

ConvergeWriter: Data-Driven Bottom-Up Article Construction

Binquan Ji, Jiaqi Wang, Ruiting Li, Xingchen Han, Haibo Luo,
Yiyang Qi, Shichao Wang, Yifei Lu, Yuantao Han, Feiliang Ren*

School of Computer Science and Engineering,
Northeastern University, Shenyang 110819, China
jibinquan@foxmail.com, renfeiliang@cse.neu.edu.cn

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., 2026; Yue, 2025). However, effectively incorporating external knowledge to produce detailed, credible, and hallucination-free long-form documents remains challenging. This task

*Corresponding author.

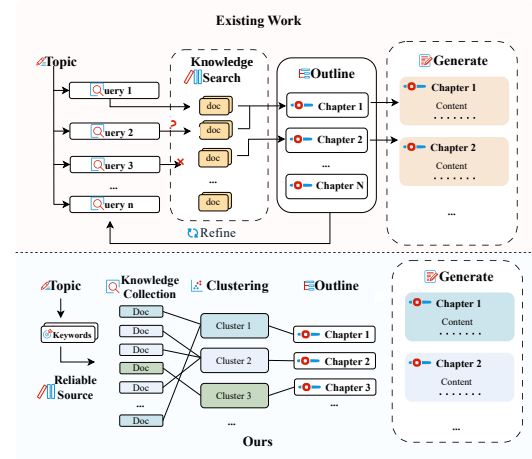


Figure 1: Comparison of article-construction pipelines. Top-down methods generate an outline or questions from the input topic and then retrieve evidence for the predefined structure. ConvergeWriter instead retrieves and clusters evidence to estimate the corpus’s knowledge boundary, induces an outline from the resulting knowledge clusters, and generates sections grounded in those clusters.

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).

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: outlining a structure or raising questions, with retrieved evidence used to refine or support this framework. This approach, however, may misalign with the knowledge base, leading to inefficient retrieval, fragmented content, and reduced credibility.

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 (Sarathi 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. STORM (Shao et al., 2024) decomposes writing into iterative stages, including multi-perspective knowledge gathering, outlining, and drafting. Similarly, OmniThink (Xi et al., 2025) expands knowledge via dynamic “information tree” construction and abstraction, while LongWriter (Bai et al., 2025) separates generation into planning and sequential paragraph writing. Across these approaches, retrieval or knowledge supplementation mainly validates or supplements a predefined outline constructed before the full knowledge scope is known (Bai et al., 2025).

Multi-agent frameworks further extend this paradigm. AutoPatent (Wang et al., 2024a) employs Planner/Author/Reviewer agents guided by a predefined “PGTree,” using Reference-Reviewer-Augmented Generation (RRAG) for local optimization. The cognitive writing framework of Wan et al. (2025) simulates human writing through hierarchical planning and parallel generation with iterative revision. WebThinker (Li et al., 2025) integrates 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.

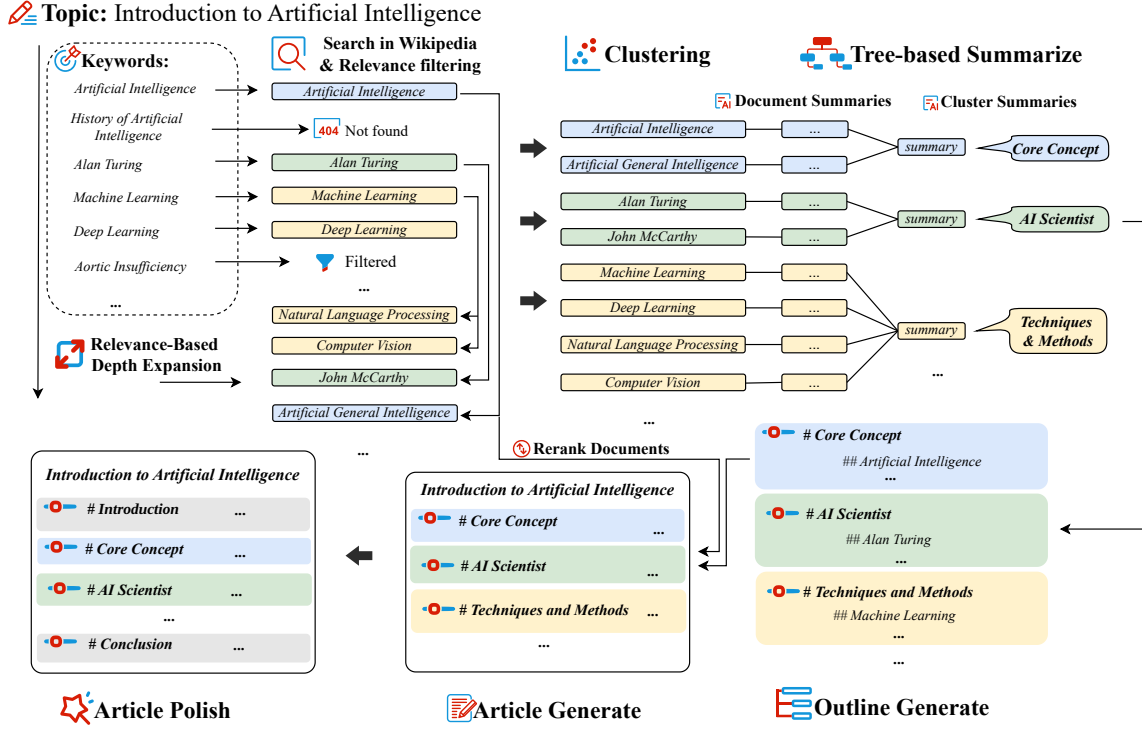


Figure 2: ConvergeWriter’s workflow: (1) iterative relevance-expanding knowledge retrieval and document screening; (2) unsupervised clustering with tree-structured summarization; (3) LLM-based outline generation; and (4) recall-enhanced chapter-wise content generation, introduction/conclusion supplementation,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

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 Figure 2. The method begins with multiple rounds 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, we use a Large Language Model (LLM) $\mathcal{M}$ to generate an initial keyword set based on topic T . Here and throughout, $\mathcal{I}$ denotes the prompt or instruction template used by the corresponding LLM module:

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

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:

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

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:

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

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 coefficient (Rousseeuw, 1987) 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 $\bar{s}(k)$:

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

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 $\{C_1, C_2, \dots, C_{k^*}\}$.

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:

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

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 .

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

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 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 $\{\text{Sec}_1, \text{Sec}_2, \dots, \text{Sec}_K\}$. 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:

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

After generating all main sections Sec_i^* independently, we concatenate them to form the article body $\mathcal{A}_{\text{draft}}$. We then use the topic and $\mathcal{A}_{\text{draft}}$ to generate the “Introduction” section Sec_1^* and “Conclusion” section Sec_K^* separately. Combining these sections with the article body yields the complete draft $\mathcal{A}_{\text{full_draft}}$.

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 experiments on a fixed subset of topics from the **WildSeek** dataset (Jiang et al., 2024). WildSeek is derived from real users’ information-exploration sessions with the open-web application STORM (Shao et al., 2024), and is designed to evaluate 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 and organize relevant information and generate well-structured, informative articles. This task setup simulates applications in vertical domains (e.g., finance and scientific report writing) with well-defined knowledge boundaries and high information-credibility requirements. Details of the dataset subset and evaluation protocol are provided in Appendix A.2.

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 whitespace-delimited word count of generated articles within the fixed context-length limit of the LLM. This metric directly measures 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 1–5 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 assess the validity of this automatic evaluation method through two complementary consistency studies: a parallel evaluation using DeepSeek-v4-flash as a heterogeneous LLM judge, and a comparison between LLM-based evaluation and human judgments. Detailed experimental procedures and results are provided in Appendices A.4 and A.5.

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) Segment the article into paragraph-level verification units using the deterministic preprocessing rules in Appendix A.2.
 - ii) For each paragraph, retrieve the two most similar documents from the document collection in the knowledge base; an evaluation of the number of retrieved documents is provided in Appendix A.7.

- 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)),

Method	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	<u>4.39</u>	<u>3.98</u>	4.14	7.12
STORM	14,732.00	4.97	<u>4.80</u>	<u>4.58</u>	4.35	2.92	<u>4.16</u>	24.91
OmniThink	<u>27,376.73</u>	<u>6.03</u>	4.43	4.12	3.79	3.35	3.92	<u>53.88</u>
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	<u>8.67</u>	<u>4.93</u>	<u>4.93</u>	<u>4.92</u>	3.72	<u>4.62</u>	<u>61.96</u>
Two-Stage RAG	68,524.68	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	<u>35,917.12</u>	9.52	4.43	4.38	4.37	<u>4.13</u>	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 the WildSeek dataset. Length is the average whitespace-delimited word count; Cited Docs is the average number of cited documents; Rubric Grading reports four LLM-evaluated dimensions and their average; and Coverage is the percentage of supported paragraphs. Bold and underlined values denote the best and second-best results within each model setting, respectively.

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 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) and **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 helps anchor the

generated content in retrieved evidence, reducing unsupported content 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. Additionally, we evaluate claim-level evidence support across all five methods. ConvergeWriter continues to outperform all baselines in terms of claim support rate, further supporting the conclusion above. The complete evaluation protocol and results are reported in Appendix A.3.

(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). Al-

though 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.

Additionally, we conduct a case study of the outputs generated by ConvergeWriter and STORM, as well as topic-level paired significance tests on the LLM-based evaluation results described above; detailed results are provided in Appendices A.8 and A.6, respectively.

4.5 Further Analysis

4.5.1 Clustering Ablation

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.

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.

Method	Wall time (s)	Article words	Model tokens	s/1k words	tokens/1k words
ConvergeWriter	289.1	3,323.1	66,446.6	92.7	21,113.5
Direct RAG	65.7	1,413.7	8,302.1	48.1	6,126.1
Two-Stage RAG	269.4	10,768.6	25,339.9	25.2	2,363.0
STORM	139.5	4,176.0	64,995.4	34.9	16,222.2
OmniThink	343.8	3,310.6	195,347.2	181.0	105,060.6

Table 2: Mean latency, output length, and model-token usage on the auxiliary 10-topic efficiency benchmark. Per-1,000-word metrics are computed for each topic and then averaged.

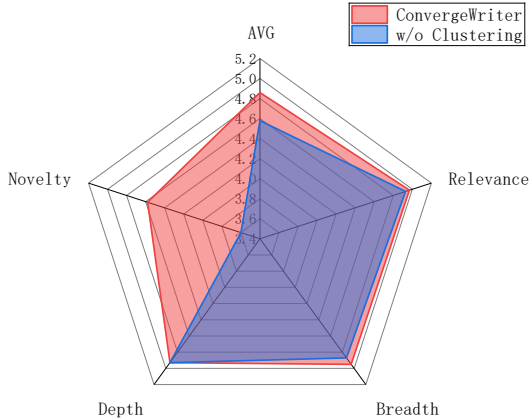


Figure 3: Clustering ablation. Sequential splitting fragments knowledge organization, causing a substantial novelty decline ($4.58 \rightarrow 3.60$) and less coherent output.

4.5.2 Computational Efficiency

To quantify the latency and computational overhead of ConvergeWriter, we benchmark it against four representative baselines on the same 10 topics. Table 2 reports end-to-end wall time and model-token usage together with output length and per-1,000-word measurements; detailed configurations and measurement rules are provided in Appendix A.9.

On a per-1,000-word basis, ConvergeWriter requires 92.7 seconds and 21.1K model tokens, placing its computational cost between STORM (34.9 seconds and 16.2K tokens) and OmniThink (181.0 seconds and 105.1K tokens) among multi-stage long-form writing systems. The additional computation reflects ConvergeWriter’s evidence-oriented pipeline, including multi-round retrieval and filtering, clustering, cluster summarization, reranking, and section-wise generation, which constructs and exploits an evidence-induced article structure throughout drafting. Together with the clustering ablation and the article-quality, Document Coverage, and claim-level support results reported above, these results characterize a fidelity–efficiency trade-off: ConvergeWriter invests additional computation in evidence-grounded organization, while re-

maining substantially below OmniThink in both latency and token consumption. A stage-wise cost breakdown is provided in Appendix A.9.

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 and evidence sufficiency assessment. In this direction, iterative RAG methods for question answering, such as FAIR-RAG (Asl et al., 2025), assess evidence sufficiency for a given query and refine retrieval to address identified evidence gaps. If such query-level refinement were adapted from question answering to section-level evidence assessment in long-form writing, it could complement ConvergeWriter’s corpus-level structure induction.

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.

Acknowledgments

This work is funded by the National Natural Science Foundation of China (No.62576085).

References

- Mohammad Aghajani Asl, Majid Asgari-Bidhendi, and Behrooz Minaei-Bidgoli. 2025. [FAIR-RAG: Faithful adaptive iterative refinement for retrieval-augmented generation](#). *Preprint*, arXiv:2510.22344.
- 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 K ttler, Mike Lewis, Wen-tau Yih, Tim Rockt schel, 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, Yongkang Wu, Ji-Rong Wen, Yutao Zhu, and Zhicheng Dou. 2025. [Webthinker: Empowering large reasoning models with deep research capability](#). In *Advances in Neural Information Processing Systems*, volume 38, pages 133164–133204.
- 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.
- Michael D. Skarlinski, Sam Cox, Jon M. Laurent, James D. Braza, Michaela Hinks, Michael J. Hammerling, Manvitha Ponnampati, Samuel G. Rodrigues, 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. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 9832–9844, Vienna, Austria. Association for Computational Linguistics.

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, Yong Jiang, Pengjun Xie, Fei Huang, Hua-jun Chen, and Ningyu Zhang. 2025. [OmniThink: Expanding knowledge boundaries in machine writing through thinking](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 956–976, Suzhou, China. Association for Computational Linguistics.

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](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3378–3465, Toronto, Canada. Association for Computational Linguistics.

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, Zican Dong, Yupeng Hou, Beichen Zhang, Yingqian Min, Junjie Zhang, Peiyu Liu, Xiaolei Wang, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, and 4 others. 2026. [A survey of large language models](#). *Frontiers of Computer Science*, 20(12):2012627.

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. Article-generation requests use the API default settings for top_k and min_p. During text evaluation, we set temperature=0.1 to enhance the rigor and reproducibility of evaluation results.

A.2 Dataset Subset and Coverage Preprocessing

Dataset subset. WildSeek contains 100 topics. Because running every generation method under multiple model settings is computationally expensive, our main experiments use the same fixed subset of 60 topics for all methods. Consequently, method comparisons in the main experiments are conducted on matched topics.

Coverage preprocessing. We form paragraph-level verification units deterministically. We first split each generated article at Markdown and natural-paragraph boundaries. We then remove headings, empty paragraphs, citation-only fragments, and formatting-only fragments, and merge or discard fragments too short to form independent semantic units. Each remaining paragraph is treated as one verification unit.

A.3 Claim-Level Evidence Support Analysis

To examine the evidential support of generated content at a finer granularity, we conduct a claim-level evidence support analysis across all five methods. We randomly sample 30 topics and use the same sampled topics for all methods. We process all segments of each article, limiting each segment to 8,000 characters, and use DeepSeek-v4-flash to extract up to three atomic factual claims from each segment. For each claim, we rank the saved documents associated with the corresponding method–topic pair using BM25 and provide the top five documents, each limited to 12,000 characters, to DeepSeek-v4-flash for verification. We define the claim support rate as the number of supported claims divided by the number of verified claims. The results are shown in Table 3.

Method	Claims	Supported	Support Rate
Direct RAG	815	582	<u>0.714</u>
Two-Stage RAG	2,603	148	0.057
STORM	1,590	803	0.505
OmniThink	920	472	0.513
ConvergeWriter	818	589	0.720

Table 3: Claim-level evidence support results on the same 30 randomly sampled topics. Claims and Supported denote the numbers of verified and evidence-supported claims, respectively. Bold and underlined values denote the best and second-best support rates, respectively.

As shown in Table 3, ConvergeWriter achieves the highest claim support rate of 0.720, providing further claim-level support for the advantage of its bottom-up, data-driven strategy in terms of verifiability and credibility.

A.4 Parallel Evaluation with a Heterogeneous LLM Judge

We conduct a parallel evaluation of the same articles reported in the Qwen3-32B-with-reasoning block of Table 1 using DeepSeek-v4-flash as a heterogeneous LLM judge. The generated articles, five methods, and fixed set of 60 common topics remain the same across the Qwen and DeepSeek evaluations. Method names are hidden during evaluation, and the same four rubric dimensions—*Relevance*, *Breadth*, *Depth*, and *Novelty*—are used. The evaluation results are presented in Table 4.

Method	Relevance	Breadth	Depth	Novelty	AVG
Direct RAG	4.900	4.133	4.017	<u>2.700</u>	3.938
Two-Stage RAG	3.233	3.517	3.483	2.200	3.108
STORM	4.767	<u>4.400</u>	<u>4.317</u>	2.633	<u>4.029</u>
OmniThink	3.967	3.617	3.300	2.600	3.371
ConvergeWriter	<u>4.817</u>	4.483	4.433	3.000	4.183

Table 4: Evaluation on the same fixed 60-topic article set using DeepSeek-v4-flash as a heterogeneous LLM judge. Bold and underlined values denote the best and second-best results, respectively.

Overall, ConvergeWriter ranks first in three dimensions and in average score under the DeepSeek judge, broadly aligning with the original Qwen evaluation and providing complementary cross-judge evidence.

A.5 Human-LLM Evaluation Consistency

To compare the LLM-based evaluation used in our main experiments with human judgments, we conduct a human evaluation 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. All evaluators were informed that their judgments would be used for research and publication purposes and voluntarily agreed to participate.

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

Since the LLM evaluator assigns discrete scores (1–5), ties can occur when converting scores into pairwise preferences. For the consistency analysis, we apply strict filtering by retaining only dimension-level comparisons in which the two articles receive different LLM scores. The resulting agreement statistics are shown in Table 5. Across the resulting 71 comparisons, the overall agreement rate is 0.69, with Kendall’s τ of 0.46 and Cohen’s κ of 0.42. These results indicate a degree of consistency between LLM-based scores and human judgments under this comparison setting.

A.6 Topic-Level Paired Significance Tests

For the Qwen3-32B-with-reasoning setting, we conduct topic-level paired tests on the same fixed set of 60 common topics. For each topic, we calculate the difference between ConvergeWriter’s average score across the four rubric dimensions and that of each baseline under the original Qwen judge. We report the mean difference, a 95% confidence interval estimated with 10,000 paired bootstrap resamples, wins/losses/ties, and a two-sided sign-test p -value computed over non-tied topic pairs.

As shown in Table 6, all four comparisons yield positive mean differences. The paired bootstrap confidence intervals exclude zero, and the two-sided sign-test p -values are below 0.001 in all cases. These results indicate that, under the original Qwen judge, ConvergeWriter’s average score across the

Method	AVG	Δ	95% Bootstrap CI	Wins/Losses/Ties	Sign-test p
ConvergeWriter	4.858	–	–	–	–
Direct RAG	4.625	+0.233	[+0.113, +0.396]	41/4/15	1.0×10^{-8}
Two-Stage RAG	4.237	+0.621	[+0.325, +0.954]	32/8/20	0.0001822
STORM	4.533	+0.325	[+0.125, +0.575]	32/9/19	0.0004309
OmniThink	4.329	+0.529	[+0.354, +0.717]	42/9/9	3.39×10^{-6}

Table 6: Topic-level paired significance tests for the average score across the four rubric dimensions under the original Qwen judge. Δ denotes ConvergeWriter minus the listed baseline; dashes indicate that the paired-comparison statistics are not applicable to the ConvergeWriter reference row.

four rubric dimensions is significantly higher than that of each baseline in the topic-level paired comparisons.

A.7 Sensitivity to the Number of Retrieved Documents

To examine the effect of the number of retrieved documents on Document Coverage, we use Qwen3-8B with thinking disabled as the verifier and evaluate $k \in \{1, 2, 3, 5\}$ on the same 60 topics for ConvergeWriter, Direct RAG, STORM, and OmniThink. Across different values of k , we use the same paragraph segmentation, BM25 retrieval, and strict support-verification procedure, without truncating paragraphs or retrieved documents. The results are shown in Table 7.

Method	$k = 1$	$k = 2$	$k = 3$	$k = 5$
Direct RAG	27.88	34.85	42.42	40.61
STORM	18.04	24.05	26.73	31.82
OmniThink	28.50	32.05	35.10	32.58
ConvergeWriter	33.45	40.27	52.22	45.73

Table 7: Document Coverage (%) under different numbers of retrieved documents.

As shown in Table 7, ConvergeWriter achieves the highest Coverage at every tested value of k . Meanwhile, increasing k does not yield consistent monotonic improvements. Considering both the evaluation results and computational cost, we use $k = 2$ for the Document Coverage evaluation to balance evidence availability and verification cost: compared with $k = 1$, it provides more evidence while limiting the additional context noise and context-length risks associated with larger values of k .

A.8 Qualitative Case Study of Generated Outputs

To provide a concrete qualitative view of the generated outputs, we examine ConvergeWriter and STORM on the topic ‘‘Artificial Intelligence

and Copyright Dilemma.’’ On this topic, ConvergeWriter received an average score of 4.5 from the DeepSeek-v4-flash judge across the four evaluation dimensions, and the claim-level verifier classified 30 of 33 extracted claims as supported by its saved retrieved evidence (0.909); the corresponding results for STORM were 4.0 and 37 of 59 (0.627). Table 8 compares their outline and output patterns with the assessments against their respective saved evidence pools.

A.9 Computational Efficiency Details

We conduct the auxiliary efficiency benchmark on the same set of 10 topics for all five methods. To control variation arising from model context limits, online Wikipedia retrieval, and external service concurrency, all methods use DeepSeek-v4-flash for LLM calls and retrieve from a dense index of the preprocessed English Wikipedia corpus released for HotpotQA, derived from the October 1, 2017 Wikipedia dump and hosted on a local server. Wall time denotes the end-to-end elapsed time for processing a single topic, and model-token usage is the sum of prompt and completion tokens. Seconds and model tokens per 1,000 words are computed separately for each topic and then averaged across the 10 topics. Each method retains the native internal parallelism and output-length configuration used in its benchmark implementation.

The phase-level breakdown shows that 35.8% of ConvergeWriter’s end-to-end time is devoted to retrieval, knowledge structurization, and outline generation, while 64.2% is devoted to article generation. Among the major timed components, reranking and cluster summarization have the largest wall times. This distribution complements the fidelity–efficiency trade-off discussed in Section 4.5.2.

Case Aspect	STORM: Top-Down Outline and Output	Evidence-Pool Assessment	ConvergeWriter: Retrieval-First Outline and Output
Regulatory requirements	The outline includes “AI Act transparency/accountability provisions,” followed by the claim that the EU AI Act mandates documentation of training data for high-risk systems.	The saved retrieved documents do not explicitly mention this training-data documentation requirement.	The article organizes this discussion under broader sections on “AI and Generative Technologies” and “Regulatory and Ethical Challenges,” without making this specific claim.
U.S. copyright decisions	The outline includes “U.S. Copyright Office guidelines on AI-generated content,” and the article states that the U.S. Copyright Office rejected an AI-generated art portfolio in 2023.	The saved retrieved documents do not mention this specific 2023 rejection.	The article discusses general issues of authorship and ownership without referring to this specific institutional event.
Legal and policy implications	The outline includes “Future Directions,” “global harmonization,” and “liability for AI-generated infringement,” and the article makes claims about future disputes and rights allocation for AI-generated works.	Several future-impact and policy-implication claims in these sections are not explicitly supported by the saved retrieved documents.	The article organizes the discussion around generative AI, copyright frameworks, and moral rights and authorship drawn from its evidence clusters.
Case-level summary	The generated article presents a broad legal and policy discussion.	Eighteen unsupported claims fall into at least one of the analyzed categories: ethical issues, future impacts, or policy implications.	The article uses a broader structure centered on retrieved evidence clusters.

Table 8: Qualitative comparison of the outlines and generated outputs of ConvergeWriter and STORM on “Artificial Intelligence and Copyright Dilemma.” Evidence-pool assessments indicate whether the displayed STORM statements are explicitly supported by STORM’s saved retrieved evidence.

Pipeline phase / major timed component	Mean wall (s)	Share	Model tokens	LM calls
Phase I	103.6	35.8%	41,852.3	154.9
Retrieval and filtering	11.1	—	—	—
Expansion-keyword generation	4.4	—	—	—
Embedding and clustering	10.9	—	—	—
Cluster summarization	52.6	—	—	—
Phase II	185.5	64.2%	24,594.3	5.9
Reranking	87.2	—	—	—

Table 9: Phase-level and major-component computational breakdown of ConvergeWriter on the auxiliary 10-topic efficiency benchmark. Phase I comprises relevance-expanding retrieval, knowledge structurization, and structured knowledge mapping-based outline generation; Phase II comprises outline–knowledge cluster joint-driven section-wise retrieval and article generation. Only the principal timed components are shown; minor pipeline operations are omitted.

A.10 Details of Further Analysis

ConvergeWriter. Specific metric scores are shown in Table 10.

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 10: Ablation results with and without document clustering. Bold values indicate the better result in each column.

In Section 4.5, we evaluated the performance gap before and after removing the clustering module in

B Prompts Used

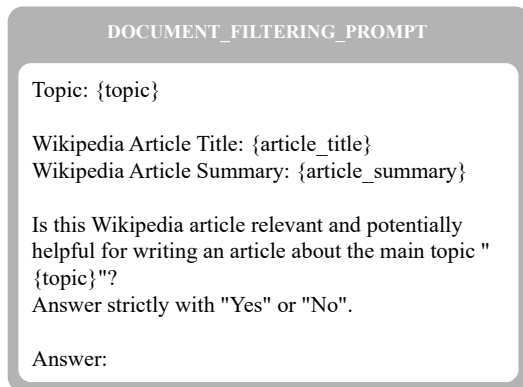


Figure 4: Prompt for document filtering.

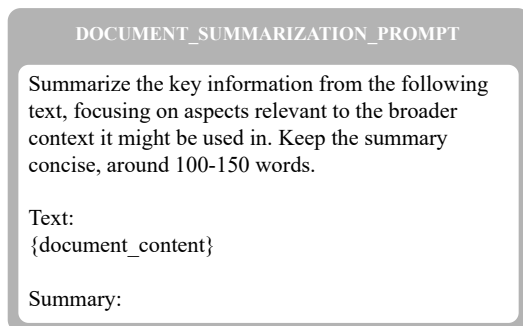


Figure 5: Prompt for document summarization.

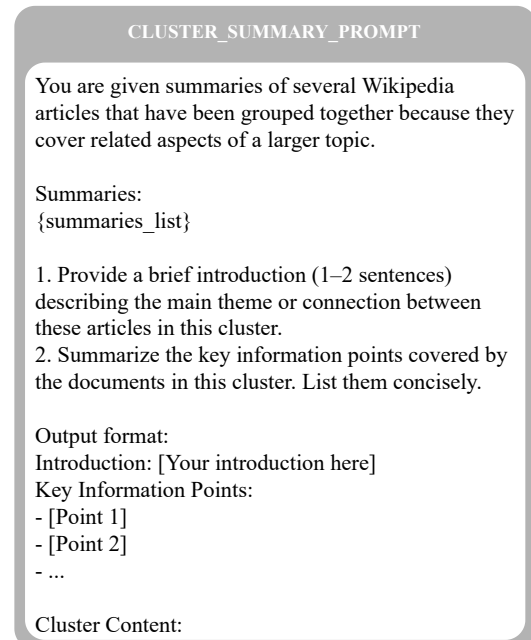


Figure 6: Prompt for cluster summarization.

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: Prompt for 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: Prompt for 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: Prompt for 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: Prompt for introduction generation.

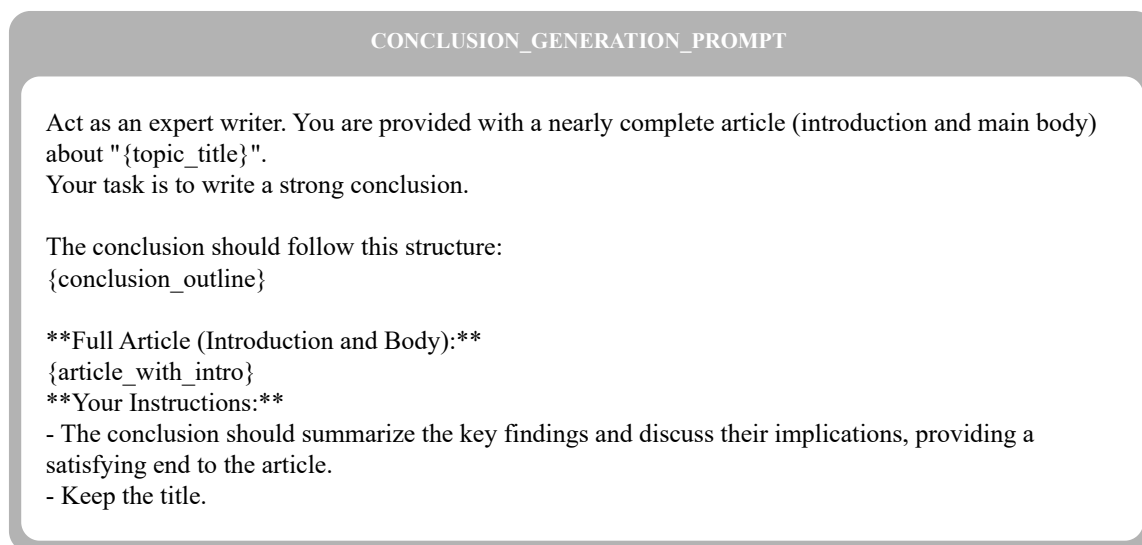


Figure 11: Prompt for conclusion generation.

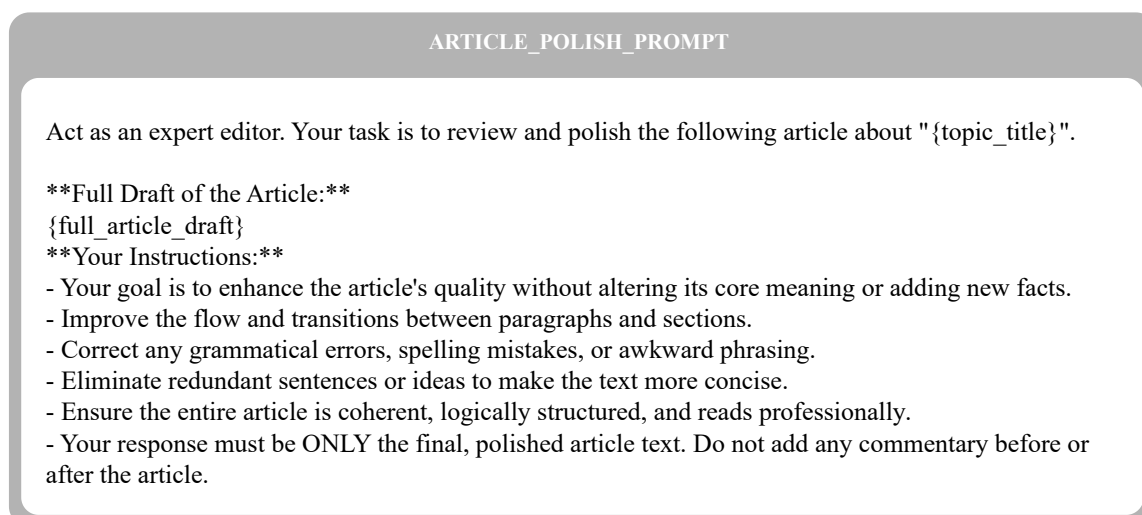


Figure 12: Prompt for final article polishing.