimization First, we employ a pre-trained embedding model $\mathcal{E}$ to map each document d_i in $\mathcal{D}^*$ to a semantic vector $v_i = \mathcal{E}(d_i)$, projecting the entire document set into a high-dimensional vector space.

Next, to partition these vectors into semantically cohesive topic clusters, we utilize the K-means clustering algorithm. To automatically determine the optimal number of clusters k, we adopt the silhouette(Rousseeuw, 1987) coefficient as the evaluation metric. We iterate over k values within a predefined range $[k_{\min}, k_{\max}]$ and select the k that maximizes the average silhouette coefficient $\bar{s}(k)$:

$$k^* = \underset{k \in [k_{\min}, k_{\max}]}{\arg\max} \ \bar{s}(k) \quad (5)$$

To enhance computational efficiency for large document sets, we approximate the silhouette coefficient calculation through random sampling. Finally, using the determined optimal k^*, we execute the K-means algorithm on all document embedding vectors, obtaining a collection of k^* knowledge clusters $\{C_1, C_2, \ldots, C_{k^*}\}$.

2. Tree-Structured Summarization for Long Contexts After clustering, each cluster C_j may contain numerous documents. Directly concatenating document contents could easily exceed the model's context window limit. To address this, we adopt a hierarchical tree-structured summarization strategy:

- **Leaf Node Summarization:** For each document d_i in cluster C_j, we invoke an LLM to generate a concise summary:

$$s_i = \mathcal{M}(d_i; \mathcal{I}_{\text{summarize}}). \quad (6)$$

- **Root Node Summarization:** We concatenate all document summaries within the cluster $\{s_i\}_{d_i \in C_j}$ and input them into the LLM again to generate a cluster-level descriptive summary:

$$S_j = \mathcal{M}\left(\texttt{concat}(\{s_i\}); \mathcal{I}_{\text{cluster-summarize}}\right) \quad (7)$$

Through this process, the original document set $\mathcal{D}^*$ is transformed into a collection of semantically cohesive knowledge clusters $\{C_j\}_{j=1}^{k^*}$, each accompanied by a refined summary S_j. This hierarchical knowledge representation not only reveals the intrinsic structure of the available knowledge but also provides direct and reliable input for generating a high-quality article outline based on this structure in the subsequent stage.

3.3 Structured Knowledge Mapping-Based Outline Generation

After completing the previous stage, a set of highly cohesive knowledge clusters $\{C_j\}_{j=1}^{k^*}$ is obtained, where each cluster C_j encapsulates the core information regarding a specific dimension of topic T in the knowledge base, characterized by its integrated summary S_j. The objective of this stage is to leverage these structured knowledge summaries to generate a logically rigorous and focus-oriented outline O for an article on topic T.

$$O = \mathcal{M}\left(\{S_j\}_{j=1}^{k^*}; \mathcal{I}_{\text{outline_gen}}\right) \quad (8)$$

To ensure the outline strictly adheres to the actual content boundaries of the knowledge base and mitigates hallucination risks, the outline generation process is subject to dual constraints: the

model must autonomously optimize the sequence of sections to form a cognitively logical argument flow, while exclusively extracting highly relevant core information for topic T from the summary set $\{S_j\}_{j=1}^{k^*}$. Furthermore, to guarantee machine readability, format consistency, and provide explicit guidance for subsequent generation, formatting constraints are imposed on the large language model (LLM) output. Specifically, O must strictly comply with Markdown syntax, and with the exception of the "Introduction" and "Conclusion" sections, each main section Sec_i must explicitly and uniquely map to a knowledge cluster C_j, i.e.:

$$\forall \text{Sec}_i \in O_{\text{body}}, \exists! \, C_j \in \mathcal{C} \; : \; f(\text{Sec}_i) = C_j \quad (9)$$

The core design principle ensures the **verifiability** and **source determinacy** of the article's central arguments, effectively circumventing hallucination risks.

3.4 3.4 Outline-Knowledge Cluster Joint-Driven Section-Wise Retrieval for Article Generation

This stage constructs the complete article based on the generated outline O. First, O is parsed into discrete section units $\{\text{Sec}_1, \text{Sec}_2, \ldots, \text{Sec}_K\}$. For each main section Sec_i ($i \notin \{1, K\}$), relevant documents from its corresponding knowledge cluster C_j are retrieved, and a Ranker model re-ranks them to produce an enhanced document set $\mathcal{D}^*_{\text{sec}_i}$. The section content is then generated as:

$$\text{Sec}_i^* = \mathcal{M}(\text{Sec}_i, \mathcal{D}^*_{\text{sec}_i}; \mathcal{I}_{\text{section_gen}}) \quad (10)$$

Upon independent generation of all main sections Sec_i^*, they are sequentially concatenated to form the draft body of the article $\mathcal{A}_{\text{draft}}$. Subsequently, leveraging the overall topic and the draft body $\mathcal{A}_{\text{draft}}$, the "Introduction" section Sec_1^* and "Conclusion" section Sec_K^* are generated separately. Integrating these introductory and concluding sections with the draft body yields the complete draft article $\mathcal{A}_{\text{full_draft}}$.

Finally, the large language model $\mathcal{M}$ performs global polishing on $\mathcal{A}_{\text{full_draft}}$ to produce a grammatically fluent, stylistically consistent, and coherent final article:

$$\mathcal{A}_{\text{final}} = \mathcal{M}(\mathcal{A}_{\text{full_draft}}; \mathcal{I}_{\text{refine}}) \quad (11)$$

4 Experiments

To comprehensively evaluate the effectiveness of **ConvergeWriter**, we follow the experimental paradigm of prior work (e.g., OmniThink(Xi et al., 2025)) and conduct exhaustive experiments on the **WildSeek**(Jiang et al., 2024) dataset. WildSeek is a dataset derived from real users' multi-turn information exploration sessions on open web applications STORM(Shao et al., 2024) (and Co-STORM(Jiang et al., 2024)), specifically designed for evaluating complex information retrieval and knowledge consolidation systems. Our experiments focus on a core task: given a topic title and a single, designated knowledge source (i.e., the Wikipedia retrieval API), assess the capability of different methods to collect, organize relevant information, and generate well-structured, informative propositional articles. This task setup aims to rigorously simulate application scenarios in vertical domains (e.g., finance, scientific report writing) with well-defined knowledge boundaries and demanding high information credibility.

4.1 Evaluation Metrics

To comprehensively and multi-dimensionally evaluate the performance of different methods in knowledge organization and article generation, and to address the core challenges outlined in the introduction (massive context processing, knowledge structuring, high trustworthiness), we employ the following four complementary evaluation metrics:

1. **Average Article Length (Length)**: Records the average word count of generated articles within the fixed context length limit of the LLM. This metric provides a direct measure of the textual output volume achievable by a method under the given constraint.

2. **Cited Documents (Cited Docs)**: Counts the average number of explicitly cited supporting documents per article, measuring the breadth and depth of evidence collection and application.

3. **LLM Automatic Evaluation**: Uses **Qwen3-32B**(Yang et al., 2025) as an evaluator to automatically score articles generated by all methods (on a 0-5 scale) across four key quality dimensions:

   - **Relevance**: Assesses how closely the article content aligns with the initial topic

intent, avoiding the introduction of irrelevant content or digressions.

- **Breadth**: Measures the comprehensiveness of an article by evaluating how broadly it covers the core aspects of the topic.
- **Depth**: Gauges the extent to which the article provides in-depth, detailed analysis of the topic and related domains, surpassing superficial descriptions.
- **Novelty**: Evaluates the article's capability to introduce relevant, non-directly derived new insights or perspectives beyond the user's initial intent.

We further validate the reliability of this automatic evaluation by conducting a consistency study with human judgments, as detailed in Appendix A.2.

4. **Document Coverage (Coverage %)**: This metric quantifies the extent to which the content of the generated article is actually supported by the knowledge repository, directly addressing the challenge of high fidelity and low hallucination. The calculation procedure is as follows:

   i) Divide the generated article into logical paragraphs.
   ii) For each paragraph, retrieve the two most similar documents from the document collection within the knowledge repository.
   iii) Prompt the LLM to strictly judge whether the retrieved documents support the paragraph's description (i.e., content is relevant and conflict-free).
   iv) If at least one document is judged to support the paragraph, the paragraph is considered supported; otherwise, it is considered unsupported.
   v) Calculate the percentage of supported paragraphs relative to the total number of paragraphs in the article.

4.2 Baseline Models

To accurately evaluate ConvergeWriter's performance, we rigorously follow the incremental comparison principle, selecting representative Retrieval-Augmented Generation (RAG) baselines for comparison:

- **Direct RAG**(Lewis et al., 2020): The foundational paradigm retrieves relevant documents once based on the topic and directly inputs them to the LLM for single-step generation of the complete article.
- **Two-Stage RAG**: A two-stage approach: First, it generates a retrieval-augmented outline; then, it expands content section by section based on the outline (retrieving topic-relevant documents again for each section); finally, it concatenates the sections into a full article.
- **STORM**(Shao et al., 2024): It delves into topic details through simulated multi-perspective dialogues and systematically integrates information to produce a structured outline. The article is subsequently written section by section according to this outline.
- **OmniThink**(Xi et al., 2025): It extends knowledge breadth via an "information tree," deepens cognitive associations through a "concept pool," and dynamically broadens knowledge boundaries to accomplish concept abstraction, outline generation, and citation-aware paragraph writing.

These baselines span from basic retrieval augmentation (**Direct RAG**(Lewis et al., 2020)) and structured staged augmentation (**Two-Stage RAG**) to state-of-the-art complex knowledge planning and integration methods (**STORM**(Shao et al., 2024), **OmniThink**(Xi et al., 2025)), aiming to isolate and assess the impact of diverse knowledge utilization strategies and content organization structures on long-text generation.

4.3 Experimental Details

To systematically evaluate the generalization performance of **ConvergeWriter** across varying language model scales and reasoning capabilities, we conduct comprehensive experiments on two representative model configurations:

1. **Qwen3-14B (Basic Reasoning Configuration)**(Yang et al., 2025): We disable the model's deep thinking mode to assess method performance under the model's fundamental capabilities.
2. **Qwen3-32B (Enhanced Reasoning Configuration)**(Yang et al., 2025): We enable the

Methods	Length	Cited Docs	Rubric Grading					Coverage %
			Relevance	Breadth	Depth	Novelty	AVG	
Qwen3-14B without Reasoning								
Direct RAG	4,848.95	4.27	3.92	3.90	3.79	2.81	3.60	36.67
Two-Stage RAG	**54,744.76**	3.63	3.77	4.22	4.39	3.98	4.14	7.12
Storm	14,732.00	4.97	4.80	4.58	4.35	2.92	4.16	24.91
OmniThink	27,376.73	6.03	4.43	4.12	3.79	3.35	3.92	53.88
ConvergeWriter (ours)	25,913.52	**9.08**	**4.93**	**4.95**	**4.97**	**4.22**	**4.77**	**80.14**
Qwen3-32B with Reasoning								
Direct RAG	9,252.43	8.67	4.93	4.93	4.92	3.72	4.62	61.96
Two-Stage RAG	**68524.68**	3.54	3.54	4.46	4.54	4.08	4.16	2.08
Storm	24,105.05	7.95	4.78	4.75	4.75	3.85	4.53	35.84
OmniThink	35,917.12	**9.52**	4.43	4.38	4.37	4.13	4.33	43.22
ConvergeWriter (ours)	22,629.35	7.98	**4.97**	**4.95**	**4.93**	**4.58**	**4.86**	**70.51**

Table 1: Evaluation results on WildSeek dataset comparing ConvergeWriter with baselines. Metrics include: average article length (Length), average cited documents per article (Cite Docs), LLM-based rubric scores across four dimensions + average (Rubric Grading), and average document coverage percentage (Coverage %).

model's deep thinking mode to verify whether methods can effectively harness and guide the enhanced reasoning capabilities of more powerful models to yield superior results.

All experiments enforce a maximum context length limit of 24K tokens. For baseline methods involving multi-round interactions (e.g., **STORM**(Shao et al., 2024), **OmniThink**(Xi et al., 2025)), the per-round context length is capped at 12K tokens to prevent overflow and ensure fair comparison. Key experimental settings include:

- **Knowledge Source and Retrieval**: We uniformly employ the Wikipedia official retrieval API as the external knowledge source. All retrieval-augmented methods (including **ConvergeWriter** and RAG baselines) acquire knowledge through this API.
- **Embedding and Reranking Models**: The embedding model (Qwen3-Embedding-0.6B (Zhang et al., 2025)) and reranker model (Qwen3-Reranker-0.6B (Zhang et al., 2025)) are consistently used across all methods to maintain retrieval process integrity.
- **Computational Resources**: All experiments are executed on three NVIDIA A6000 GPUs.

4.4 Experimental Results

Our experimental results are detailed in Table 1. The results demonstrate that ConvergeWriter exhibits significant advantages across core metrics, validating the effectiveness of its design principles. Specifically:

(1) Bottom-Up Data-Driven Strategy Significantly Enhances Verifiability and Credibility
The Document Coverage of ConvergeWriter on the 14B and 32B models reaches 80.14% and 70.51%, respectively, significantly outperforming the baselines. This confirms that its core strategy—"retrieval-first for knowledge, clustering for structure"—can a priori build a traceable knowledge cluster framework. This ensures the generated content is strictly grounded in retrieved evidence, effectively mitigating content hallucination while enhancing credibility and verifiability. In contrast, traditional "top-down" approaches, due to their inherently "hypothesis-driven" nature, often lead to conceptualization that diverges from the actual content of the knowledge base, resulting in generated content lacking reliable source support.

(2) Objective Generation Based on Clustered Knowledge Structures Improves Overall Article Quality
Compared to other structure-oriented methods like STORM(Shao et al., 2024) and OmniThink(Xi et al., 2025), ConvergeWriter's core advantage lies in its article structure being driven by objective data (derived from document clustering results) rather than subjective model conceptualization. This enables the article's breadth and depth to more authentically reflect the full spectrum of the knowledge base. Notably, ConvergeWriter also achieved the highest scores on the Novelty metric (4.22 and

4.58). This indicates that by clustering documents, the model can uncover latent, non-intuitive connections between knowledge points, generating more insightful perspectives rather than merely combining information.

(3) Relevance-Expanding Knowledge Retrieval Optimizes Citation Breadth and Content Focus

On the 14B model, the number of Cited Docs for ConvergeWriter reaches 9.08, demonstrating that its iterative retrieval and clustering mechanism effectively broadens the scope of knowledge exploration. Simultaneously, it generates articles of moderate length, achieving content conciseness while ensuring high quality and high credibility. In contrast, although the Two-Stage RAG method produces extremely long text, its low Support Document Coverage and mediocre ratings indicate substantial content redundancy and deviation from knowledge sources, resulting in ineffective information.

(4) Framework Design Confers Strong Model Generalization Capability

Experimental results exhibit consistent advantages across model scales (14B vs. 32B) and inference modes (standard vs. Deep Thinking Mode). Although performance generally improves with the more capable 32B model (with Deep Thinking Mode enabled), ConvergeWriter maintains its performance lead. This confirms that its success stems from the inherent advantages of the framework design, rather than reliance on specific model capabilities, demonstrating strong generalization and portability.

4.5 Further Analysis

To further examine the effectiveness of clustering in constructing a knowledge framework, we conduct an ablation study. Specifically, we remove the clustering component from ConvergeWriter (denoted as "w/o Clustering") and instead divide retrieved documents sequentially into five equal parts, while keeping all other components unchanged. The results are shown in Figure 3.

As illustrated, removing clustering significantly degrades article quality, with the average score dropping from 4.86 to 4.58. The largest decline occurs in "Novelty", decreasing from 4.58 to 3.60. We attribute this to the lack of effective knowledge organization. ConvergeWriter groups unstructured documents into semantically coherent clusters with high cohesion and low coupling, providing a focused basis for outline generation and ensuring thematic consistency. In contrast, the sequential partitioning in "w/o Clustering" is arbitrary and disrupts semantic relationships among documents, leading to disorganized knowledge and forcing the model to concatenate weakly related content, which harms coherence and novelty.

A similar trend is observed in "Breadth" (4.95→4.87). Clustering helps identify diverse thematic dimensions within the document set, enabling comprehensive coverage of relevant subtopics. Sequential splitting instead fragments related concepts and limits the model's ability to integrate information from multiple perspectives.

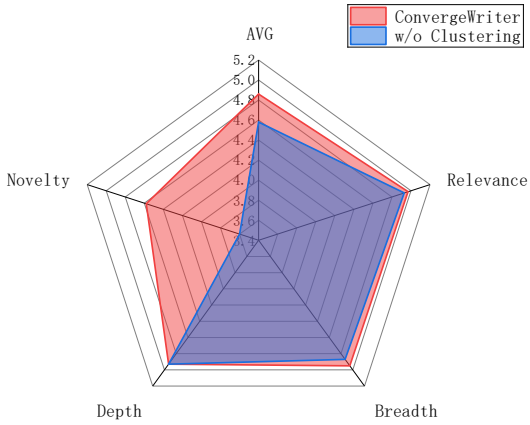

Figure 3: Clustering ablation: Sequential splitting fragments knowledge organization, causing significant novelty decline (4.58→3.60) and incoherent output.

In summary, clustering-based pre-construction of a knowledge framework is crucial for generating high-quality structured long-form text, forming a key foundation of the proposed method.

5 Conclusion

This paper introduces ConvergeWriter, a bottom-up long text generation framework for addressing content credibility challenges in external knowledge base utilization. Its retrieval-first, clustering-structured mechanism constructs a knowledge framework grounded in available evidence via preliminary retrieval and clustering, ensuring generated content remains anchored within knowledge base boundaries. This design guarantees traceability and mitigates hallucinations. Experimental results show that ConvergeWriter achieves performance on par with or superior to state-of-the-art methods across model scales, demonstrating the effectiveness of this knowledge-driven paradigm in closed-knowledge scenarios.

6 Limitations

Despite its effectiveness, ConvergeWriter has several limitations. First, the framework is designed for knowledge-constrained generation settings where the underlying corpus serves as a relatively reliable knowledge source. Extending the approach to highly open or noisy environments may require additional mechanisms for retrieval quality control.

Second, although clustering provides an effective mechanism for organizing retrieved knowledge, the resulting cluster granularity may not always perfectly align with the ideal discourse structure of the target article.

Finally, the retrieval-first pipeline introduces additional computational overhead due to multi-round retrieval, document filtering, clustering, and hierarchical summarization, resulting in higher generation latency compared with top-down approaches.

7 Ethics Statement

This work focuses on improving the reliability of long-form document generation by grounding LLM outputs in external knowledge sources. By constraining generation within retrieved evidence, our framework aims to reduce hallucinations and improve factual consistency, which may help mitigate risks associated with misinformation in automated text generation.

Our experiments use publicly available corpora (e.g., Wikipedia) and do not involve personal or sensitive data. A small-scale human evaluation was conducted for research purposes to assess generation quality, where annotators compared model outputs without providing personal information. However, as with other retrieval-augmented generation systems, the quality of generated content still depends on the underlying knowledge sources, and applications in high-stakes domains should rely on trustworthy corpora and appropriate verification mechanisms.

AI-based writing assistants were used to help improve the clarity and language of the manuscript. All technical content was written and verified by the authors.

References

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.

Yushi Bai, Jiajie Zhang, Xin Lv, Linzhi Zheng, Siqi Zhu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2025. Longwriter: Unleashing 10,000+ word generation from long context LLMs. In *The Thirteenth International Conference on Learning Representations*.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Comput. Surv.*, 55(12).

Yucheng Jiang, Yijia Shao, Dekun Ma, Sina Semnani, and Monica Lam. 2024. Into the unknown unknowns: Engaged human learning through participation in language model agent conversations. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 9917–9955, Miami, Florida, USA. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

Xiaoxi Li, Jiajie Jin, Guanting Dong, Hongjin Qian, Yutao Zhu, Yongkang Wu, Ji-Rong Wen, and Zhicheng Dou. 2025. Webthinker: Empowering large reasoning models with deep research capability. *Preprint*, arXiv:2504.21776.

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.

Peter J. Rousseeuw. 1987. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D Manning. 2024. RAPTOR: Recursive abstractive processing for tree-organized retrieval. In *The Twelfth International Conference on Learning Representations*.

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 *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6252–6278, Mexico City, Mexico. Association for Computational Linguistics.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Richard James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2024. REPLUG: Retrieval-

augmented black-box language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8371–8384, Mexico City, Mexico. Association for Computational Linguistics.

Michael D. Skarlinski, Sam Cox, Jon M. Laurent, James D. Braza, Michaela Hinks, Michael J. Hammerling, Manvitha Ponnapati, Samuel G. Rodriques, and Andrew D. White. 2024. Language agents achieve superhuman synthesis of scientific knowledge. *Preprint*, arXiv:2409.13740.

Kaiyang Wan, Honglin Mu, Rui Hao, Haoran Luo, Tianle Gu, and Xiuying Chen. 2025. A cognitive writing perspective for constrained long-form text generation. *CoRR*, abs/2502.12568.

Qiyao Wang, Shiwen Ni, Huaren Liu, Shule Lu, Guhong Chen, Xi Feng, Chi Wei, Qiang Qu, Hamid Alinejad-Rokny, Yuan Lin, and Min Yang. 2024a. Autopatent: A multi-agent framework for automatic patent generation. *Preprint*, arXiv:2412.09796.

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. 2024b. Autosurvey: Large language models can automatically write surveys. In *Advances in Neural Information Processing Systems*, volume 37, pages 115119–115145. Curran Associates, Inc.

Zekun Xi, Wenbiao Yin, Jizhan Fang, Jialong Wu, Runnan Fang, Ningyu Zhang, Jiang Yong, Pengjun Xie, Fei Huang, and Huajun Chen. 2025. Omnithink: Expanding knowledge boundaries in machine writing through thinking. *Preprint*, arXiv:2501.09751.

Renjun Xu and Jingwen Peng. 2025. A comprehensive survey of deep research: Systems, methodologies, and applications. *Preprint*, arXiv:2506.12594.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Kevin Yang, Dan Klein, Nanyun Peng, and Yuandong Tian. 2023. Doc: Improving long story coherence with detailed outline control. *Preprint*, arXiv:2212.10077.

Murong Yue. 2025. A survey of large language model agents for question answering. *Preprint*, arXiv:2503.19213.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *Preprint*, arXiv:2506.05176.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, and 3 others. 2025. A survey of large language models. *Preprint*, arXiv:2303.18223.

A Experimental Details

A.1 Hyperparameter Settings

For LLM hyperparameter settings, in text generation tasks, we set `temperature=0.9` and `top_p=0.9` to meet the need for richer and more vivid expressions. During text evaluation, we set `temperature=0.1` to enhance the rigor and reproducibility of evaluation results.

A.2 Human–LLM Evaluation Consistency

To examine whether the LLM-based evaluation used in our main experiments aligns with human judgment, we conduct a small-scale human study. We randomly sample 32 article pairs from the test set, where each pair contains two articles generated by different methods under the same topic. Eight researchers with computer science backgrounds independently perform blind pairwise comparisons across the four evaluation dimensions: *Relevance*, *Breadth*, *Depth*, and *Novelty*. For each dimension, annotators indicate whether Article A is better, Article B is better, or the two are tied.

Since the LLM evaluator assigns discrete scores (1–5), ties frequently occur when converting scores into pairwise preferences. To fairly measure consistency, we adopt a *strict filtering* strategy that retains only samples where the LLM produces a clear preference. The resulting agreement statistics are shown in Table 2. Overall, the LLM evaluation achieves an agreement rate of 0.69 with human judgments, with Kendall's τ of 0.46 and Cohen's κ of 0.42. Notably, consistency is higher when the score margin between two articles is larger, indicating that the LLM evaluator is particularly reliable when quality differences are more pronounced.

Metric	Agreement	Kendall τ	Cohen κ	N
Relevance	0.60	0.46	0.31	10
Breadth	0.71	0.49	0.46	21
Depth	**0.78**	**0.64**	**0.58**	18
Novelty	0.64	0.32	0.30	22
Overall	**0.69**	**0.46**	**0.42**	71

Table 2: Consistency between human evaluation and LLM-based scoring under strict filtering (excluding LLM tie cases).

These results support the validity of using LLM-based evaluation for large-scale comparison.

A.3 Details of Further Analysis

In Section 4.5 (Further Analysis), we evaluated the performance gap before and after removing the clustering module in ConvergeWriter. Specific metric scores are shown in Table 3:

Methods	Relevance	Breadth	Depth	Novelty	AVG
ConvergeWriter	4.97	4.95	4.93	4.58	4.86
w/o Clustering	4.93	4.87	4.93	3.60	4.58

Table 3: Comparison of w/o Clustering

B Prompts Used

DOCUMENT_FILTERING_PROMPT

```
Topic: {topic}

Wikipedia Article Title: {article_title}
Wikipedia Article Summary: {article_summary}

Is this Wikipedia article relevant and potentially
helpful for writing an article about the main topic "
{topic}"?
Answer strictly with "Yes" or "No".

Answer:
```

Figure 4: The prompt of document filtering.

DOCUMENT_SUMMARIZATION_PROMPT

```
Summarize the key information from the following
text, focusing on aspects relevant to the broader
context it might be used in. Keep the summary
concise, around 100-150 words.

Text:
{document_content}

Summary:
```

Figure 5: The prompt of document summarization.

CLUSTER_SUMMARY_PROMPT

```
You are given summaries of several Wikipedia
articles that have been grouped together because they
cover related aspects of a larger topic.

Summaries:
{summaries_list}

1. Provide a brief introduction (1–2 sentences)
describing the main theme or connection between
these articles in this cluster.
2. Summarize the key information points covered by
the documents in this cluster. List them concisely.

Output format:
Introduction: [Your introduction here]
Key Information Points:
- [Point 1]
- [Point 2]
- ...

Cluster Content:
```

Figure 6: The prompt of cluster summarization.

KEYWORD_GENERATION_PROMPT

Analyze the main topic: "{topic}"

Think step-by-step about how to write a comprehensive article on this topic. Consider the following points:
- Key aspects: What are the core elements and components of this topic?
- Concepts: Are there any essential ideas or theories closely related to it?
- Historical background: What historical events or developments have shaped this topic?
- Related fields: What other fields or disciplines are connected to this topic?
- Important figures: Who are the key people or organizations related to this topic?
- Controversies or debates: Are there any ongoing discussions or disagreements about this topic?

Based on these considerations, generate a list of specific and relevant keywords (titles) that would be useful for gathering information to write a Wikipedia article about "{topic}". These keywords should be:
- Specific
- Likely to correspond to existing Wikipedia pages
- Covering different aspects such as history, concepts, figures, controversies, and related fields

Output the keywords as a Python list of strings. For example:

Example:
Topic: "Climate Change"

Keywords:
["Global Warming", "Greenhouse Gases", "Climate Models", "Environmental Policy", "Carbon Emissions", "Renewable Energy", "Paris Agreement", "Climate Change Debate", "Scientific Consensus on Climate Change", "International Climate Negotiations"]

Topic: {topic}

Keywords:

Figure 7: The prompt of keyword generation.

OUTLINE_GENERATION_PROMPT

Generate a structured article outline in English based on the main topic and the provided cluster summaries. The outline must be logical, coherent, and demonstrate a clear narrative progression.

Topic: "{topic}"

Cluster Summaries (each summary block separated by a blank line):
{cluster_summaries}

Core Instructions for Content Generation:
1. **Determine Optimal Chapter Order:** First, analyze all cluster summaries to determine the most logical sequence for the chapters. You have the authority to reorder the provided clusters to build the most coherent narrative (e.g., from foundational concepts to advanced topics, or chronologically).
2. **Prioritize Logic over Completeness:** Your primary goal is to create a logical narrative flow, not to include every detail from the summaries.
3. **Select and Synthesize:** For each cluster, identify the core themes. Select only the most significant and relevant points that support these themes. Discard minor, redundant, or tangential information.
4. **Group and Structure:** Logically group the selected points into subsections (### Section X.Y). Create meaningful, descriptive titles for these subsections that reflect the content within.
5. **Ensure Coherence:** The points within each subsection should be closely related, and the subsections themselves should follow a logical progression within the chapter.

Format Requirements:
1. Main chapters prefixed with "## Chapter X: Title" (X starts from 1).
2. Subsections prefixed with "### Section X.Y: Title" (Y is the subsection number).
3. Bullet points start with "- " and must contain no more than one concise sentence each.
4. The outline must begin with "## Chapter 1: Introduction" and end with "## Chapter N: Conclusion."
5. The number of main chapters between Introduction and Conclusion (i.e., Chapters 2 through N–1) must equal the number of cluster summaries provided.
6. Use each cluster's leading title (text before the first colon ":") as the chapter title for that cluster, regardless of its new position.
7. **Numbering must be continuous and hierarchical based on the final chapter order you decide upon.** For example, if you move the third cluster to become Chapter 2, its subsections must be numbered 2.1, 2.2, etc.

Article Outline:

Figure 8: The prompt of outline generation.

CHAPTER_GENERATION_PROMPT

Act as an expert writer. Your current task is to write a single chapter for an article on the topic: "{topic_title}".

Adhere strictly to the provided outline for this chapter.

Chapter Outline to Follow:
{chapter_outline}
Knowledge Base:
You must use the following reference documents as your primary source of information. Write the chapter based on the content that is relevant to the outline. If a point in the outline is not covered by the documents, you may use your own knowledge. You must ignore any information in the documents that is not related to the current chapter's outline.

{knowledge_base if knowledge_base.strip() else "No reference documents were found for this chapter. Please rely on your own knowledge."}
Your Instructions:
- Write a comprehensive and well-structured chapter in English.
- Ensure there are smooth transitions between different points of the outline.
- You MUST include the chapter title (e.g., "## Chapter 2...") in your output.

Figure 9: The prompt of chapter generation.

INTRODUCTION_GENERATION_PROMPT

Act as an expert writer. You are provided with the main body of an article about "{topic_title}".
Your task is to write a compelling introduction for this article.

The introduction should follow this structure:
{intro_outline}

Article Main Body:
Here is the main content of the article that
{article_body}
Your Instructions:
- The introduction should set the stage, provide necessary background, and state the article's purpose, leading smoothly into the first chapter.
- Keep the title.

Figure 10: The prompt of introduction generation.

CONCLUSION_GENERATION_PROMPT

Act as an expert writer. You are provided with a nearly complete article (introduction and main body) about "{topic_title}".
Your task is to write a strong conclusion.

The conclusion should follow this structure:
{conclusion_outline}

Full Article (Introduction and Body):
{article_with_intro}
Your Instructions:
- The conclusion should summarize the key findings and discuss their implications, providing a satisfying end to the article.
- Keep the title.

Figure 11: The prompt of conclusion generation.

ARTICLE_POLISH_PROMPT

Act as an expert editor. Your task is to review and polish the following article about "{topic_title}".

Full Draft of the Article:
{full_article_draft}
Your Instructions:
- Your goal is to enhance the article's quality without altering its core meaning or adding new facts.
- Improve the flow and transitions between paragraphs and sections.
- Correct any grammatical errors, spelling mistakes, or awkward phrasing.
- Eliminate redundant sentences or ideas to make the text more concise.
- Ensure the entire article is coherent, logically structured, and reads professionally.
- Your response must be ONLY the final, polished article text. Do not add any commentary before or after the article.

Figure 12: The prompt of article polish.