Select, Read, and Write: A Multi-Agent Framework of Full-Text-based Related Work Generation

Anonymous ACL submission

Abstract

Automatic related work generation (RWG) can save people's time and effort when writing a draft of related work section (RWS) for further revision. However, existing methods for RWG always suffer from shallow comprehension due to taking the limited portions of references papers as input and isolated explanation for each reference due to ineffective capturing the relationships among them. To address these issues, we focus on full-text-based RWG task and propose a novel multi-agent framework. Our framework consists of three agents: a *selector* that decides which section of the papers is going to read next, a *reader* that digests the selected section and updates a shared working memory, and a *writer* that generates RWS based on the final curated memory. To better capture the relationships among references, we also propose two graph-aware strategies for selector, enabling to optimize the reading order with constrains of the graph structure. Extensive experiments demonstrate that our framework consistently improves performance across three base models and various input configurations. The graph-aware selectors outperform alternative selectors, achieving state-of-the-art results. The code and data will be available.

1 Introduction

With the exponential growth of academic publications (Wang et al., 2024a), automatic related work generation (RWG) becomes more and more attractive to research communities because it can save time and effort in preparing the first draft of the related work section (RWS) (Şahinuç et al., 2024; Martin-Boyle et al., 2024). Although the RWG task has a long history (Hoang and Kan, 2010) and the advancement of LLMs significantly improves the general ability of text understanding and generation, writing a good RWS is not trivial. Even for experienced researchers, they have to spend a bunch of time to draft the RWS after intensive reading of all references. They need to deeply comprehend the similarities and differences between references, organize them in a reasonable taxonomy, and position the current work by pointing out its novelty. However, existing methods are far from being as excellent as experienced researchers in writing RWS. There are at least two main challenges: 1) misinterpretations or hallucinations due to using limited portions of references (*C1*) and 2) isolated explanation for each reference due to ineffective exploitation of the relationships (*C2*).

C1. Due to the limitations of input window sizes in language models, previous methods for RWG always rely on limited portions of references, such as abstracts (AbuRa'ed et al., 2020; Ge et al., 2021; Li et al., 2022; Mandal et al., 2024), introduction and conclusion (Chen and Zhuge, 2019; Deng et al., 2021), related work (Xing et al., 2020; Ge et al., 2021), or retrieved text spans (Li et al., 2023; Li and Ouyang, 2024), rather than leveraging the full texts. The lack of rich full-text information often prevents models from fully capturing the content and relationships among references, leading to frequent misinterpretations and hallucinations (Xu et al., 2024). However, full-text-based RWG task faces the challenge of limited context window size. It often requires the inclusion of numerous lengthy references. Although models with long context windows have emerged (such as GPT-4o, 128K-token), directly feeding all textual data into the model in a single pass is not optimal. These models face diminishing performance when approaching their maximum context window (Liu et al., 2024).

C2. A high-quality RWS in academic writing needs to provide precise and in-depth comparisons across reference papers, highlight the novelty of the paper being written, and avoid isolated explanation of each reference. These criteria underscore an essential aspect of RWG: capturing and explaining the relationships among references. However, this is a common struggle in previous models, where

loose relationships among references and isolated explanations for each reference are frequent issues (Li and Ouyang, 2024). While some works leverage graph structure (Chen et al., 2021; Wang et al., 2022) to model inter-paper relationships, they integrate graph structures implicitly, failing to effectively address the aforementioned issues.

We overcome the two challenges by proposing a multi-agent framework (*C1*) and a graph-aware selector within the framework (*C2*). We design our framework as a system comprising three agents: a *selector*, a *reader*, and a *writer*. The first two agents work collaboratively and iteratively process input content while maintaining a shared working memory. The *selector* decides the reading order of papers' sections, and the *reader* digests the selected content and updates the memory. Then the *writer* generates the RWS based on the final curated memory. To better capture the relationships among references, we introduce the graph structure within our framework. We build two kinds of relationship graphs: a co-occurrence graph and a citation graph. Based on these graphs, we propose the graph-aware selector, which is able to explicitly obtain the structure of the graph and utilizes the relationships among references. Extensive experiments demonstrate that our framework consistently improves performance across three base models (Llama3-8B, GPT-4o, and Claude-3-Haiku) in terms of LLM-based and graph-based metrics. Among selectors with different strategies, our graph-aware selectors perform the best.

Our contributions can be summarized as follows:

- We propose a multi-agent framework for full-text-based RWG task. Our framework creatively delegates iterative reading to two distinct agents: the *selector* and the *reader*. And the *writer* generates the final RWS.
- We design two kinds of graphs and propose a graph-aware selector within our framework, which acts under the constraints of the graph.
- We conduct in-depth experiments on the impact of different selecting strategies and input configurations. Our framework consistently improves the performance across different configurations.

2 Related Work

2.1 Related Work Generation

Existing approaches for RWG can be categorized into two types: extractive and abstractive methods. Extractive methods focus on selecting key sentences from cited papers and concatenating them to form the related work section (Hoang and Kan, 2010; Hu and Wan, 2014; Wang et al., 2018; Chen and Zhuge, 2019; Wang et al., 2019). Recent RWG models predominantly adopt abstractive methods (Chen et al., 2021, 2022; Liu et al., 2023). However, due to the limitations of input window sizes, these methods always rely on limited portions of reference papers, such as abstracts (AbuRa'ed et al., 2020; Ge et al., 2021; Li et al., 2022; Mandal et al., 2024), introductions and conclusions (Chen and Zhuge, 2019; Deng et al., 2021), related work section (Xing et al., 2020; Ge et al., 2021) or retrieved text spans (Li et al., 2023; Li and Ouyang, 2024). This lack of full-text information prevents models from fully capturing the content and relationships among references, leading to frequent misinterpretations and hallucinations (Xu et al., 2024). In addition, explaining the relationships among references is a critical aspect of RWG tasks. This is also a common struggle in previous models, where loose relationships among references and isolated explanations for each reference are frequent issues (Li and Ouyang, 2024). Although some works attempt to model inter-paper relationships using relation graph (Chen et al., 2021) or knowledge graph (Wang et al., 2022), they integrate graph structures implicitly and the challenge remains largely unresolved. To address the above challenges, we focus on the full-text-based RWG task and incorporate explicit graph structure constraints within a multi-agent framework.

2.2 Long-Sequence Modeling

Extensive approaches are proposed to address the input length limitations of language models, which can be categorized into four types: context window scaling, recurrence-based methods, retrieval-based methods, and agent-based methods. Context window scaling methods extrapolate the positional embeddings (Press et al., 2021; Chen et al., 2023b) or employ modified self-attention mechanisms (Beltagy et al., 2020; Guo et al., 2022). However, the attention mechanism may become less effective as sequence length increases (Liu et al., 2024). Recurrence-based methods use recursive mechanism to encode text, which are explored for different base models (Miller et al., 2016; Chevalier et al., 2023). However, each recurrence step can introduce information loss. Retrieval-based methods retrieve relevant portions based on the query (Izac-

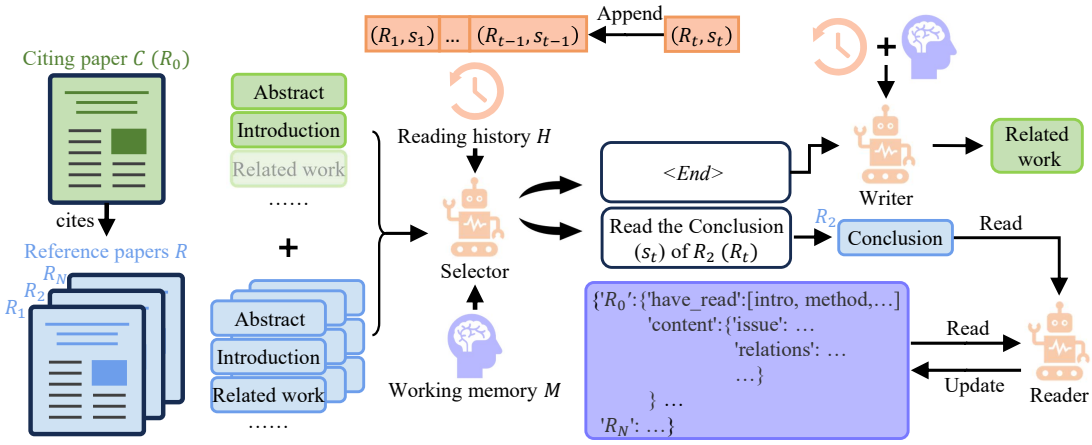

Figure 1: Overview of our multi-agent framework. The framework comprises a *selector*, a *reader*, and a *writer*, which collaboratively read the papers and generate the related work section.

ard and Grave, 2021; Wu et al.). However, such methods risk overlooking critical information. In agent-based frameworks, models operate as agents that dynamically read portions of the text and take flexible actions (Nakano et al., 2021; Yao et al., 2022; Chen et al., 2023a; Wang et al., 2024b). We adopt an agent-based framework, however, existing agent-based methods primarily focus on question answering (QA) tasks, where the agent only needs to locate an answer and a single agent suffices. In contrast, in RWG tasks, the reading order can impact the model's understanding of the papers and their relationships. Our multi-agent framework creatively delegates the reading process to two distinct agents, enabling to optimize both reading order and updating memory.

3 Problem Formulation

Before introducing our framework, we first define the problem formulation and notations used throughout this paper.

The input to the task consists of two main components: the citing paper C, which represents the paper being written, and a set of reference papers $\mathcal{R} = \{R_1, R_2, \ldots, R_N\}$, where R_i denotes a single reference paper and N is the total number of references. For simplicity, C is also denoted as R_0. Following prior RWG tasks, $\mathcal{R}$ is assumed to be given and corresponds to the references cited by the ground-truth related work section. Given the above inputs $\{R_0\} \cup \mathcal{R}$, the goal of the task is to generate a **related work section (RWS)** for C that incorporates all references in $\mathcal{R}$ while maintaining coherence with the context of C.

Since we focus on full-text-based RWG task, each reference paper R_i contains its entire content, represented as $R_i = \{s_{i,1}, s_{i,2}, \ldots, s_{i,L_i}\}$, where $s_{i,j}$ denotes the j-th section of R_i, and L_i is the total number of sections in R_i. Similarly, the input also includes all sections of the citing paper $R_0 = \{s_{0,1}, s_{0,2}, \ldots, s_{0,L_0}\}$, except for the related work section, which is to be generated.

4 Multi-Agent Framework

4.1 Overall Framework

As shown in Figure 1, our proposed framework is designed as a multi-agent system comprising three specialized agents: a *selector*, a *reader*, and a *writer*. These agents work collaboratively to iteratively process input content while maintaining a shared working memory M and a prior reading history H. The working memory M is designed in a well-organized JSON format, storing key information deemed essential for drafting the RWS. The prior reading history H records the sequence of previously read content in the form of tuples (paper ID, section name) in order to prevent cyclic reading of the input materials.

Selector. The selector is responsible for selecting the next section to read based on the abstracts of all papers $\{R_0\} \cup \mathcal{R}$, the current working memory M_{t-1}, and the reading history H_{t-1}. It outputs a tuple (R_t, s_t), representing the selected paper ID and section name to be read at step t:

$$(R_t, s_t) = \text{Selector}((s_{0,1}, \ldots, s_{N,1}), M_{t-1}, H_{t-1}), \quad (1)$$

Here, $s_{0,1}$ to $s_{N,1}$ denote the abstracts of all papers. Importantly, the selected section (R_t, s_t) must not already exist in H_{t-1}. When the selector deter-

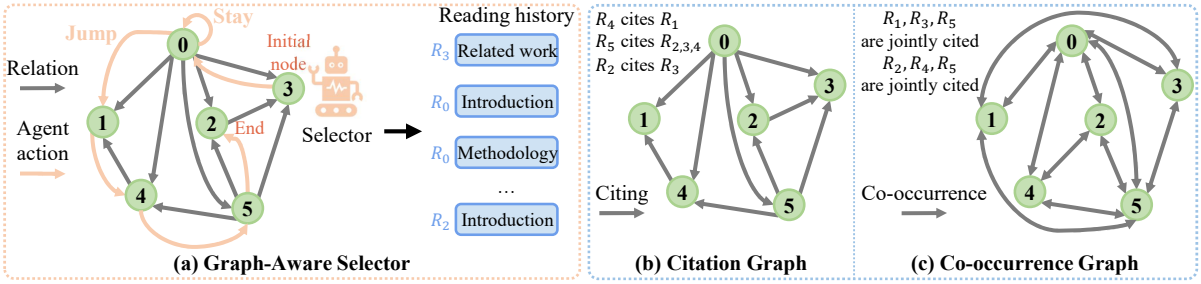

Figure 2: Illustration of our graph-aware selector. (a) Under the constraints of the graph structure, the selector selects either to continue reading the current paper or jump to an adjacent paper. We design two types of graphs: a (b) citation graph and a (c) co-occurrence graph.

mines that no further reading is necessary, it explicitly outputs a special termination symbol <*End*> to conclude the iterative process.

Reader. The reader processes the content of the selected section (R_t, s_t) and updates the working memory M_{t-1}:

$$M_t = \text{Reader}((R_t, s_t), M_{t-1}), \quad (2)$$

Given the task's requirement to handle numerous lengthy references, the working memory M can easily exceed the model's context size limitation. To address this, we enforce an explicit size constraint on M (e.g., 4096 tokens) and require the *reader* to reorganize its contents at each step, discarding irrelevant information to maintain a concise and task-relevant memory. After each iteration, the reading history H is updated as follows: $H_t = (H_{t-1}; (R_t, s_t))$. This iterative process continues until the selector signals termination.

Writer. Once the above iterative process concludes, the writer generates the final related work section based on the ultimate working memory M_T and reading history H_T:

$$\text{RWS} = \text{Writer}(M_T, H_T), \quad (3)$$

Here, T represents the total number of iterations.

To guide the writer in understanding what constitutes a high-quality RWS in academic writing, we prompt the writer with explicit instructions (e.g., avoid isolated descriptions of each reference; explain the relationships between papers; group similar studies together). In addition, we provide the writer with an example of a well-crafted related work section to leverage the in-context learning capabilities of LLMs.

4.2 Different Strategies for Selector

Since the reading order can impact the model's understanding of the papers and their relationships, our framework is designed to allow diverse strategies for selector. In this paper, we investigate the following five distinct selectors, each offering a unique strategy for determining the reading order of papers and sections.

Sequential Reading (SR). The selector determines the reading order by following the papers' IDs sequentially. It reads each section of a paper in order before moving on to the next. Formally, the selector generates a reading history H_T as:

$$H_T = \{(R_0, s_{0,1}), \ldots, (R_N, s_{N,L_N})\}, \quad (4)$$

Here, $s_{i,j}$ denotes the j-th section of paper R_i.

Random Reading (RR). Sequential reading may introduce biases due to the fixed order of reading, such as prioritizing earlier-read papers. To mitigate the potential bias, we implement a random reading strategy. The selector shuffles the sequential reading history into a random order:

$$H_T = \text{shuffle}(\{(R_0, s_{0,1}), \ldots, (R_N, s_{N,L_N})\}), \quad (5)$$

Vanilla LLM-Based Selector (Vanilla). We explore a vanilla LLM-based selector that dynamically determines the reading order. At each step t, the selector selects the next paper and section as:

$$(R_t, s_t) = \text{LLM}((s_{0,1}, \ldots, s_{N,1}), M_{t-1}, H_{t-1}), \quad (6)$$

This implementation takes advantage of the contextual reasoning abilities of LLMs to adaptively prioritize reading based on the task requirements.

Graph-Aware Selector. Understanding the relationships among references is crucial for RWG tasks. The graph structure is an intuitive way to describe relationships. Therefore, we propose a novel graph-aware selector, which constrains the reading order within the graph, enabling the selector

to capture the relationships among papers. Specifically, we propose building two types of graphs: a co-occurrence graph and a citation graph.

Co-occurrence Graph (Graph-Co). In practice, the RWS of reference papers $\mathcal{R}$ can provide valuable guidance for writing the RWS of the citing paper R_0. If the RWS of a reference discusses certain papers together, it is likely that these papers share a strong connection. To model this, we construct a co-occurrence graph (as shown in Figure 2(c)), where each node represents a reference paper, and an edge between two nodes indicates that the two papers are co-occurred in the same sentence of a prior paper's RWS. For convenience, we define the co-occurrence graph as a directed graph $G_{co} = (V_{co}, E_{co})$, vertices $V_{co} = \{R_0\} \cup \mathcal{R}$, edges $E_{co} = \{(R_i, R_j) \mid R_i \text{ and } R_j \text{ are jointly cited in } R_k\text{'s RWS}\}$. Importantly, the citing paper R_0 is assumed to be connected to all the reference papers in the graph to ensure its accessibility. The co-occurrence graph can effectively capture the implicit relationships among papers as exhibited in prior works.

Citation Graph (Graph-Ci). For all papers $\{R_0\} \cup \mathcal{R}$, there is a citation graph $G_{ci} = (V_{ci}, E_{ci})$, vertices $V_{ci} = \{R_0\} \cup \mathcal{R}$, edges $E_{ci} = \{(R_i, R_j) \mid R_i \text{ cites } R_j\}$. Citation relationships between papers can provide a more direct and explicit way to model inter-paper connections. Importantly, the citing paper R_0 is also included in the graph and cites all the reference papers in $\mathcal{R}$.

As shown in Figure 2(a), the graph-aware selector begins by selecting an initial paper R_{init} based on G. At each step $t-1$, the selector is positioned at a paper R_{t-1} and operates within its one-hop subgraph $G_{t-1} = (V_{t-1}, E_{t-1})$, defined as:

$$V_{t-1} = \{R_{t-1}\} \cup \{R_i \mid (R_i, R_{t-1}) \in G \text{ or } (R_{t-1}, R_i) \in G\}, \quad (7)$$

$$E_{t-1} = \{(R_i, R_j) \mid R_i, R_j \in V_{t-1}, (R_i, R_j) \in G\}, \quad (8)$$

Within this subgraph, the selector selects either to continue reading the current paper or jump to an adjacent paper:

$$(R_t, s_t) = \text{Selector}((s_{0,1}, \ldots, s_{N,1}), M_{t-1}, H_{t-1}, G), \quad R_t \in V_{t-1}, \quad (9)$$

We grant the selector access to the entire graph G as well as the abstracts of all papers, enabling it to make globally informed decisions.

5 Experiments

5.1 Experiment Setup

Dataset. We utilize OARelatedWork dataset (Docekal et al., 2024), currently the only dataset supporting full-text-based RWG. It has an average input length of 70k tokens in the test set. Unlike other datasets that are often domain-specific and focus on computer science (Lu et al., 2020; Chen et al., 2022), OARelatedWork is an open-domain dataset. Due to the substantial size of the dataset, all our experiments are conducted on 10% of the dataset.

Implementation Details. We experiment with three advanced LLMs in our multi-agent framework: one open-source model Llama3-8B and two closed-source models Claude-3-Haiku [1] and GPT-4o [2]. While the use of closed commercial LLMs is common in NLP research, it poses challenges for reproducibility. To address this concern, we ensure that all experiments conducted with Llama3-8B are performed on-site, providing strict reproducibility. As detailed in Section 4.2, our framework implements five distinct variants of the selector. These variants are distinguished using subscripts throughout the paper. The prompts and implementation details for each agent are provided in the Appendix C.

Baselines. We compare our framework against baselines of three categories: 1) **Abstract-based RWG Models.** Due to the input length limitations of language models, most previous works generate RWS solely based on the abstracts. We take several state-of-the-art models as our baselines, including the traditional language models PRIMERA (Xiao et al., 2022) and STK5SciSumm (To et al., 2024), as well as advanced LLMs (Llama3-8B, Claude-3-Haiku, and GPT-4o). 2) **Retrieval-based Full-Text RWG Models.** Many studies address the challenge of processing long inputs by leveraging retrieval-based methods (Izacard and Grave, 2021; Wu et al.). In RWG tasks, the Greedy Oracle (GO) (Nallapati et al., 2017) is a popular choice for selecting sentences. The selected sentences are then used as input to fit the context window of the model. We take the same models mentioned in abstract-based RWG category but extend their input to include sentences selected by the GO. 3) **LLMs with Extended Context Windows.** Certain advanced LLMs are equipped with long input windows, enabling them to process the full-text of all references

[1] Version claude-3-haiku-20240307

[2] Version gpt-4o-2024-08-06

Model	Graph-based Metrics			LLM-based Evaluation			
	Avg. Edges	Avg. Node Degree	Clustering Coefficient	Coverage	Logic	Relevance	Overall
Abstract-based RWG Models							
STK5SciSumm	0.0	0.0	0.0	1.02	1.04	2.18	1.41
PRIMERA	-	-	-	1.60	1.66	3.42	2.23
Llama3-8B	1.000	0.348	0.054	2.64	3.16	4.04	3.28
Claude-3-Haiku	1.729	0.448	0.084	2.84	3.40	4.10	3.45
GPT-4o	1.180	0.439	0.057	3.16	3.70	4.22	3.69
Retrieval-based Full-Text RWG Models							
STK5SciSumm + GO	0.0	0.0	0.0	1.08	1.14	2.48	1.57
PRIMERA + GO	-	-	-	1.78	1.72	3.42	2.31
Llama3-8B + GO	1.511	0.350	0.054	2.68	3.16	4.02	3.29
Claude-3-Haiku + GO	2.308	0.530	0.100	2.90	3.48	4.18	3.52
GPT-4o + GO	1.611	0.535	0.096	3.22	<u>3.76</u>	4.28	3.75
LLMs with Extended Context Windows							
Claude-3-Haiku	2.344	<u>0.869</u>	0.097	2.34	3.32	3.74	3.13
GPT-4o	1.244	0.624	0.136	3.18	3.66	4.20	3.68
Ours							
Llama3-8B $_{\text{Graph-Co}}$	1.162	0.644	0.135	2.74	3.20	3.98	3.31
Llama3-8B $_{\text{Graph-Ci}}$	1.410	0.651	0.154	2.80	3.34	4.18	3.44
Claude-3-Haiku $_{\text{Graph-Co}}$	<u>2.840</u>	0.832	<u>0.210</u>	2.98	3.48	4.22	3.56
Claude-3-Haiku $_{\text{Graph-Ci}}$	**3.240**	**0.942**	**0.231**	3.00	3.62	4.22	3.61
GPT-4o $_{\text{Graph-Co}}$	1.900	0.649	0.123	<u>3.28</u>	3.74	<u>4.34</u>	<u>3.79</u>
GPT-4o $_{\text{Graph-Ci}}$	2.125	0.667	0.128	**3.32**	**3.86**	**4.44**	**3.87**

Table 1: Performance of different models on the OARelatedWork dataset. The best and runner-up are in **bold** and <u>underlined</u>. Graph-based metrics for the PRIMERA model are not reported because its generated results do not distinguish between different references, making it infeasible to construct the corresponding graph.

simultaneously. We choose Claude-3-Haiku (200K-token) and GPT-4o (128K-token) as baselines.

5.2 Metrics

To avoid the poor correlation with human judgments in traditional automatic metrics (Chen et al., 2024), we choose two kinds of evaluation methods specifically for the RWG task.

Graph-based Metrics. To evaluate how well the generated RWS integrates and relates references, we adopt graph-based metrics (Martin-Boyle et al., 2024). A co-occurrence graph is constructed from the generated RWS, where each node represents a reference paper, and edges indicate that two references are jointly cited in a single sentence. A denser graph can reflect a more interrelated explanation of the references. We select three simple yet insightful graph statistics as metrics: **Average Number of Edges**, **Average Node Degree**, and **Clustering Coefficient**. Clustering coefficient can evaluate the tendency of references to form tightly connected clusters and avoid overestimating quality for connections between unrelated references.

LLM-based Evaluation. Previous LLM-based methods for evaluation predominantly focus on linguistic quality (Ge et al., 2021; Li et al., 2022). To provide more accurate evaluations tailored to RWG tasks, we carefully design three metrics (inspired by Wang et al.)—coverage, logic, and relevance—to assess the generated content's alignment with the essential characteristics of a high-quality RWS. **Coverage**: whether the generated RWS covers all key topics and provide detailed and thorough discussions about the references. **Logic**: whether the RWS is tightly structured and logically coherent, with content arranged in a clear and reasonable manner. **Relevance**: whether the RWS aligns with all papers and avoids hallucinations or factual inaccuracies. Each metric is scored on a 5-point scale. To enhance the accuracy and consistency, we employ chain-of-thought (CoT) prompting (Yu et al., 2023) with explicit scoring criteria. To mitigate potential biases introduced by the preferences of individual LLMs, we utilize three advanced LLMs: GPT-4o [3], Claude-3.5-haiku, and Gemini-1.5-Pro. The final results are the average of these three models. Details on the prompt design and scoring criteria for each metric are provided in the Appendix C.

5.3 Main Results

We compare our framework with three types of baselines and present the results in Table 1. We re-

[3] A distinct version, gpt-4o-2024-05-13

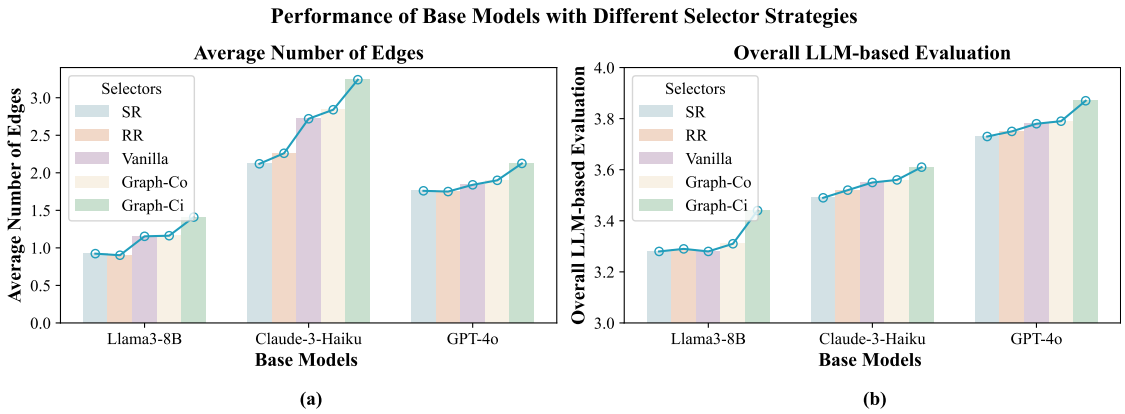

Figure 3: Performance comparison of five different selector strategies across three base models: (a) average number of edges in graph-based metrics, (b) overall LLM-based evaluation.

port the performance of our framework with graph-aware selectors, as they achieve the best performance. The key findings from the table are as follows: (1) **Full-text-based RWG models outperform abstract-based models.** The performance of full-text-based RWG models (including retrieval-based models and our framework) is significantly better than that of abstract-based RWG models. This trend holds consistently across five baselines and two kinds of evaluation metrics. It validates our motivation for full-text-based RWG tasks. (2) **Feeding all textual data in a single pass is not optimal.** While many advanced LLMs claim to handle long inputs, their methods for extending input windows often come at the cost of performance. As shown in Table 1, for GPT-4o and Claude-3-Haiku, feeding all the content does not perform as well as providing just the abstracts. Claude-3-Haiku's performance on LLM-based evaluation even drops by as much as 9.3% in overall. (3) **Consistent performance improvement with our framework.** Our framework shows consistent performance improvements across all three base models. Compared to retrieval-based models, our framework with Graph-Ci improves performance by 4.6%, 2.6%, and 3.2% on Llama3-8B, Claude-3-Haiku, and GPT-4o, respectively. However, the base model's capabilities still play a dominant role. The performance of Llama3-8B $_{\text{Graph-Ci}}$ is still lower than that of the abstract-based Claude-3-Haiku.

5.4 Different Strategies for Selector

We experiment with five different selector strategies across three base models. As shown in Figure 3, the performance trends of the five strategies are similar across the base models, following the order: SR < RR < Vanilla < Graph-Co < Graph-Ci. These results are expected: RR helps mitigate the potential bias in SR. Integrating LLMs into the decision-making process allows for a more intelligent selection of the reading order. Introducing the graph constraint enables the agent to more clearly capture the relationships among references. The inferior performance of Graph-Co compared to Graph-Ci may be attributed to the high connectivity of the co-occurrence graph, which imposes limited constraints on the agent's decision-making. In addition, there are minimal performance differences in Llama3-8B, which could be due to its relatively weaker capabilities, making it less sensitive to different selectors. The more detailed data can be found in Table 3 in Appendix A.

Input	Model	Avg. Edges (Graph)	Overall LLM -based
Intro. & Con.	Llama3-8B	1.063	2.93
	Llama3-8B $_{\text{Graph-Ci}}$	1.163	3.29
	Claude-3-Haiku	1.452	3.33
	Claude-3-Haiku $_{\text{Graph-Ci}}$	2.413	3.41
	GPT-4o	1.033	3.69
	GPT-4o $_{\text{Graph-Ci}}$	1.735	3.71
Related Work	Llama3-8B	1.088	3.22
	Llama3-8B $_{\text{Graph-Ci}}$	1.385	3.31
	Claude-3-Haiku	2.324	3.29
	Claude-3-Haiku $_{\text{Graph-Ci}}$	2.796	3.49
	GPT-4o	1.938	3.71
	GPT-4o $_{\text{Graph-Ci}}$	1.918	3.73

Table 2: Performance of different models under two common input configurations. Our proposed framework consistently improves the performance of all three base models across both settings.

5.5 Different Input Configurations

In addition to the abstracts, existing works also utilize introduction and conclusion (Chen and Zhuge,

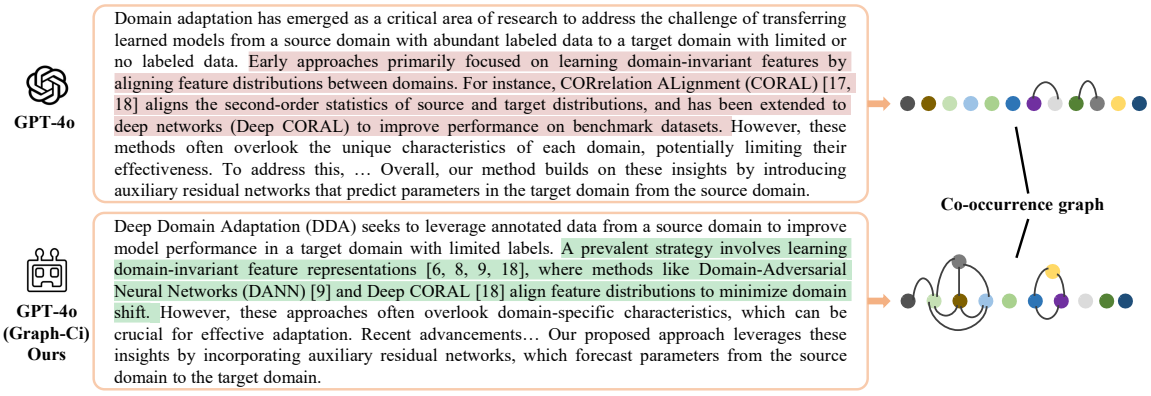

Figure 4: A case study comparing the RWS generated by GPT-4o and GPT-4o $_{\text{Graph-Ci}}$. On the right, a co-occurrence graph for graph-based metrics is constructed from the generated RWS. Our GPT-4o $_{\text{Graph-Ci}}$ model gives a more cohesive and interrelated explanation of the references, which is much easier for readers to follow. In contrast, GPT-4o fails to establish connections between references.

2019; Deng et al., 2021) or RWS (Xing et al., 2020; Ge et al., 2021) to represent references. We also apply our framework in these scenarios with limited portions of papers as text inputs and present the results in Table 2. There are some interesting findings: (1) When limiting the task input to only the Intro. & Con. or RWS, our framework still improves the performance of all base models. It underscores the robustness and adaptability of our framework when applied to text inputs of varying lengths. (2) A comparison between Table 1 and Table 2 reveals that for both Llama3-8B and Claude-3-Haiku, providing additional sections results in a performance decline. It could stem from the relatively weaker long-text processing capabilities of these models. In contrast, for GPT-4o, the inclusion of additional sections improves its performance. (3) Even with the integration of our framework, models based on partial sections still fall short in performance compared to models utilizing the full text. It emphasizes the necessity of full-text-based RWG task. (4) Models leveraging the RWS consistently outperform those based on the Intro. & Con. This aligns well with common academic writing practices, where the RWS of previous work is often a primary source for crafting the RWS of one's own paper. The detailed data can be found in Table 4 in Appendix A.

5.6 Case Study

Figure 4 presents a case study comparing the RWS generated by GPT-4o without and with our framework. The RWS generated by GPT-4o $_{\text{Graph-Ci}}$ is significantly more organized, with a clearer structure and stronger connections between references. The corresponding co-occurrence graph is also notably denser. It gives a more cohesive and interrelated explanation of the references, which is much easier for readers to follow. In contrast, GPT-4o fails to establish connections between references. The explanations of individual references are overly detailed and disjointed. As a result, it is less coherent and harder for readers to grasp, which is a common struggle in previous models. By constraining the reading process to the citation graph, our GPT-4o $_{\text{Graph-Ci}}$ model is better able to capture the relationships among references, resulting in a more logically structured and tightly connected output.

6 Conclusion

In this paper, we propose a multi-agent framework along with a graph-aware selector within the framework for full-text-based related work generation (RWG) tasks. The framework consists of three agents: a *selector*, a *reader*, and a *writer*, which work collaboratively to read the papers in selected order and finally generate the related work section (RWS). Our framework enables to optimize both the reading order and memory update. Our graph-aware selector can operate under the constraints of the graph to better capture the relationships among references. Extensive experiments demonstrate that our framework consistently improves the performance of different base models across various input configurations and the graph-aware selector based on the citation graph achieves the best performance. Case study reveals that our framework generates more logically coherent and tightly connected RWS.

Limitations

While our proposed framework improves graph-based metrics across different base models, indicating that the generated related work sections better capture the relationships among references, there is still a significant gap compared to human-written related work. Human-written related work can achieve an average number of edges of 9.48, whereas our best model, Claude-3-Haiku $_{\text{Graph-Ci}}$, only reaches 3.24. This gap is primarily due to the model's inability to effectively handle the level of detail in different references. For example, references that could be summarized in a single sentence may be overly elaborated by the model, leading to lower coherence and relevance in the generated related work. Addressing this issue is a key focus for our future work.

Our framework also requires that users provide a set of references in advance. Significant effort still needs to be spent on manually retrieving and selecting relevant papers. It limits the practical applicability of our method. We aim to develop a unified framework in the future where users can simply provide keywords or the citing paper, and the system will automatically retrieve the relevant papers from a vast corpus, pipelining the process of generating related work.

Ethical Statement

Given the exponential growth of academic publications, manually curating a comprehensive and relevant related work section has become increasingly challenging and time-consuming. The RWG task aims to enhance the efficiency of scientific work by reducing the time and effort required for authors to draft the related work section of their papers. However, the misuse of automatic RWG tools could raise ethical concerns, such as the potential for the generated related work to inadvertently plagiarize content or misrepresent the details of reference papers. Therefore, the related work generated by our model is intended to serve only as a preliminary draft, helping authors save time during the writing process. Authors are still required to carefully revise and verify the output to ensure academic integrity. We believe that using such models as assistive tools rather than a replacement for thorough reading and writing can enhance the exploration of vast scientific literature. The benefits of these tools are expected to outweigh the risks, provided they are used responsibly.

References

Ahmed AbuRa'ed, Horacio Saggion, Alexander Shvets, and Àlex Bravo. 2020. Automatic related work section generation: experiments in scientific document abstracting. *Scientometrics*, 125:3159–3185.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Howard Chen, Ramakanth Pasunuru, Jason Weston, and Asli Celikyilmaz. 2023a. Walking down the memory maze: Beyond context limit through interactive reading. *arXiv preprint arXiv:2310.05029*.

Jingqiang Chen and Hai Zhuge. 2019. Automatic generation of related work through summarizing citations. *Concurrency and Computation: Practice and Experience*, 31(3):e4261.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. 2023b. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*.

Xiuying Chen, Hind Alamro, Mingzhe Li, Shen Gao, Rui Yan, Xin Gao, and Xiangliang Zhang. 2022. Target-aware abstractive related work generation with contrastive learning. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*, pages 373–383.

Xiuying Chen, Hind Alamro, Mingzhe Li, Shen Gao, Xiangliang Zhang, Dongyan Zhao, and Rui Yan. 2021. Capturing relations between scientific papers: An abstractive model for related work section generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6068–6077.

Xiuying Chen, Tairan Wang, Qingqing Zhu, Taicheng Guo, Shen Gao, Zhiyong Lu, Xin Gao, and Xiangliang Zhang. 2024. Rethinking scientific summarization evaluation: Grounding explainable metrics on facet-aware benchmark. *arXiv preprint arXiv:2402.14359*.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. Adapting language models to compress contexts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3829–3846.

Zekun Deng, Zixin Zeng, Weiye Gu, Jiawen Ji, and Bolin Hua. 2021. Automatic related work section generation by sentence extraction and reordering. In *AII@ iConference*, pages 101–110.

Martin Docekal, Martin Fajcik, and Pavel Smrz. 2024. Oarelatedwork: A large-scale dataset of related work sections with full-texts from open access sources. *arXiv preprint arXiv:2405.01930*.

Yubin Ge, Ly Dinh, Xiaofeng Liu, Jinsong Su, Ziyao Lu, Ante Wang, and Jana Diesner. 2021. Baco: A background knowledge-and content-based framework for citing sentence generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1466–1478.

Mandy Guo, Joshua Ainslie, David C Uthus, Santiago Ontanon, Jianmo Ni, Yun-Hsuan Sung, and Yinfei Yang. 2022. Longt5: Efficient text-to-text transformer for long sequences. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 724–736.

Cong Duy Vu Hoang and Min-Yen Kan. 2010. Towards automated related work summarization. In *Coling 2010: Posters*, pages 427–435.

Yue Hu and Xiaojun Wan. 2014. Automatic generation of related work sections in scientific papers: an optimization approach. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1624–1633.

Gautier Izacard and Édouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 874–880.

Xiangci Li, Yi-Hui Lee, and Jessica Ouyang. 2023. Cited text spans for citation text generation. *arXiv preprint arXiv:2309.06365*.

Xiangci Li, Biswadip Mandal, and Jessica Ouyang. 2022. Corwa: A citation-oriented related work annotation dataset. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5426–5440.

Xiangci Li and Jessica Ouyang. 2024. Explaining relationships among research papers. *arXiv preprint arXiv:2402.13426*.

Jiachang Liu, Qi Zhang, Chongyang Shi, Usman Naseem, Shoujin Wang, Liang Hu, and Ivor Tsang. 2023. Causal intervention for abstractive related work generation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2148–2159.

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.

Yao Lu, Yue Dong, and Laurent Charlin. 2020. Multi-xscience: A large-scale dataset for extreme multi-document summarization of scientific articles. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8068–8074.

Biswadip Mandal, Xiangci Li, and Jessica Ouyang. 2024. Contextualizing generated citation texts. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 3849–3854.

Anna Martin-Boyle, Aahan Tyagi, Marti A Hearst, and Dongyeop Kang. 2024. Shallow synthesis of knowledge in gpt-generated texts: A case study in automatic related work composition. *arXiv preprint arXiv:2402.12255*.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1400–1409.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31.

Ofir Press, Noah A Smith, and Mike Lewis. 2021. Train short, test long: Attention with linear biases enables input length extrapolation. *arXiv preprint arXiv:2108.12409*.

Furkan Şahinuç, Ilia Kuznetsov, Yufang Hou, and Iryna Gurevych. 2024. Systematic task exploration with llms: A study in citation text generation. *arXiv preprint arXiv:2407.04046*.

Huy Quoc To, Hung-Nghiep Tran, Andr'e Greiner-Petter, Felix Beierle, and Akiko Aizawa. 2024. Skt5scisumm-a hybrid generative approach for multi-document scientific summarization. *arXiv preprint arXiv:2402.17311*.

Pancheng Wang, Shasha Li, Kunyuan Pang, Liangliang He, Dong Li, Jintao Tang, and Ting Wang. 2022. Multi-document scientific summarization from a knowledge graph-centric view. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 6222–6233.

Pancheng Wang, Shasha Li, Haifang Zhou, Jintao Tang, and Ting Wang. 2019. Toc-rwg: Explore the combination of topic model and citation information for automatic related work generation. *IEEE Access*, 8:13043–13055.

Yidong Wang, Qi Guo, Wenjin Yao, Hongbo Zhang, Xin Zhang, Zhen Wu, Meishan Zhang, Xinyu Dai, Min Zhang, Qingsong Wen, et al. 2024a. Autosurvey: Large language models can automatically write surveys. *arXiv preprint arXiv:2406.10252*.

Yongzhen Wang, Xiaozhong Liu, and Zheng Gao. 2018. Neural related work summarization with a joint context-driven attention mechanism. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1776–1786.

Yu Wang, Nedim Lipka, Ryan A Rossi, Alexa Siu, Ruiyi Zhang, and Tyler Derr. 2024b. Knowledge graph prompting for multi-document question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19206–19214.

Yuhuai Wu, Markus Norman Rabe, DeLesley Hutchins, and Christian Szegedy. Memorizing transformers. In *International Conference on Learning Representations*.

Wen Xiao, Iz Beltagy, Giuseppe Carenini, and Arman Cohan. 2022. Primera: Pyramid-based masked sentence pre-training for multi-document summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5245–5263.

Xinyu Xing, Xiaosheng Fan, and Xiaojun Wan. 2020. Automatic generation of citation texts in scholarly papers: A pilot study. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6181–6190.

Ziwei Xu, Sanjay Jain, and Mohan Kankanhalli. 2024. Hallucination is inevitable: An innate limitation of large language models. *arXiv preprint arXiv:2401.11817*.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757.

Zihan Yu, Liang He, Zhen Wu, Xinyu Dai, and Jiajun Chen. 2023. Towards better chain-of-thought prompting strategies: A survey. *arXiv preprint arXiv:2310.04959*.

A Detailed Results

Table 3 reports all the evaluation results for five different selector implementations across three base models (Llama3-8B, Claude-3-Haiku, and GPT-4o), representing the raw data for Figure 3. These additional results are consistent with the conclusions drawn in Section 5.4.

Table 4 reports all the evaluation results for the performance of different models under two common input configurations across three base models (Llama3-8B, Claude-3-Haiku, and GPT-4o). These additional results are consistent with the conclusions drawn in Section 5.5.

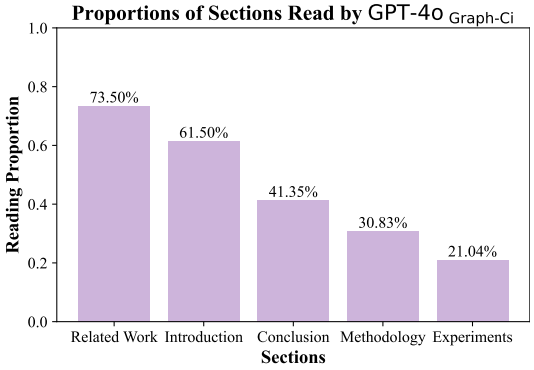

Figure 5: The proportion of sections that are selected for reading by GPT-4o Graph-Ci.

B Section Reading Statistics

We first report the average proportion of content read by our framework with different selectors (i.e., the number of sections read / total sections). As shown in Table 5, both SR and RR require reading all content. When LLMs are used for decision-making, the amount of content read is significantly reduced. The graph-aware selector further reduces the amount of required reading and Graph-Ci requires the least content to be read, at just 25.81%. It further emphasizes the advantage of Graph-Ci, which achieves the highest performance while reducing both time and computational costs.

To understand what content our framework selects for reading to achieve optimal performance, we conduct an analysis of the sections read by the best-performing model, GPT-4o Graph-Ci. We categorize the sections of the papers into five categories: Introduction, Related Work, Methodology, Experiments, and Conclusion. We then calculate the proportion of these sections that are selected for reading, as shown in Figure 5. We do not include the abstracts because we provide the abstracts of all papers for the model. The results reveal that the selector mainly selects the RWS, with a reading proportion of 73.5%, while the Experiments sections are the least read. This is consistent with our experience in writing the related work section. By focusing on these high-proportion sections, researchers could reduce the reading overhead while still obtaining sufficient information to write high-quality related work.

C Prompts for Agents and Evaluation

Figure 6 - Figure 11 present the detailed prompts for agents and evaluation.

Model	Graph-based Metrics			LLM-based Evaluation			
	Avg. Edges	Avg. Node Degree	Clustering Coefficient	Coverage	Logic	Relevance	Overall
Llama3-8B $_{SR}$	0.923	0.413	0.063	2.70	3.12	4.02	3.28
Llama3-8B $_{RR}$	0.902	0.433	0.094	2.70	3.20	3.96	3.29
Llama3-8B $_{Vanilla}$	1.154	0.455	0.077	2.76	3.10	3.98	3.28
Llama3-8B $_{Graph\text{-}Co}$	1.162	0.644	0.135	2.74	3.20	3.98	3.31
Llama3-8B $_{Graph\text{-}Ci}$	**1.410**	**0.651**	**0.154**	**2.80**	**3.34**	**4.18**	**3.44**
Claude-3-Haiku $_{SR}$	2.120	0.602	0.119	2.88	3.50	4.08	3.49
Claude-3-Haiku $_{RR}$	2.260	0.617	0.108	2.92	3.52	4.12	3.52
Claude-3-Haiku $_{Vanilla}$	2.720	0.668	0.117	2.94	3.50	4.20	3.55
Claude-3-Haiku $_{Graph\text{-}Co}$	2.840	0.832	0.210	2.98	3.48	**4.22**	3.56
Claude-3-Haiku $_{Graph\text{-}Ci}$	**3.240**	**0.942**	**0.231**	**3.00**	**3.62**	**4.22**	**3.61**
GPT-4o $_{SR}$	1.760	0.602	0.106	3.20	3.78	4.20	3.73
GPT-4o $_{RR}$	1.750	0.572	0.117	3.22	3.76	4.28	3.75
GPT-4o $_{Vanilla}$	1.840	0.563	0.108	3.22	3.84	4.28	3.78
GPT-4o $_{Graph\text{-}Co}$	1.900	0.649	0.123	3.28	3.74	4.34	3.79
GPT-4o $_{Graph\text{-}Ci}$	**2.125**	**0.667**	**0.128**	**3.32**	**3.86**	**4.44**	**3.87**

Table 3: Performance of five different selector implementations across three base models on the OARelatedWork dataset. The best for each base model are in **bold**.

Input	Model	Graph-based Metrics			LLM-based Evaluation			
		Avg. Edges	Avg. Node Degree	Clustering Coefficient	Coverage	Logic	Relevance	Overall
Intro. & Con.	Llama3-8B	1.063	0.505	0.094	2.38	2.60	3.80	2.93
	Llama3-8B $_{Graph\text{-}Ci}$	1.163	0.522	0.124	2.66	3.12	4.08	3.29
	Claude-3-Haiku	1.452	0.525	0.107	2.52	3.30	4.18	3.33
	Claude-3-Haiku $_{Graph\text{-}Ci}$	2.413	0.776	0.164	2.76	3.40	4.08	3.41
	GPT-4o	1.033	0.537	0.117	3.14	3.68	4.26	3.69
	GPT-4o $_{Graph\text{-}Ci}$	1.735	0.545	0.107	3.20	3.62	4.30	3.71
Related Work	Llama3-8B	1.088	0.442	0.125	2.52	3.16	3.98	3.22
	Llama3-8B $_{Graph\text{-}Ci}$	1.385	0.534	0.115	2.76	3.14	4.04	3.31
	Claude-3-Haiku	2.324	0.538	0.110	2.62	3.20	4.06	3.29
	Claude-3-Haiku $_{Graph\text{-}Ci}$	2.796	0.736	0.173	2.90	3.46	4.12	3.49
	GPT-4o	1.938	0.536	0.084	3.20	3.70	4.24	3.71
	GPT-4o $_{Graph\text{-}Ci}$	1.918	0.560	0.117	3.20	3.68	4.32	3.73

Table 4: Performance of different models under two common input configurations. Our proposed framework consistently improves the performance of all three base models across both settings.

Model	Average proportion of content read (%)
GPT-4o $_{SR}$	100.00
GPT-4o $_{RR}$	100.00
GPT-4o $_{Vanilla}$	35.27
GPT-4o $_{Graph\text{-}Co}$	28.53
GPT-4o $_{Graph\text{-}Ci}$	**25.81**

Table 5: The average proportion of content read by our framework (GPT-4o base) with different selectors.

Prompt for *Selector*-Vanilla

System Prompt: You are a research worker with excellent paper reading skills.

User Prompt:

I am writing a scientific paper. Now I need to cite some reference papers and write the Related Work section for the paper. Given the limitation on input length, your current task is to decide to read the content of each article (the paper currently being written and all cited papers) and remember the content needed for writing the Related Work section.

You need to explicitly maintain a memory of limited size (4096 tokens). Each time, I will provide you with the information you have read before and your memory after reading the previous content. Then, please select the content you want to read next.

I will list the abstracts and content structures of all papers in JSON format, for example, {'id': the id of the paper, 'abstract': the abstract of the paper, 'structure': the list of sections included in the paper}. The JSON information of the paper currently being written is as follows: {The Json information of the citing paper.} The JSON information of the cited papers is as follows: {The Json information of all the cited papers.}

Your previous memory is: {Working Memory M_{t-1}}

The content you have read before (in the JSON format) is: {Reading History H_{t-1}} This information is crucial because you cannot request to re-read content that has already been read.

Please select the content you want to read next based on the previous memory and the papers' information, and give the reason. Please answer in the JSON format {"id": the id of the article to be read, "section": the name of the section to be read in the article, "rationale": the reason}. You must ensure that the section name appears in the structure of the corresponding article. You also must ensure that the section has not been read before. If you think that there is no need to read any more content, please only respond with *'End'*. Respond *'End'* at any appropriate time, as many sections of the paper are irrelevant to writing the Related Work section. You need to minimize the consumption brought about by reading. Be strictly follow the format, and do not respond with any other additional content!

Figure 6: Prompt for *Selector*-Vanilla.

Prompt for *Selector*-Graph-Co

System Prompt: You are a research worker with excellent paper reading skills.

User Prompt:

I am writing a scientific paper. Now I need to cite some reference papers and write the Related Work section for the paper. Given the limitation on input length, your current task is to decide to read the content of each article (the paper currently being written and all cited papers) and remember the content needed for writing the Related Work section.

We will provide a co-occurrence graph of the papers where each node represents a reference paper, and edges indicate that two references are jointly cited in a single sentence in previous Related Work. Your role is to act as an intelligent agent traversing this graph. At each step, you will know your current position in the graph (the paper you are currently reading), along with its adjacent papers. Your task is to decide what to read next within this local graph structure. You can choose to continue reading the current paper or move to a connected paper (essentially making a jump in the graph).

You also need to explicitly maintain a memory of limited size (4096 tokens). Each time, I will provide you with the information you have read before and your memory after reading the previous content. Then, please decide the content you want to read next.

I will first provide global co-occurrence graph information. The graph is presented in the form of a dictionary with the format: {"paper_id": [list of its co-occurrence papers IDs]}. The specific co-occurrence graph is as follows: {Co-occurrence Graph G}

I will list the abstracts and content structures of papers in JSON format, for example, {'id': the id of the paper, 'abstract': the abstract of the paper, 'structure': the list of sections included in the paper}. An ID of -1 represents the paper currently being written. The JSON information of the paper currently being read is as follows: {The Json information of the paper currently being read.} The JSON information of the adjacent papers is as follows: {The JSON information of the adjacent papers} We additionally provide information about other papers that are not part of the local subgraph, which may assist in your selection. However, you are not allowed to request the content of these papers, as your task is to choose what to read within the local subgraph. The JSON information of the other papers is as follows: {The Json information of the other papers.}

Your previous memory is: {Working Memory M_{t-1}}

The content you have read before (in the JSON format) is: {Reading History H_{t-1}} This information is crucial because you cannot request to re-read content that has already been read.

Please select the content you want to read next based on the previous memory and the papers' information, and give the reason. Please answer in the JSON format {"id": the id of the article to be read, "section": the name of the section to be read in the article, "rationale": the reason}. You must ensure that the section name appears in the structure of the corresponding article. You also must ensure that the section has not been read before. If you think that there is no need to read any more content, please only respond with *'End'*. Respond *'End'* at any appropriate time, as many sections of the paper are irrelevant to writing the Related Work section. You need to minimize the consumption brought about by reading. Be strictly follow the format, and do not respond with any other additional content!

Figure 7: Prompt for *Selector*-Graph-Co.

Prompt for *Selector*-Graph-Ci

System Prompt: You are a research worker with excellent paper reading skills.

User Prompt:

I am writing a scientific paper. Now I need to cite some reference papers and write the Related Work section for the paper. Given the limitation on input length, your current task is to decide to read the content of each article (the paper currently being written and all cited papers) and remember the content needed for writing the Related Work section.

We will provide a citation graph of the papers, and your role is to act as an intelligent agent traversing this graph. At each step, you will know your current position in the graph (the paper you are currently reading), along with the papers cited by this paper and those that cite it. Your task is to decide what to read next within this local graph structure. You can choose to continue reading the current paper or move to a connected paper (essentially making a "jump" in the graph).

You also need to explicitly maintain a memory of limited size (4096 tokens). Each time, I will provide you with the information you have read before and your memory after reading the previous content. Then, please decide the content you want to read next.

I will first provide global citation graph information. The citation graph is presented in the form of a dictionary with the format: {"paper_id": [list of its referenced papers IDs]}. The specific citation graph is as follows: {Citation Graph G}

I will list the abstracts and content structures of papers in JSON format, for example, {'id': the id of the paper, 'abstract': the abstract of the paper, 'structure': the list of sections included in the paper}. An ID of -1 represents the paper currently being written. The JSON information of the paper currently being read is as follows: {The Json information of the paper currently being read.} The JSON information of the papers cited by the paper currently being read is as follows: {The Json information of the papers cited by the paper currently being read.} The JSON information of the papers citing the paper currently being read is as follows: {The Json information of the papers citing the paper currently being read.} We additionally provide information about other papers that are not part of the local subgraph, which may assist in your selection. However, you are not allowed to request the content of these papers, as your task is to choose what to read within the local subgraph. The JSON information of the other papers is as follows: {The Json information of the other papers.}

Your previous memory is: {Working Memory M_{t-1}}

The content you have read before (in the JSON format) is: {Reading History H_{t-1}} This information is crucial because you cannot request to re-read content that has already been read.

Please select the content you want to read next based on the previous memory and the papers' information, and give the reason. Please answer in the JSON format {"id": the id of the article to be read, "section": the name of the section to be read in the article, "rationale": the reason}. You must ensure that the section name appears in the structure of the corresponding article. You also must ensure that the section has not been read before. If you think that there is no need to read any more content, please only respond with *'End'*. Respond *'End'* at any appropriate time, as many sections of the paper are irrelevant to writing the Related Work section. You need to minimize the consumption brought about by reading. Be strictly follow the format, and do not respond with any other additional content!

Figure 8: Prompt for *Selector*-Graph-Ci.

Prompt for *Reader*

System Prompt: You are a research worker with excellent paper reading skills.

User Prompt:

I am writing a scientific paper. Now I need to cite some reference papers and write the Related Work section for the paper. Given the limitation on input length, your current task is to read the content of each article (the paper currently being written and all cited papers) and remember the content needed for writing the Related Work section.

You need to explicitly maintain a memory of limited size (4096 tokens). Each time, I will provide you with the information you have read before and your memory after reading the previous content. At the same time, I will provide you with the content you were asked to read last time. You need to read this content and update what you have in your memory.

I will list the abstracts of all papers in JSON format, for example, {'id': the id of the paper, 'abstract': the abstract of the paper}. The JSON information of the paper currently being written is as follows: {The Json information of the citing paper.} The JSON information of the cited papers is as follows: {The Json information of all the cited papers.}

The content you requested to read last time is the {Section s_t} of paper {R_t}: {The content of s_t.}

The content you have read before (in the JSON format) is: {Reading History H_{t-1}} This information is crucial because you cannot request to re-read content that has already been read.

Your previous memory is: {Working Memory M_{t-1}}

You need to differentiate between different articles by the paper id in the memory. When writing the Related Work section, understanding the relationships between different papers is very important, so you can try to keep track of this in your memory. Please answer your updated memory based on the provided content and the previous memory. Feel free to modify the content in the memory, adding information that you believe will be useful for writing the Related Work section and removing any irrelevant or redundant information in the memory. Due to the memory size limitations, you should aim to record as much useful information as possible. Please also give the reason. Please answer in the JSON format {"memory": the updated memory, "rationale": the reason}. Be strictly follow the format, and do not respond with any other additional content!

Figure 9: Prompt for *Reader*.

Prompt for *Writer*

System Prompt: You are a research worker with excellent paper writing skills.

User Prompt:

I am writing a paper and have already written all the sections except for the Related Work section. Now I need to cite some reference papers. Please write a Related Work section for me to conform to the format of scientific conferences. Please note that explain the connections between the papers rather than summarize each article separately. Also, please summarize all the papers at the beginning and elaborate on the relationship between papers.

The abstract that has already been written is as follows: {The abstract of the citing paper.}

I will list the abstracts of all cited papers in JSON format like {'id': id of the cited paper, 'abstract': abstract of the cited paper}. Note that the order of the provided papers is random, so you need to reorganize the order based on the relationship between the papers. The Json information of cited papers is as follows: {The Json information of all the cited papers.}

In order to get more detailed information about the papers, one of your peers has read the full content of all the articles and has maintained a memory they believe are important for writing the Related Work section. You might find this memory helpful and receive assistance from it. The content of the memory is as follows: {Working Memory M_T}

In the Related Work section, it is crucial to maintain a high level of synthesis and cohesion. Therefore, you should group similar studies together, highlighting their shared themes, and clearly explain the relationships between the referenced works. Incorporate multiple references within a single sentence, rather than introducing each reference in isolation! Here is an example of a Related Work section with a high level of synthesis and cohesion: {An example of the Related Work section.}

Please note that cite the corresponding papers by their ids and return the Related Work section in the format {'related_work': the content of the Related Work}. Be strict to the format and do not answer any other extra content!

Figure 10: Prompt for *Writer*.

Prompt for Evaluation

System Prompt: You are a strict paper reviewer.

User Prompt:

Coverage

Please evaluate the 'Coverage' of the Related Work section of papers based on the abstracts of all the cited papers.

The scoring criteria are as follows:

- Score 1: The 'Related Work' has limited coverage, only touching on a small portion of the topic and lacking discussion on key areas.
- Score 2: The 'Related Work' covers some parts of the topic but has noticeable omissions, with significant areas either underrepresented or missing.
- Score 3: The 'Related Work' is generally comprehensive in coverage but still misses a few key points that are not fully discussed.
- Score 4: The 'Related Work' covers most key areas of the topic comprehensively, with only very minor topics left out.
- Score 5: The 'Related Work' comprehensively covers all key and peripheral topics, providing detailed discussions and information.

Logic

Please evaluate the 'Logic' of the Related Work section of papers based on the abstracts of all the cited papers.

The scoring criteria are as follows:

- Score 1: The 'Related Work' lacks logic, with no clear connections between sentences, making it difficult to understand the content.
- Score 2: The 'Related Work' has weak logical flow with some content arranged in a disordered or unreasonable manner.
- Score 3: The 'Related Work' has a generally reasonable logical structure, with most content arranged orderly, though some links and transitions could be improved such as repeated explanation.
- Score 4: The 'Related Work' has good logical consistency, with content well arranged and natural transitions, only slightly rigid in a few parts.
- Score 5: The 'Related Work' is tightly structured and logically clear, with all content arranged most reasonably, and transitions between adjacent sentences smooth without redundancy.

Relevance

Please evaluate the 'Relevance' of the Related Work section of papers based on the abstracts of all the cited papers.

The scoring criteria are as follows:

- Score 1: The content is outdated or unrelated to the field it purports to review, offering no alignment with the topic.
- Score 2: The 'Related Work' is somewhat on topic but with several digressions; the core subject is evident but not consistently adhered to.
- Score 3: The 'Related Work' is generally on topic, despite a few unrelated details.
- Score 4: The 'Related Work' is mostly on topic and focused; the narrative has a consistent relevance to the core subject with infrequent digressions.
- Score 5: The 'Related Work' is exceptionally focused and entirely on topic; the article is tightly centered on the subject, with every piece of information contributing to a comprehensive understanding of the topic.

I will list the abstracts of all cited papers in JSON format like {'id': id of the cited paper, 'abstract': abstract of the cited paper}: {The Json information of all the cited papers.} The content of the 'abstract' section of the current paper is: {The abstract of the citing paper.}

And the content of the 'Related Work' section to be evaluated is: {Related Work section to be evaluated}

Provide your reasoning for the score first, and then give the final score. Use JSON format for your response, like: {"reason": "The reason for your score", "score": "The final given score"}. Be strict to the format and do not answer any other extra content!

Figure 11: Prompt for Evaluation.