ARTICLE
https://doi.org/10.26599/AIR.2026.9150003

CAAI Artificial Intelligence Research

FG-RAG: A Fine-Grained Semantic Optimization Framework for Retrieval-Augmented Generation

Mingan Xu[1,2], Tingjie Liu[1,2], Youhua Xia[1] (✉)

ABSTRACT

Large Language Models (LLMs) have demonstrated remarkable capabilities, yet frequently encounter issues with hallucinations. Retrieval-augmented generation (RAG) mitigates this problem by integrating external knowledge. However, current RAG approaches face significant limitations, such as redundant tokens that dilute semantic focus of LLMs and suboptimal prompt ordering. To address these challenges, this paper introduce a fine-grained framework called FG-RAG, with two key components: (1) Refined Prompt Generation. Standalone propositions are firstly extracted from raw documents and dynamically organized into a semantic graph to capture their interrelationships. After retrieving relevant subgraphs from this graph, a Directional Diffusion Model (DDM) is adapted to iteratively refine these graph representations, which are subsequently transformed into soft prompts compatible with LLMs; **(2) Prompt Ordering.** With these soft prompts, this work formulates prompt ordering as a Markov Decision Process (MDP) and optimize it through Reinforcement Learning (RL). By leveraging reward derived from prompts performance, the RL agent learns to order prompts to maximize the accuracy of the LLMs' output, ensuring optimal utilization of the generated prompts. Extensive evaluations of question-answering benchmarks demonstrate that FG-RAG outperforms state-of-the-art RAG methods, with 1.3% EM and 1.5% F1 score improvements.

KEYWORDS

Graph data, Artificial intelligence, Information process, Retrieval augmented generation (RAG), Reinforcement learning

Just Accepted

1 Introduction

Large Language Models (LLMs) have demonstrated exceptional performance in a wide spectrum of knowledge-intensive Natural Language Processing (NLP) tasks [1][2]. However, these models are constrained by inherent limitations in their training data and the inability to incorporate real-time knowledge updates, which often lead to the generation of inaccurate or fabricated content. Retrieval-Augmented Generation (RAG) [3] has emerged as a promising solution to address these challenges. RAG significantly enhances the model's factual grounding and contextual accuracy by integrating relevant external knowledge through retrieval mechanisms [4]. This innovative approach not only maintains the robust generative capabilities of LLMs but also effectively mitigates the issue of hallucinations, thus improving the reliability and quality of the generated content.

While RAG improves LLMs by grounding them in external knowledge[5][6], existing methods suffer from two critical bottlenecks: (1) Coarse-Grained and Fragmented Retrieval. Many RAG frameworks [7][8] default to document-level retrieval for simplicity, this can introduce redundancy when only small segments are relevant. For example, injecting full documents may force the LLM to process irrelevant text, potentially obscuring key information and increasing hallucination risks [9]. Furthermore, by processing documents in isolation, these frameworks fail to capture inter-document relationships, leading to fragmented and incomplete outputs, especially for complex queries requiring multi-hop reasoning; (2) Rigid Prompt Ordering. Many RAG frameworks [10] prioritize prompts based on human-defined relevance rankings, such as placing the most semantically similar passages at the forefront. However, due to the unique characteristics of LLMs, including their self-attention mechanisms, their input preferences often diverge from human intuition, a phenomenon termed the "preference gap" [11], leading to suboptimal knowledge utilization.

Existing works attempt to solve these challenges through two primary avenues: (1) Enhancing Retrieval Quality through Text Chunking or Knowledge Graph Integration. For retrieval granularity, text chunking (e.g., sliding window approaches [12]) reduces document-level redundancy but struggles to extract self-contained semantic units, which can mislead LLMs by introducing ambiguous or contradictory information. The enhancement of knowledge graphs [13][14] injects structured relationships into LLM contexts. However, static graphs fail to adapt to query-specific contexts, and their rigid schemas limit coverage of dynamic or domain-specific knowledge; (2)

1 Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ), Shenzhen 518107, China
2 College of Computer Science and Software Engineering, Shenzhen University, Sehnzhen 518060, China
Email: xiayouhua@gml.ac.cn

Leveraging LLMs for Prompt Ranking. The emergence of LLMs has empowered researchers to develop various generative ranking models. For instance, RankGPT [15] and LRL [16] fine-tune LLMs to schedule prompts dynamically. While representing an advance over static retrieval approaches, this method is computationally expensive and inherently limited by the LLM's flaws, such as hallucination tendencies and reasoning errors when processing complex prompt relationships. These limitations underscore the necessity for a framework that ensures both high-quality prompts for LLMs and a reliable ranking method capable of fully utilizing these prompts.

To address the abovementioned difficulties, this work proposes FG-RAG, a fine-grained framework that synergistically integrates optimized prompt generation with dynamic prompt orchestration. This approach comprises two core components: (1) Refined Prompt Generation. To tackle the inherent complexity of multi-theme documents, proposition transfer is first employed to decompose raw text into standalone statements. These propositions are then organized into a semantic graph, where nodes represent individual propositions and edges encode contextual relationships. Query-relevant subgraphs are retrieved from this graph and converted into continuous embeddings through a Directional Diffusion Model (DDM) [17]. Through dynamic graph construction and the denoising mechanism of DDM, high-quality retrieved information is ensured. Unlike traditional Graph Neural Networks (GNNs) that propagate features through fixed aggregation rules, DDM introduces a diffusion process that iteratively refines node features, resulting in smoother and more accurate graph representations. These representations are then transformed into soft prompts compatible with LLMs; (2) Prompt Ordering. To optimize the sequential presentation of prompts to the LLM, the ordering task is formulated as a Markov Decision Process (MDP), and a Reinforcement Learning (RL) solution is developed. The RL agent iteratively optimizes prompt arrangement by refining its ranking policy based on reward derived from prompt performance. This adaptive process dynamically aligns the prompt sequence with the LLM's response patterns, ensuring optimal utilization of soft prompts and enhancing output accuracy. RL provides an effective solution that maintains high performance without demanding excessive computational resources. The synergistic operation of these components ensures high-quality prompts for the LLM and guarantees their optimal sequencing to maximize model performance.

The main contributions of this paper are as follows:

- Adaptation of DDM for high-quality prompt generation. Through dynamic proposition graph construction and DDM's diffusion-based denoising mechanism, this process generates optimized graph representations that effectively preserve semantic relationships while significantly reducing redundancy. These refined graph representations are then converted into LLM-compatible soft prompts through an adaptive projection layer, significantly improving prompt quality.

- The first framework that jointly applies diffusion models and reinforcement learning to proposition-level graph representations for RAG. Building upon the DDM-based soft prompts, prompt ordering is optimized via reinforcement learning by formulating it as an MDP. The RL agent learns an optimal prompt sequence, dynamically aligning it with model reasoning patterns to bridge the retrieval-generation preference gap, thereby maximizing the utility of prompts for generation tasks.

- Empirical results on both multi-hop and single-hop reasoning tasks demonstrate that the FG-RAG framework outperforms existing frameworks, proving the effectiveness of the proposed approach. By jointly optimizing these complementary components, the proposed approach achieves an average performance improvement of 1.3% in the EM score and 1.5% in the F1 score.

2 Related Work

Parameter-Efficient Fine-Tuning (PEFT). The field of LLMs has witnessed significant advancements through the development of parameter-efficient fine-tuning techniques [18][19]. These methods have played a pivotal role in enhancing LLM performance while minimizing the computational burden of full parameter training. For example, LoRA [20] significantly reduces computational costs by decomposing weight updates into lower-rank matrices, thereby maintaining the original model's performance with fewer trainable parameters. This work builds upon the soft-prompt approach but is also extendable to other parameter-efficient fine-tuning methods. Most relevant to this work is ID-PT [21], where the input is fed into a separate trainable neural network to generate the soft prompt. This work extends this idea by leveraging a graph diffusion model to encode structured data, enabling the LLM generation of contextually rich soft prompts.

Retrieval on Graphs. Recent studies [13][14] have explored integrating external knowledge graphs [22][23] into the RAG framework to enhance reading comprehension. Unlike these methods that rely on static graphs with coarse entity or chunk nodes, FG-RAG employs dynamic proposition-level graph construction. This approach builds query-adaptive structures from atomic semantic units, ensuring finer granularity and better alignment with specific information needs. Moreover, LLMs, primarily trained on textual data, have difficulty directly interpreting graph-structured information. Although soft prompts have been proposed as a bridge to help LLMs process non-textual data, generating effective graph-based soft prompts remains a significant challenge. Recent efforts [24]) have employed GNNs to encode graph structures into LLM-compatible representations. However, traditional GNNs rely on fixed aggregation rules, resulting in fragmented and discontinuous representations that struggle to capture the full spectrum of continuous semantic relationships, making them unsuitable for tasks that demand nuanced semantic modeling. To address this, DDM [17] is adapted. Comparison with existing graph diffusion models, DDM differs from prior graph diffusion models (e.g., GDSS [25], GRAND [26] in two key aspects. First, it employs a direction-aware diffusion process that preferentially propagates information along semantically similar proposition paths, whereas standard models diffuse noise uniformly across all graph edges. Second, its denoising objective is optimized to produce semantically continuous embeddings that bridge isolated clusters in the feature space — a property essential for generating coherent soft prompts for LLMs.

Just Accepted

RL for information retrieval. RL has been extensively applied in information retrieval through two primary paradigms: document ranking and query reformulation, with emerging extensions to prompt optimization in the LLM era. For document ranking, RL framework like DRN [27] formulates list construction as an MDP to optimize ranking metrics such as NDCG [28]. For query reformulation, method like C-3PO [29] learns to refine queries based on retrieval feedback to improve document relevance.

Recent work extends RL to prompt optimization, treating prompts as learnable parameters optimized through LLM interaction. RLPrompt [30] generates discrete prompt tokens via policy networks. TEMPERA [31] enables test-time prompt editing by learning editing policies conditioned on query characteristics. While these approaches optimize the content of a single prompt, FG-RAG addresses a distinct challenge: optimizing the order of multiple graph-structured prompts in RAG. The MDP selects which prompt to append next from a candidate set, rather than generating or editing token sequences.

3 Problem Formulation

Retriever. For a given input q, the objective of the retriever is to obtain a sorted list of entries from the corpus $D = \{d_i\}_{i=1}^{N}$ that are pertinent to q. This study considers the common scenario where a pre-trained dense retriever is utilized. This retriever typically employs a dual encoder system to process the input q and each document d. Specifically, the encoder maps each document to an embedding $E(d)$. The relevance between the input and a document is determined by calculating the cosine similarity between their respective embeddings. The top-k documents, which exhibit the most significant similarity to the input q, are selected based on the following expression:

$$D_{\text{retr}}^{k} = \text{argtop-}k\big[E(d_i)^{\top} \cdot E(q) \mid i = \{1 \dots N\}\big]. \quad (1)$$

Propositions Graph. For the retrieved top-k documents, they are decomposed into propositions(i.e., self-contained sentences with comprehensive contextual information) and construct a graph using these propositions as nodes. The edges in the graph capture the relationships between the propositions. Formally, the proposition graph can be defined as $G = (V, E, \{T_n\}_{n \in V})$, where V and E denote the node set and edge set, respectively. T_n represents the natural language attributes of the corresponding node $n \in V$.

Subgraphs. Subgraphs are subgraph structures within a proposition graph, e.g., G with finite node set V and edge set E, and its subgraph set:

$$\mathcal{S}(G) = \{g = (V', E', \{T_n\}_{n \in V'}) \mid V' \in \mathcal{P}(V), E' \in \mathcal{P}(E)\}, \quad (2)$$

where $\mathcal{P}(V)$ and $\mathcal{P}(E)$ represent the power set of V and E, respectively.

Soft Prompt. For a proposition graph, the top-M subgraphs most relevant to the input query q are indexed. Each subgraph is then transformed into a graph representation using an MLP. The encoded graph representation is thereby treated as as a soft prompt p for LLMs.

Ordering. This work models the process of selecting l representations from a set of m graph representations and arranging them in a specific order as an MDP. The combined prompt, denoted as P, can be described as follows: Let $\mathcal{P}$ be the set of m soft prompts. The selection of l prompts in a specific order is represented by the ordered sequence:

$$P = (p_1, p_2, \dots, p_l), \quad (3)$$

where each element p_t (for $t = 1, 2, \dots, l$) represents the selected prompt at time step t, with t sequentially increasing from 1 to l.

Large Language Models. The LLM, parameterized by weights θ, takes the prompt P and query tokens Q as inputs and generates a sequence of tokens $Y = \{y_1, y_2, \dots, y_r\}$ as output. Formally, the probability distribution of the final output sequence is given by:

$$p_\theta(Y|[P;Q]) = \prod_{i=1}^{r} p_\theta(y_i|y_{<i}, [P;Q]), \quad (4)$$

where notation $[;]$ indicates the concatenation operator, $y_{<i}$ denotes the prefix tokens of the sequence Y up to position $i-1$, $p_\theta(y_i|y_{<i}, [P;Q])$ represents the conditional probability of generating the token y_i given the prefix tokens $y_{<i}$ and the concatenated input $[P;Q]$.

4 FG-RAG

This section details the FG-RAG framework. As illustrated in Figure 1, a graph is first constructed with nodes representing propositions extracted from retrieved documents. After retrieving relevant sugraphs from this graph, their graph representations are refined using a DDM, which are then converted into soft prompts for the downstream component. The agent's policy network learns to optimize the sequence of these prompts during interactions with the LLM environment, leveraging a reward function to dynamically adjust the prompt ordering strategy for maximal task performance. Below, this paper will introduce the framework focusing on three core aspects: graph construction, graph-based information processing, and reinforcement learning-based prompt ordering.

4.1 Graph construction

Typically, a document contains multiple topics and diverse content. To address these limitations, this work leverages proposition transfer [32], a technique that extracts standalone, self-contained statements from raw text using rule-based or heuristic methods, such as sentence segmentation and dependency parsing. These extracted propositions are then organized into a proposition graph, where nodes represent individual propositions and edges capture their semantic relationships measured by embedding cosine similarity. To efficiently search for the subgraph from the entire graph, inspired by GRAG [33], this work uses a divide-and-conquer strategy that balances the number of candidate subgraphs with computational intensity. Specifically, each node is encoded with its surrounding neighborhoods as ego-graphs. During subgraph retrieval, a pool of promising candidates is indexed (subgraph indexing), and the top-ranked ones are further ranked and retained (subgraph ranking).

Just Accepted

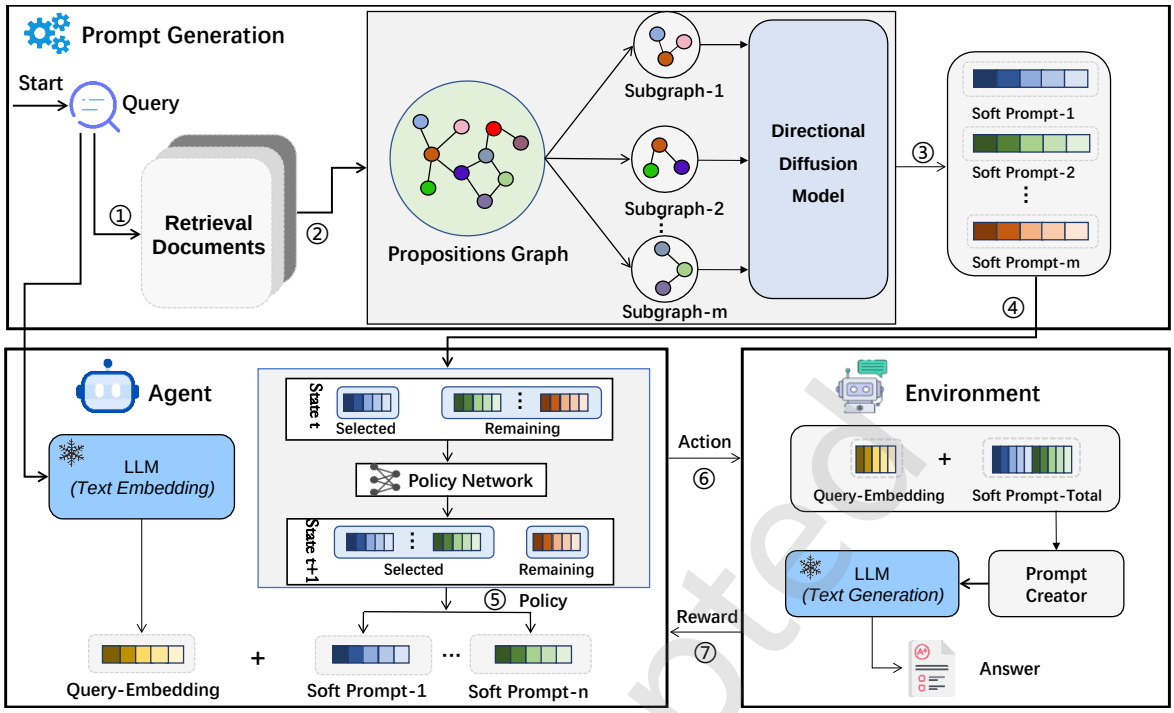

Fig. 1 The proposed FG-RAG contains two key components: (1) Refined Prompt Generation(The Upper Part), which extracts standalone propositions from raw documents and organizes them into propositions graphs. From this graph, relevant subgraphs are retrieved and transformed into soft prompts; (2) Prompt Ordering(The Lower Part). This part optimizes prompt ordering via an MDP trained with RL, where the final action combines the query with selected prompts as LLM input, and the LLM's response quality serves as the reward signal for policy optimization.

Subgraph indexing. This module indexes w-hop ego-graphs in G and convert them to graph representations. These representations are designed to capture semantic meaning, as they will be compared with the question embedding during retrieval. To achieve this, this work employs a Pre-trained Language Model (PLM) to transform the text attributes of nodes into embeddings. Subsequently, a mean pooling is applied operation to aggregate these node embeddings into a single graph embedding, denoted as $x_{sub} \in \mathbb{R}^{d_{PLM}}$ for each subgraph $sub \in \mathcal{S}(G)$, where d is the dimensionality of the embedding:

$$x_{sub} = \text{POOL}(\text{PLM}(\{T_n\}_{n \in V_{sub}})). \tag{5}$$

Subgraph Ranking. In the retrieval phase, the same PLM is utilized as an encoder to generate the question embedding:

$$x_q = \text{PLM}(q) \in \mathbb{R}^{d_{PLM}}, \tag{6}$$

Here, x_q shares the same dimensionality d with the graph embeddings. The cosine similarity between the question embedding and each w-hop ego-graph embedding then determines the top-m most relevant subgraphs:

$$\mathcal{S}(G)_m = argtop[m]_{sub \in \mathcal{S}(G)} \cos(x_q, x_{sub}), \tag{7}$$

where $\cos(\cdot, \cdot)$ denotes the cosine similarity function. The resulting set $\mathcal{S}(G)_m \subseteq \mathcal{S}(G)$ contains the m subgraphs with the highest similarity to the question, which is selected for subsequent processing.

4.2 Graph-based information process

For the retrieved subgraphs, using GNNs to transform structured graph data into representations that are understandable by LLMs is a commonly used method. However, the graph representations learned by GNNs tend to cluster in some areas of the space [17], forming multiple isolated clusters with sharp boundaries called "cliffs." This issue stems from the fact that GNNs rely on fixed aggregation rules (e.g., sum, mean, or max) to propagate features; these simple linear rules are inadequate for modeling complex semantic interactions between nodes. While prior graph diffusion models also employ diffusion processes, they rely on isotropic noise that treats all edges uniformly, failing to preserve semantic proximity between propositions. DDM addresses this limitation by leveraging the advantages of the direction-aware diffusion mechanism. The original features are diffused into the surrounding semantic space by adding and removing iterative noise. This process allows features to transition smoothly across the semantic space, preventing the loss of fine-grained semantic information and maintaining the feature space's continuity. As a result, the model can generate continuous and semantically rich representations essential for high-quality soft prompts. Moreover, during the denoising process, the model learns to distinguish between noise and valuable information, retaining information that aids the task. Since there is a representation gap between the graph encoder and LLM, we first aggregate the node embeddings from the DDM-refined subgraph into a single graph-level vector via attention-based pooling. A multilayer perceptron (MLP) then aligns this pooled vector with the vector

Just Accepted

Table 1 Main experimental results. Bold value indicates the best performance among all methods in this model.

Model	Method	HotpotQA		2WikiMultiHopQA		TriviaQA	
		EM	F1	EM	F1	EM	F1
Mistral-7B	Direct [34]	16.5	24.6	15.1	23.9	33.2	41.3
	Naive RAG [3]	19.1	26.8	18.3	25.7	37.7	46.7
	IRCoT [10]	20.2	28.5	21.3	28.9	38.0	47.9
	REPLUG [35]	19.9	27.7	22.6	31.9	42.7	50.3
	Self-RAG [7]	23.7	30.6	24.4	32.1	41.9	49.4
	RECOMP [8]	24.3	32.7	25.7	34.2	44.4	51.8
	RARTOR [36]	25.8	34.2	27.0	35.8	44.6	53.2
	AdaComp [37]	27.7	34.9	26.4	35.1	46.6	54.3
	HippoRAG [38]	27.1	34.5	27.2	35.3	45.6	53.7
	GraphRAG [14]	28.3	35.7	26.8	35.4	45.9	54.8
	FG-RAG(ours)	29.1 (+0.8)	37.8 (+2.1)	28.9 (+2.1)	37.7 (+2.3)	47.8 (+1.2)	55.7 (+0.9)
LLaMA-2-7B	Direct [34]	14.7	22.6	13.3	21.9	30.5	38.3
	Naive RAG [3]	16.5	24.4	17.7	26.6	38.2	45.6
	IRCoT [10]	17.0	26.2	20.3	28.1	36.7	45.4
	REPLUG [35]	17.5	25.1	19.7	28.8	41.7	50.6
	Self-RAG [7]	21.9	29.5	22.6	32.0	43.5	51.3
	RECOMP [8]	24.6	31.8	23.3	32.5	42.8	50.2
	RARTOR [36]	23.4	32.0	23.5	33.7	41.6	51.1
	AdaComp [37]	26.4	33.5	25.8	33.6	45.1	53.8
	HippoRAG [38]	25.2	32.2	24.8	34.1	44.2	52.6
	GraphRAG [14]	26.5	32.9	25.9	34.3	45.5	53.6
	FG-RAG(ours)	27.4 (+0.9)	35.6 (+1.7)	26.7 (+0.8)	35.2 (+0.9)	46.9 (+1.4)	54.2 (+0.6)
Flan-T5-Large	Direct [34]	13.6	22.1	12.5	20.7	30.1	37.7
	Naive RAG [3]	15.2	23.4	15.8	23.4	37.6	44.2
	IRCoT [10]	16.3	23.7	18.3	25.8	37.2	44.6
	REPLUG [35]	16.2	23.9	18.2	26.0	41.6	48.9
	Self-RAG [7]	18.6	27.2	21.3	28.7	40.6	49.1
	RECOMP [8]	23.5	31.8	22.2	29.8	42.2	50.3
	RAPTOR [36]	23.8	32.6	25.1	32.4	43.4	52.6
	AdaComp [37]	24.9	32.6	23.8	32.5	44.0	52.3
	HippoRAG [38]	24.7	32.9	24.4	33.8	43.3	53.5
	GraphRAG [14]	25.8	33.4	24.0	32.1	42.9	52.2
	FG-RAG(ours)	26.4 (+0.6)	34.8 (+1.4)	25.6 (+1.6)	33.8 (+1.7)	45.2 (+2.3)	54.1 (+1.9)

Just Accepted

space of the LLM. The resulting encoded graph representation is treated as a soft prompt p for LLMs, it can be defined as:

$$p = \text{MLP}_{\phi}(POOL(DDM(sub))) \in \mathbb{R}^{d_{LLM}}, \quad (8)$$

where d_{LLM} is the LLM's hidden embedding dimension.

4.3 Reinforcement learning-based prompt ordering

To align the ordering preferences of the large language model concerning these soft prompts, the top-l preferred prompts are selected from m prompts. The process can be formalized as an MDP $M = (S, A, \mathcal{T}, \mathcal{R}, \pi)$. Selecting the preferred soft prompts can be considered a sequential decision-making task. At each time step, the agent selects the most suitable prompt from the remaining candidates for a specific position in the sequence. Specially, at time step t, the action $a_t \in A(s_t)$ select a prompt p for the position t+1.The MDP model M is defined by the following components: states, actions, transition dynamics, reward function, and policy, which are detailed as follows:

States S. At each time step t, state s_t represents the current status the agent is faced with. Thus, the state at step t is defined as:

$$s_t = \{\mathcal{C}_t, \mathcal{U}_t\}, \quad (9)$$

where $\mathcal{C}_t = (p_1, p_2, \ldots, p_t)$ is the sequence of prompts selected up to time step t, and $\mathcal{U}_t \in \mathcal{P}$ is the set of remaining(unselected) candidate prompts, with $\mathcal{U}_t = \mathcal{P} \setminus (p_1, p_2, \ldots, p_t)$. Here, the operator $\setminus$ denotes set difference, meaning $\mathcal{U}_t$ contains all prompts in $\mathcal{P}$ that have not yet been chosen.

Actions A. At each time step t, the action $a_t \in A(s_t)$ involves selecting a prompt p from the remaining candidate prompts to append to the sequence. The set of available actions

$A(s_t)$ is dynamically determined by the current state s_t, which includes the partially constructed sequence and the remaining candidates.

Transition Dynamics $\mathcal{T}$. The transition function $\mathcal{T}(s_{t+1}|s_t, a_t)$ defines the probability of transitioning to state s_{t+1} after taking action a_t in state s_t. Given the formulation of state and action, the next state s_{t+1} is deterministic. Specifically, if the current state is $s_t = (\mathcal{C}_t, \mathcal{U}_t)$, the next state s_{t+1} is uniquely determined by the action a_t, which selects a prompt $p_{t+1} \in \mathcal{U}_t$. Formally, the transition function is defined as:

$$\mathcal{T}(s_{t+1} \mid s_t, a_t) = \begin{cases} 1 & \text{if } s_{t+1} = \{(\mathcal{C}_t, p_{t+1}), \mathcal{U}_t \setminus (p_{t+1})\}, \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

where s_{t+1} is constructed by appending the selected prompt p_{t+1} to the sequence $\mathcal{C}_t$ and removing p_{t+1} from the set of remaining prompts $\mathcal{U}_t$.

Reward Function $\mathcal{R}$.The reward function $\mathcal{R}(s_t, a_t)$ evaluates the quality of the combined prompt P. This work uses the weighted sum of relevance score and answer similarity as the reward:

$$\mathcal{R} = \alpha \cdot \cos(E(q), E(P)) + (1 - \alpha) \cdot \mathrm{EM}(y, \hat{y}), \tag{11}$$

where $E(q)$ denotes the embedding of the input query q, and $E(P)$ denotes the embedding of the combined prompt P. These embeddings are generated using a common pre-trained language model encoder to capture semantic representations. The term $\cos(\cdot, \cdot)$ represents the cosine similarity between the query embedding and the prompt embedding, measuring their semantic relevance. The variables y and $\hat{y}$ are the LLM's actual output and expected output, respectively, and α is a balancing factor. EM represents the exact match between LLM's actual output and expected output. These two parts correspond to the retrieval quality and response quality within the RAG system, respectively.

policy π. The policy describes the agent's behaviors and is a probabilistic distribution of possible actions at time step t.

In parameter optimization, this work opts for the PPO [39] algorithm primarily for the following reasons: First, it boasts excellent stability. The PPO algorithm employs the clip mechanism to restrict the magnitude of policy updates, thereby preventing excessive policy changes and ensuring the stability of the optimization process. Second, the reward signal in this scenario is relatively sparse, as rewards are only obtained after the selection is completed. In this case, the value network of PPO can effectively assist us in better estimating the value of intermediate states, thus enabling effective optimization even with sparse rewards.

5 Experimental Setups

Datasets. To comprehensively evaluate performance across datasets with diverse characteristics, this work utilizes the following three representative datasets: TriviaQA [40], HotpotQA [41], and 2WikiMultiHopQA [42]. Among these, TriviaQA focuses on single-hop reasoning tasks, while HotpotQA and 2WikiMultiHopQA are designed for multi-hop reasoning tasks.

Models. Since the inputs to the large language model are soft prompts, the open-source model must enable users to process inputs directly without relying on API calls. Accordingly, the performance of the proposed framework is evaluated using Mistral-7B [43], LLaMA-2-7B [44], and Flan-T5-Large [45].

Retriever and Corpus.Fair evaluation requires the same retriever to search the same corpus across different approaches. Accordingly, Contriever-MS-MARCO [46] is used as the retriever on the Wikipedia corpus from December 20, 2018 [47], with articles segmented into non-overlapping 100-word fragments.

Metrics. The Exact Match (EM) and F1 score for the answer strings serve as the performance metrics. EM assesses exact correctness, while F1 evaluates answers that are close to but not necessarily exact, providing a nuanced view of how well-predicted answers overlap with the correct ones.

Baselines. To ensure a comprehensive evaluation and fair comparison, identical datasets, retrievers, and corpora are utilized to benchmark the proposed method against the following baselines:

Direct [34] involves presenting questions directly to the LLM, prompting it to generate corresponding answers without explanations.

Naive RAG [3] assists in answering questions by retrieving information from external documents. The retrieved information and the question are concatenated as input in the experiment.

IRCoT [10] enhances each step of the chain-of-thought generation process by incorporating knowledge retrieval steps during the generation process.

REPLUG [35] adapts the framework to the corresponding downstream tasks by fine-tuning the Retriever. This method enhances retrieval effectiveness by improving the relevance of the retrieved text.

Self-RAG [7] provides a framework by training an LLM to learn specific reflection tokens, thereby controlling the decision of whether to retrieve during reasoning and examining the relevance of the retrieved content.

RECOMP [8] compresses retrieved documents into textual summaries before prepending them to improve the efficiency of retrieval-augmented language models with minimal performance loss.

AdaComp [37] is a low-cost extractive context compression method that adaptively determines the compression rate based on query complexity and retrieval quality for RAG systems.

RAPTOR [36] builds a hierarchical tree index via recursive summarization, enabling retrieval of both fine-grained details and high-level themes from lengthy documents.

HippoRAG [38] mimics human long-term memory by using an LLM to construct a knowledge graph and employing Personalized PageRank for efficient multi-hop retrieval.

GraphRAG [14] constructs a knowledge graph from documents and retrieves information based on graph structures to capture complex interconnections beyond text-only methods.

Implementation details. For the projection MLP, we employ a simple two-layer architecture. The input layer receives the DDM-refined graph embedding, followed by a hidden linear layer with 2048 units and a GELU activation. A second linear layer projects to d_{LLM}, with dropout 0.1 applied after GELU. For proposition transfer, this work utilizes the propositionizer-wiki-flan-t5-large model. The optimization

Just Accepted

process employs the AdamW [48] optimizer with an initial learning rate of 1×10^{-5} and a weight decay of 0.05. Each experiment runs for a maximum of 20 epochs, with an early stopping mechanism implemented to prevent overfitting and ensure training efficiency. Each experiment on different datasets is replicated three times to ensure robustness and fairness, and the average results are reported as the final experimental outcomes.

6 Experiment Results

6.1 Main results

The main results are presented in Table 1. Based on these results, several key observations can be made regarding the performance and effectiveness of the proposed methods.

Superior Performance Across Datasets. The proposed FG-RAG framework demonstrates superior performance, outperforming other methods across all three datasets when evaluated with LLaMA-2-7B and Mistral-7B. On average, FG-RAG achieves significant performance improvements compared to the baseline without retrieval. When compared to the baseline with retrieval, FG-RAG not only surpasses all existing methods but also realizes average performance enhancements of 1.5% in EM score and 1.8% in F1 score over the best-performing method. This superior performance underscores the effectiveness of the fine-grained prompt generation and dynamic prompt permutation approach in enhancing the quality of inputs.

Importance of Semantic Optimization. The results reveal that semantic optimization is critical for further improving performance. By optimizing the semantic structure of retrieved information through advanced graph-based techniques and dynamically aligning the input with the language model's preferences for prompt ordering, FG-RAG ensures a more coherent and contextually grounded generation process. For instance, on the TriviaQA dataset, the direct application of RAG leads to a performance decline, with an average drop of 9.3% in EM score and 9.2% in F1 score, primarily due to the negative impact of sparse or irrelevant contents.

Mitigation of Hallucination. The proposed framework, equipped with a semantic optimization mechanism, ensures that the generated responses are grounded in accurate and relevant external knowledge, thereby reducing the likelihood of fabricated or irrelevant content. Compared to the LLaMA-2-7B + Direct approach, the enhanced LLaMA-2-7B + FG-RAG system achieves an average 13.6% improvement in EM score and 13.9% improvement in F1 score in performance across three datasets.

Advantages Over Graph-Based Methods. While recently proposed graph-enhanced approaches like GraphRAG and RAPTOR improve multi-hop reasoning by capturing document-level structures, and HippoRAG excels at associative retrieval through graph-based memory mechanisms, they primarily operate on static graph structures. In contrast, FG-RAG's dynamic proposition graph combined with directional diffusion enables finer-grained semantic denoising and query-adaptive subgraph retrieval. This results in consistently higher EM and F1 scores across all datasets, highlighting the advantage of fine-grained, dynamically optimized graph representations over static and coarse-grained graph structures.

6.2 Ablation Studies

To validate the contributions of each component in FG-RAG, ablation studies are performed on the LLaMA-2-7B model using two multi-hop QA benchmarks: HotpotQA and 2WikiMultiHopQA. The evaluation employ EM and F1 scores as performance metrics, with the principal findings detailed in Table 2. The ablation studies included the following variants:

- Ablation 1 (No Extraction): This variant removes the information extraction module and replaces proposition transfer with raw document chunks.
- Ablation 2 (No Graph): This variant disables the graph-based retrieval module and replaces it with similarity-based retrieval (proposition level).
- Ablation 3 (Exchange DDM with GNNs): This variant replaces the DDM with a standard Graph Convolutional Network (GCN).
- Ablation 4 (No MLP): This variant Removes the MLP_ϕ projection layer that aligns graph representations with the LLM's embedding space, feeding DDM-refined graph embeddings directly into the LLM.
- Ablation 5 (No RL): This variant removes the RL-based prompt sequencing module and replaces it with random prompt ordering.
- Ablation 6 (Exchange PPO with other RL Algorithm): This variant replaces PPO with REINFORCE to investigate the influence of different policy optimization strategies on prompt ordering performance.

Table 2 Performance comparison in EM socre and F1 socre on hotpotQA and 2wikiMultihopQA datasets

methods	HotpotQA(EM)	HotpotQA(F1)
w/o Exaction	23.5	31.2
w/o Graph	21.0	28.3
w/o DDM, use GCN	27.2	34.2
w/o MLP	22.5	30.8
w/o RL	25.7	32.2
w/o RL, use REINFORCE	25.8	33.5
w/ all modules	27.4	35.6
methods	2WikiMHQA(EM)	2WikiMHQA(F1)
w/o Exaction	22.7	28.9
w/o Graph	22.4	29.7
w/o DDM, use GCN	25.2	33.8
w/o MLP	23.1	30.8
w/o RL	24.9	31.6
w/o RL, use REINFORCE	25.2	33.2
w/ all modules	26.7	35.2

Information Extraction is Critical for Reducing Redundancy: Removing the proposition extraction module (Ablation 1) leads to a significant performance drop of 3.9% in EM and 3.4% in F1 on HotpotQA. This underscores the importance of extracting standalone, meaningful propositions from raw documents to reduce redundancy. Graph-Based Retrieval Enables Precise Multi-Hop Reasoning: Ablation 2 (no graph) demonstrates

Just Accepted

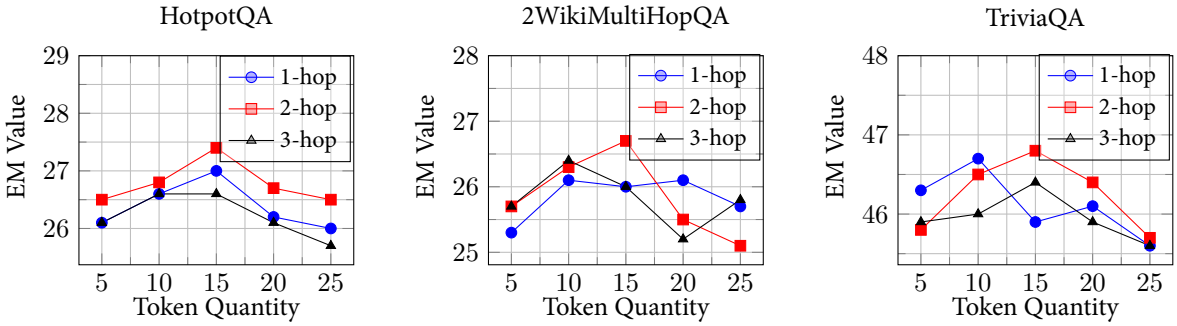

Fig. 2 Comparison of 1-hop, 2-hop and 3-hop EM values for different token quantity

the limitations of similarity-based retrieval, with EM dropping by 6.4% on HotpotQA and 4.3% on 2WikiMultiHopQA. This degradation is particularly pronounced in multi-hop questions, where semantic connections between propositions are crucial. The graph-based approach in FG-RAG effectively captures these connections, enabling more accurate and contextually grounded answers. DDM Outperforms Standard GNNs in Denoising and Semantic Preservation: Replacing DDM with a standard GCN (Ablation 4) leads to a performance drop of 0.2% in EM and 1.4% in F1 on HotpotQA, and 1.5% in EM and 1.4% in F1 on 2WikiMultiHopQA. This highlights the superiority of DDM's directional diffusion mechanism over conventional GNNs. Specifically, DDM introduces a diffusion process that iteratively refines node features, resulting in smoother and more accurate graph representations, while GNNs propagate features through fixed aggregation rules. MLP Projection Layer is Essential for Cross-Modal Alignment: Removing the MLP_ϕ projection layer (Ablation 4) causes a substantial performance degradation of 4.9% EM and 4.8% F1 on HotpotQA, and 3.6% EM and 4.4% F1 on 2WikiMultiHopQA. This drop confirms that directly feeding graph-derived representations into LLMs without proper alignment significantly harms comprehension. The MLP layer effectively bridges the representation gap between the graph encoder and the LLM's semantic space, enabling the model to interpret soft prompts faithfully. RL-Based Prompt Ordering Enhances Coherence: Randomizing prompt order (Ablation 5) results in a 3.4% drop in F1 on HotpotQA and a 3.6% drop in F1 on 2WikiMultiHopQA, with minimal impact on EM. This suggests that the RL-based ordering mechanism ensures that prompts are arranged in a way that aligns with the LLM's reasoning process, reducing contradictions and improving fluency in generated responses. PPO Outperforms REINFORCE in Sparse-Reward Settings: Replacing PPO with REINFORCE (Ablation 6) yields a performance drop of 1.6% EM and 2.1% F1 on HotpotQA, and 1.5% EM and 2.0% F1 on 2WikiMultiHopQA. This gap is attributed to PPO's clipped surrogate objective and value network, which stabilize policy updates and better estimate state values under sparse reward signals. In contrast, REINFORCE suffers from high variance and slower convergence, making it less suitable for the prompt ordering task where rewards are only obtained at the end of the episode.

6.3 Analyzing Graph Hop Count and Token Quantity in FG-RAG

To investigate the impact of critical hyperparameters in the proposed graph-based retrieval component, this work conduct ablation experiments on two key factors: (1) the hop count k during subgraph retrieval (i.e., the breadth of semantic relationships considered) and (2) the number of graph tokens N generated during graph-to-prompt conversion (i.e., the compression level of subgraph information). All experiments use the same backbone LLM (LLaMA-2-7B) and retrieval corpus to ensure a fair comparison. This section evaluated FG-RAG on the HotpotQA, 2WikiMultiHopQA, and TriviaQA benchmarks under varying configurations, with the detailed results illustrated in Figure 2:

- Hop count k: Tested $k \in \{1, 2, 3\}$, controlling the contextual scope of retrieved subgraphs.
- Graph token quantity N: Evaluated $N \in \{5, 10, 15, 20, 25\}$, analyzing the trade-off between semantic richness and information conciseness.

Hop count k significantly impacts retrieval quality, with $k = 2$ achieving optimal performance. When k increases from 1 to 2, EM improves by 0.1%, as more prominent hops capture more prosperous contextual relationships between propositions. However, further expansion to $k = 3$ degrades performance by 0.3%, likely due to noise propagation from distant nodes. Graph token quantity N balances semantic coverage and information conciseness, with $N = 15$ yielding the best results. Using $N = 15$ tokens outperforms $N = 5$ by 0.6% in EM, due to insufficient semantic coverage with $N = 5$, and $N = 25$ by 0.7% in EM, as $N = 25$ introduces excessive noise. This demonstrates that moderate tokenization granularity is essential for effectively encoding subgraph information.

6.4 Analyzing the Balance Factor in Reward Function

This work systematically analyzes the impact of the balance factor α on the RL-based prompt ranking method. The agent receives candidate prompts and learns to optimize their order via PPO, guided by a reward function combining answer quality and query-prompts similarity. All experiments use the same backbone LLM (LLaMA-2-7B) and retrieval corpus to ensure a fair comparison. To understand the individual contributions of each reward component, this work

Just Accepted

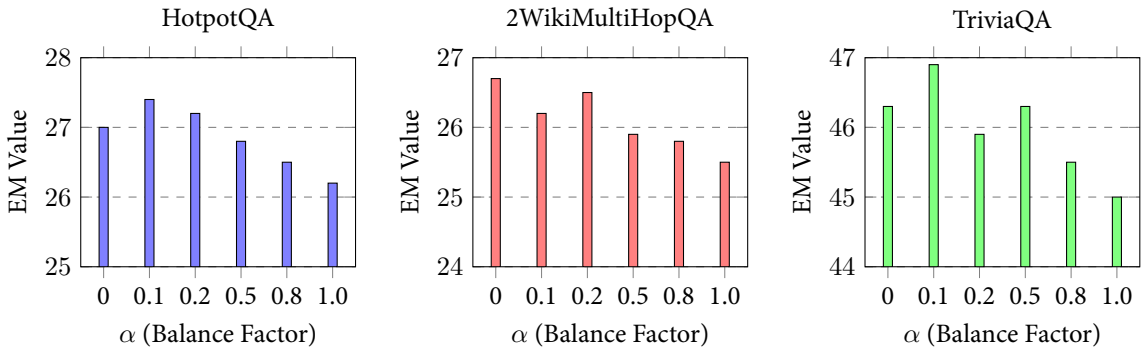

Fig. 3 Comparison of EM Values for different α (Balance Factor) values

also analyzes the extreme cases where $\alpha = 0$ (reward based solely on EM) and $\alpha = 1.0$ (reward based solely on cosine similarity). FG-RAG is evaluated on the HotpotQA, 2WikiMultiHopQA, and TriviaQA benchmarks under varying configurations, with the detailed results illustrated in Figure 3:

- Balance factor α: Test $\alpha \in \{0, 0.1, 0.2, 0.5, 0.8, 1.0\}$, controlling the influence of query-prompts similarity on experimental results.

Incorporating relevance constraints ($\alpha > 0$) enhances model robustness, with optimal performance achieved at $\alpha = 0.1$. When α increases from 0 to 0.1, EM improves by 0.2%, as relevance constraints help mitigate local optima. However, further increasing α to 0.8 degrades EM by 0.6%, as excessive focus on similarity disrupt the RL agent's ability to prioritize task-solving prompts.

7 Conclusions

This paper has introduced FG-RAG, a novel fine-grained framework designed to address the limitations of existing RAG methods, such as redundancy in retrieved documents and suboptimal prompt ordering. The proposed framework contains two key components: (1) Refined Prompt Generation, where standalone, meaningful propositions are first extracted from raw documents and organized into a semantic graph. Relevant subgraphs are then retrieved and processed through a DDM—a graph neural network tailored for denoising and refining semantic relationships. This enables the conversion of graph-structured information into soft prompts that are both concise and interpretable by LLMs; and (2) Prompt Ordering, which optimizes prompt ordering through reinforcement learning to maximize LLM performance. This work reduces noise and redundancy in retrieved information and provides a more structured and semantically enriched input for LLMs, paving the way for more reliable and efficient generative systems. However, several limitations remain: (a) graph construction introduces significant computational overhead versus standard chunking; (b) performance degrades when proposition extraction fails on domain-specific or informal texts; (c) the PLM encoder propagates pre-training biases to retrieved contexts; and (d) RL-based ordering struggles with very long prompt sequences (>50) due to exploration inefficiency.

Beyond these technical challenges, this work also acknowledges broader societal implications. The improved RAG system could potentially be misused to generate convincing misinformation, necessitating safeguards in deployment. Additionally, the Wikipedia corpus used may contain inherent cultural and linguistic biases that could propagate through the system. Finally, the two-stage training process introduces non-trivial energy consumption, motivating future work on more efficient model compression and inference techniques. Addressing these concerns will be essential for responsible deployment.

While FG-RAG demonstrates strong performance on QA benchmarks, we acknowledge its limited evaluation scope, future work will extend FG-RAG to multi-document summarization and cross-lingual settings.

8 Acknowledgement

This work was supported by the Natural Science Foundation of China under Grant 62402321.

Dates

Received: 29 December 2025; Revised: 13 April 2026;
Accepted: 20 April 2026;

References

[1] Qiushi Huang, Shuai Fu, Xubo Liu, Wenwu Wang, Tom Ko, Yu Zhang, and Lilian H. Y. Tang. Learning retrieval augmentation for personalized dialogue generation. In Conference on Empirical Methods in Natural Language Processing, 2024.

[2] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Qianyu Guo, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. ArXiv, abs/2312.10997, 2023.

[3] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. Retrieval augmented language model pre-training. In Proceedings of the International conference on machine learning, pages 3929–3938, 2020.

[4] Yuze Liu, Tingjie Liu, Tiehua Zhang, Youhua Xia, Jinze Wang, Zhishu Shen, Jiongdao Jin, and Fei Richard Yu. GRL-Prompt: Towards

Just Accepted

knowledge graph based prompt optimization via reinforcement learning. ArXiv, abs/2411.14479, 2024.

[5] Guojie LI. Big data and computing models. Big data research, 10(1):9–16, 2024.

[6] Yulin HE, Dongtong WU, Fournier-Viger Philippe, and Huang Zhexue. First filling strategy-based partitioning method to balance data in spark. Acta Electronica Sinica, 52(10):3322–3335, 2024.

[7] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In The Twelfth International Conference on Learning Representations, 2023.

[8] Fangyuan Xu, Weijia Shi, and Eunsol Choi. RECOMP: Improving retrieval-augmented LMs with compression and selective augmentation. ArXiv, abs/2310.04408, 2023.

[9] Shengnan An, Zexiong Ma, Zeqi Lin, Nanning Zheng, Jian-Guang Lou, and Weizhu Chen. Make your LLM fully utilize the context. Proceedings of the Advances in Neural Information Processing Systems, 37:62160–62188, 2024.

[10] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In Proceedings of the Annual Meeting of the Association for Computational Linguistics, pages 10014–10037, 2023.

[11] Ge Gao, Alexey Taymanov, Eduardo Salinas, Paul Mineiro, and Dipendra Misra. Aligning llm agents by learning latent preference from user edits. Proceedings of the Advances in Neural Information Processing Systems, 37:136873–136896, 2025.

[12] Zichuan Fu, Wentao Song, Yejing Wang, Xian Wu, Yefeng Zheng, Yingying Zhang, Derong Xu, Xuetao Wei, Tong Xu, and Xiangyu Zhao. Sliding window attention training for efficient large language models. ArXiv, abs/2502.18845, 2025.

[13] Junde Wu, Jiayuan Zhu, and Yunli Qi. Medical graph RAG: Towards safe medical large language model via graph retrieval-augmented generation. ArXiv, abs/2408.04187, 2024.

[14] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. From local to global: A graph RAG approach to query-focused summarization. ArXiv, abs/2404.16130, 2024.

[15] Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. Is chatgpt good at search? investigating large language models as re-ranking agents. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pages 14918–14937, 2023.

[16] Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Le Yan, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, et al. Large language models are effective text rankers with pairwise ranking prompting. In Proceedings of the Annual Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics, 2024.

[17] Run Yang, Yuling Yang, Fan Zhou, and Qiang Sun. Directional diffusion models for graph representation learning. Proceedings of the Advances in Neural Information Processing Systems, 36:32720–32731, 2023.

[18] Lingling Xu, Haoran Xie, S. Joe Qin, Xiaohui Tao, and Fu Lee Wang. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment. ArXiv, abs/2312.12148, 2023.

[19] Ruidan He, Linlin Liu, Hai Ye, Qingyu Tan, Bosheng Ding, Liying Cheng, Jiawei Low, Lidong Bing, and Luo Si. On the effectiveness of adapter-based tuning for pretrained language model adaptation. In Proceedings of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing, pages 2208–2222, 2021.

[20] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. Proceedings of the International Conference on Learning Representations, 1(2):3, 2022.

[21] Yoav Levine, Itay Dalmedigos, Ori Ram, Yoel Zeldes, Daniel Jannai, Dor Muhlgay, Yoni Osin, Opher Lieber, Barak Lenz, Shai Shalev-Shwartz, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. Standing on the shoulders of giant frozen language models. ArXiv, abs/2204.10019, 2022.

[22] Jibing Gong, Xiaohan Fang, Wenhai Zhu, Jiquan Peng, Lin Wang, Mengpan Chen, Jiangtao Zhang, and Jie Tang. Knit: Toward alleviating large language model fact knowledge hallucinations in knowledge graph completion. Big Data Mining and Analytics, 9(2):611–631, 2026.

[23] Danyang Wang, Xuan Zhang, Zhi Jin, Chen Gao, Kunpeng Du, Ming Zheng, and Tong Li. Tdp-fkgc: Task-guided diffusion prototype network for few shot knowledge graph completion. Big Data Mining and Analytics, 2025.

[24] Bryan Perozzi, Bahare Fatemi, Dustin Zelle, Anton Tsitsulin, Mehran Kazemi, Rami Al-Rfou, and Jonathan J. Halcrow. Let your graph do the talking: Encoding structured data for LLMs. ArXiv, abs/2402.05862, 2024.

[25] Jaehyeong Jo, Seul Lee, and Sung Ju Hwang. Score-based generative modeling of graphs via the system of stochastic differential equations. In International Conference on Machine Learning, 2022.

[26] Benjamin Paul Chamberlain, James Rowbottom, Maria I. Gorinova, Stefan Webb, Emanuele Rossi, and Michael M. Bronstein. Grand: Graph neural diffusion. ArXiv, abs/2106.10934, 2021.

[27] Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Jessie Li. Drn: A deep reinforcement learning framework for news recommendation. Proceedings of the 2018 World Wide Web Conference, 2018.

[28] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. ACM Transactions on Information Systems (TOIS), 20(4):422–446, 2002.

[29] Guoxin Chen, Minpeng Liao, Peiying Yu, Dingmin Wang, Zile Qiao, Chao Yang, Xin Zhao, and Kai Fan. C-3po: Compact plug-and-play proxy optimization to achieve human-like retrieval-augmented generation. ArXiv, abs/2502.06205, 2025.

[30] Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric P. Xing, and Zhiting Hu. Rlprompt: Optimizing discrete text prompts with reinforcement learning. In Conference on Empirical Methods in Natural Language Processing, 2022.

[31] Tianjun Zhang, Xuezhi Wang, Denny Zhou, Dale Schuurmans, and Joseph E. Gonzalez. Tempera: Test-time prompting via reinforcement learning. ArXiv, abs/2211.11890, 2022.

[32] Tong Chen, Hongwei Wang, Sihao Chen, Wenhao Yu, Kaixin Ma, Xinran Zhao, Hongming Zhang, and Dong Yu. Dense X retrieval: What retrieval granularity should we use? In Proceedings of the Conference on Empirical Methods in Natural Language Processing, pages 15159–15177, 2024.

Just Accepted

[33] Yuntong Hu, Zhihan Lei, Zheng Zhang, Bo Pan, Chen Ling, and Liang Zhao. Grag: Graph retrieval-augmented generation. In Findings of the Association for Computational Linguistics: NAACL 2025, pages 4145–4157, 2025.

[34] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. Proceedings of the Advances in Neural Information Processing Systems, 33:1877–1901, 2020.

[35] Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen Tau Yih. Replug: Retrieval-augmented black-box language models. In Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics, pages 8364–8377, 2024.

[36] Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. Raptor: Recursive abstractive processing for tree-organized retrieval. ArXiv, abs/2401.18059, 2024.

[37] Qianchi Zhang, Hainan Zhang, Liang Pang, Hongwei Zheng, and Zhiming Zheng. Adacomp: Extractive context compression with adaptive predictor for retrieval-augmented large language models. ArXiv, abs/2409.01579, 2024.

[38] Bernal Jimenez Gutierrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. Hipporag: Neurobiologically inspired long-term memory for large language models. ArXiv, abs/2405.14831, 2024.

[39] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. ArXiv, abs/1707.06347, 2017.

[40] Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In Proceedings of the Annual Meeting of the Association for Computational Linguistics, pages 1601–1611, 2017.

[41] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, pages 2369–2380, 2018.

[42] Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In Proceedings of the International Conference on Computational Linguistics, pages 6609–6625, 2020.

[43] Albert Qiaochu Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, L'elio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b. ArXiv, abs/2310.06825, 2023.

[44] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. ArXiv, abs/2307.09288, 2023.

[45] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. Journal of Machine Learning Research, 25(70):1–53, 2024.

[46] Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane A. Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. Few-shot learning with retrieval augmented language models. J. Mach. Learn. Res., 24:251:1–251:43, 2022.

[47] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, pages 6769–6781, 2020.

[48] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. ArXiv, abs/1711.05101, 2017.

9 Appendix

9.1 Notation Table

This paper provide a comprehensive summary of the notations used throughout the paper. Table 4 defines all symbols and indicates the section where they first appear.

9.2 the structure of the MLP

For the projection MLP, denoted as MLP_ϕ in Eq.(8), this work employ a simple yet effective two-layer architecture. The input layer takes the DDM-refined graph embedding vector x_{sub} with dimension d_{PLM}. This is followed by a hidden linear layer that projects the input to an intermediate dimension of 2048, after which a GELU activation function is applied. A second linear layer then projects the 2048-dimensional vector to the final output dimension d_{LLM}, which corresponds to the hidden state dimension of the backbone language model. To mitigate overfitting, this work apply a dropout rate of 0.1 after the GELU activation in the hidden layer.

9.3 Implementation of PPO

(a) PPO Network Architecture

In PPO implementation, this work adopt an Actor-Critic architecture to optimize the prompt ordering policy. The Actor network encodes the current state $s_t = (C_t, \mathcal{U}_t)$, where C_t denotes the sequence of prompts selected up to time step t and $\mathcal{U}_t$ represents the set of remaining candidate prompts. This state representation is processed through a Transformer-based encoder that captures the semantic relationships between selected and candidate prompts. The network then outputs a probability distribution $\pi(a_t|s_t)$ over the available actions, guiding the selection of the next prompt to append to the sequence. The Critic network shares the same encoder architecture but is followed by a fully connected layer that produces a scalar value $V(s_t)$, which estimates the expected cumulative reward from the current state. This value estimate is essential for computing the advantage function during policy updates, enabling more stable and efficient training.

(b) Dynamic Action Masking

To accommodate the varying size of $\mathcal{U}_t$ across different time steps, this work introduce a dynamic action masking mechanism. Prompts in $\mathcal{U}_t$ are aggregated via pooling and concatenated with the embedding of C_t to form the state representation. Each candidate prompt is mapped to an action embedding, and the Actor computes logits exclusively for these valid actions. A masking operation then zeros out the probabilities of already-selected prompts to prevent repetition, with the final distribution obtained by applying softmax over the unmasked actions.

9.4 Technical detail of DDM

(a) Mathematical Formulation of the Forward Diffusion Process

In DDM, the diffusion process consists of a forward process that gradually adds noise and a reverse process that learns to denoise and recover the original data. Let $\text{X}_0 = \{x_{0,i}\}_{i=1}^N$ be the original node feature matrix,

Just Accepted

Table 4 Notation Table for FG-RAG

Symbol	Description	Section
A.1 General Notation		
q	Input query	3
D	Document corpus, $D = \{d_i\}_{i=1}^{N}$	3
d_i	Individual document in the corpus	3
N	Total number of documents in the corpus	3
k	Number of top-k documents retrieved by the retriever	3
D_{retr}^{k}	Set of top-k retrieved documents	3
G	Proposition graph	3
V	Set of nodes in the proposition graph	3
E	Set of edges in the proposition graph	3
T_n	Natural language attribute (proposition text) of node n	3
$\mathcal{S}(G)$	Set of subgraphs of proposition graph G	3
g	Individual subgraph	3
m	Number of top-m subgraphs retrieved for query	4.1
p	Individual soft prompt	3, 4.2
$\mathcal{P}$	Set of soft prompts, $\mathcal{P} = \{p_1, p_2, \ldots, p_m\}$	3
l	Length of the final selected and ordered prompt sequence ($l \leq m$)	3
P	Final combined prompt sequence input to LLM, $P = \langle p_1, p_2, \ldots, p_l \rangle$	3
Q	Query tokens	3
Y	Output token sequence generated by LLM, $Y = \{y_1, y_2, \ldots, y_r\}$	3
r	Length of LLM output sequence	3
θ	Parameters of the LLM	3
w	Hop count for constructing ego-graphs	4.1
x_{sub}	Embedding vector of a subgraph	4.1
x_q	Embedding vector of the query	4.1
A.2 Core Dimensional Notation		
d_{PLM}	Embedding dimension of the PLM. Dimension of subgraph embedding x_{sub} and query embedding x_q generated by the PLM (revised from original d in the paper).	4.1
d_{LLM}	Hidden dimension of the LLM. Dimension of the LLM's input space that soft prompts p need to match after MLP projection.	4.2
A.3 Reinforcement Learning Notation		
M	Markov Decision Process (MDP), $M = (S, A, \mathcal{T}, \mathcal{R}, \pi)$	4.3
S	Set of states in MDP	4.3
s_t	State at time step t, $s_t = (C_t, L_t)$	4.3
C_t	Sequence of prompts selected up to time step t	4.3
L_t	Set of remaining candidate prompts at time step t	4.3
A	Set of actions in MDP	4.3
a_t	Action taken at time step t, i.e., selecting a prompt p_{t+1} from remaining candidates	4.3
$\mathcal{T}$	State transition function	4.3
$\mathcal{R}$	Reward function	4.3
π	Policy	4.3
α	Balancing factor in the reward function	4.3
y	Actual output of the LLM	4.3
$\hat{y}$	Expected ground truth output	4.3
A.4 Other Model Notation		
$E_d(\cdot)$	Document encoder, maps documents to embeddings	3
$E_q(\cdot)$	Query encoder, maps queries to embeddings	3
$\text{PLM}(\cdot)$	Pre-trained Language Model used as encoder	4.1
$\text{POOL}(\cdot)$	Mean pooling operation	4.1
MLP_ϕ	Multi-Layer Perceptron parameterized by ϕ, maps graph representations from dimension d_{PLM} to LLM space of dimension d_{LLM}	4.2
$\text{DDM}(\cdot)$	Directional Diffusion Model	4.2
$\text{EM}(\cdot)$	Exact Match function	4.3

Just Accepted

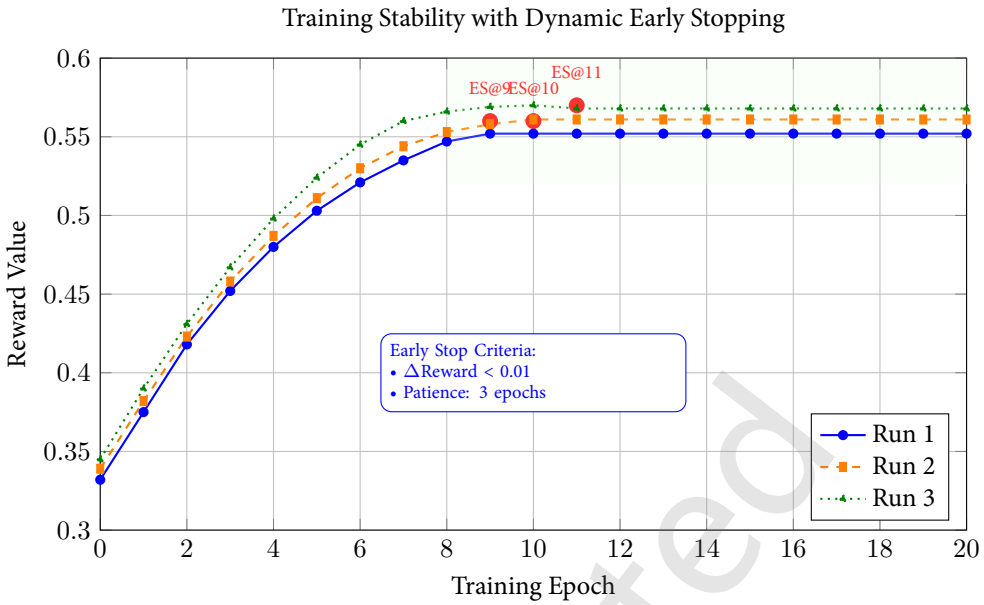

Fig. 4 Training stability with dynamic early stopping based on reward change threshold (ΔReward < 0.01). Three independent runs show consistent early stopping behavior, with triggers occurring at epochs 9, 10, and 11 respectively. All runs converge to stable reward values (0.55-0.57) with reduced variance after early stopping.

where $x_{0,i} \in \mathbb{R}^d$(d is the feature dimension), N is the number of nodes in the graph.

Forward Diffusion Process

The forward process is a discrete-time Markov chain that gradually adds directional noise to the original node features. At diffusion step t, the noisy feature $x_{t,i}$ is given by:

$$x_{t,i} = \sqrt{\bar{\alpha}_t}\, x_{0,i} + \sqrt{1-\bar{\alpha}_t}\, \epsilon'$$

Here $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$ is the cumulative product of the variance schedule. β_t is the noise schedule parameter(see Section (b) below). ϵ' is the directional noise, constructed through two key operations.

The directional noise ϵ' is generated through two sequential operations:

First, for anisotropic noise generation, the directional noise is not sampled from an isotropic Gaussian. Instead, it is data-dependent and batch-dependent. Let $\mu \in \mathbb{R}^d$ and $\sigma \in \mathbb{R}^d$ be the empirical mean and standard deviation of the node features in the current mini-batch. Then:

$$\bar{\epsilon} = \mu + \sigma \odot \epsilon, \quad \epsilon \sim \mathcal{N}(0, \mathrm{I}_d)$$

where $\odot$ denotes the Hadamard (element-wise) product. This formulation transforms the isotropic Gaussian noise ϵ into an anisotropic noise $\bar{\epsilon}$ that shares the same statistical profile as the batch data.

Second, for directional alignment, the noise is further aligned with the sign pattern of $x_{0,i}$ to preserve angular consistency. This is achieved through:

$$\epsilon' = \mathrm{sgn}(x_{0,i}) \odot |\bar{\epsilon}|$$

where $\mathrm{sgn}(\cdot)$ denotes the sign function, which extracts the sign of each element in the vector:

$$\mathrm{sgn}(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0 \end{cases}$$

and $|\cdot|$ ensures non-negative magnitudes. This alignment ensures ϵ' lies in the same orthant as $x_{0,i}$, preserving the relative signs of feature components.

Reverse Diffusion Process

The reverse process aims to recover the original features from the noisy ones. Since the forward process with directional noise does not admit a closed-form posterior, DDM adopts a direct prediction approach. A denoising network f_θ is trained to predict the clean features X_0 from the noisy input X_t and the graph structure A. The reverse process is formulated as:

$$\hat{\mathrm{X}}_0 = f_\theta(\mathrm{X}_t, \mathrm{A}, t)$$

where $\hat{\mathrm{X}}_0$ is the predicted clean feature matrix. The training objective minimizes the expected reconstruction error:

$$\mathcal{L} = \mathbb{E}_{\mathrm{X}_0, t} \left\| f_\theta(\mathrm{X}_t, \mathrm{A}, t) - \mathrm{X}_0 \right\|_2^2$$

where the expectation is taken over the data distribution and diffusion steps $t \sim \mathrm{Uniform}(1, T)$.

The denoising network f_θ typically employs a GNN-based encoder-decoder architecture with skip connections. During inference, starting from pure noise X_T, the model iteratively applies the reverse process to generate meaningful graph representations.

(b) Noise Schedule

The noise schedule $\{\beta_t\}_{t=1}^T$ follows a linear scheme similar to that used in DDPM. Specifically:

$$\beta_t = \frac{t}{T} \cdot (\beta_T - \beta_1) + \beta_1$$

Typical values are $\beta_1 = 10^{-4}$ and $\beta_T = 0.02$. The cumulative term $\bar{\alpha}_t$ thus decreases monotonically from nearly 1 to near 0, controlling the proportion of signal versus noise at each step.

Just Accepted

The key difference from vanilla diffusion is that the noise is not isotropic; it is directionally constrained and batch-adapted, which slows the decay of the signal-to-noise ratio (SNR) and allows the model to learn meaningful representations across a wider range of diffusion steps.

(c) Mathematical Definition and Implementation of "Directionality"

The directionality in DDM is realized through two complementary mechanisms:

Anisotropic Noise via Batch Statistics

By computing μ and σ per mini-batch, the noise becomes data-adaptive. This ensures that the diffusion process respects the local geometry of the feature distribution, preventing rapid washing-out of discriminative signals. This is especially important for graph data, which often exhibits strong anisotropy along a few principal directions.

Angular Preservation via Sign Alignment

The operation $\epsilon' = \text{sgn}(x_{0,i}) \odot |\bar{\epsilon}|$ guarantees that the noise vector lies in the same hyperoctant as the original feature. This avoids destructive interference that could occur if noise components opposed the original feature direction—a scenario that would accelerate the loss of semantic information.

Together, these constraints ensure that the forward process gradually corrupts the data while preserving its directional structure, enabling the denoising network to learn more informative representations.

9.5 Sensitivity to PPO-specific hyper-parameters

Experimental Setup. To investigate the sensitivity of FG-RAG to PPO-specific hyperparameters, systematic ablation studies were conducted on two key factors: the clip ratio ϵ and the generalized advantage estimation (GAE) λ. Experiments were performed on the HotpotQA dataset using LLaMA-2-7B as the backbone LLM. The clip ratio, which controls the magnitude of policy updates in PPO, was varied across $\{0.1, 0.2, 0.3, 0.4\}$. Concurrently, the GAE λ parameter, which balances bias and variance in advantage estimation, was evaluated at $\{0.90, 0.95, 0.97, 0.99\}$. All other hyperparameters were held constant, and each configuration was run with three random seeds to report average EM scores.

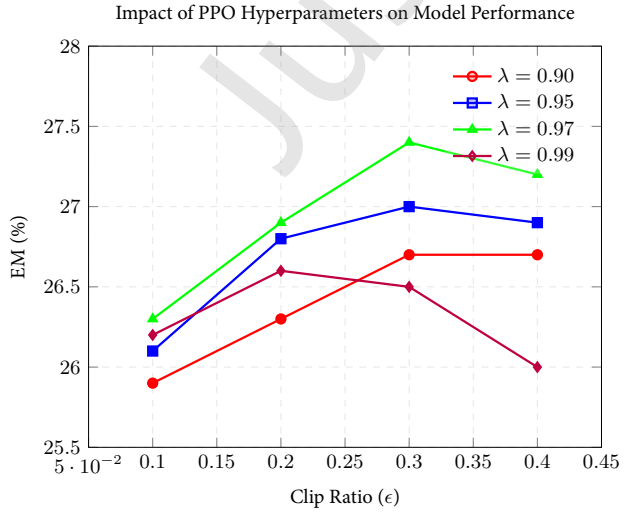

Fig. 5 Exact Match (EM) scores on HotpotQA across different clip ratio (ϵ) values for various GAE λ settings.

Experimental Conclusions. As shown in Figure 5, both hyperparameters significantly impact performance, with optimal results achieved at $\epsilon = 0.3$ and $\lambda = 0.97$ (27.4% EM). Different λ values exhibit distinct clip ratio preferences: $\lambda = 0.90$ peaks at $\epsilon = 0.4$, while $\lambda = 0.99$ favors smaller $\epsilon = 0.2$. The combination of moderate clipping ($\epsilon = 0.3$) and balanced advantage estimation ($\lambda = 0.97$) provides the best bias-variance trade-off for RL-based prompt optimization.

9.6 Training stability curves

Experimental Setup. This work analyze training stability on TriviaQA using LLaMA-2-7B. The RL agent is trained with PPO, employing early stopping (ΔReward < 0.01, patience=3). Three independent runs with different random seeds track reward mean and standard deviation across epochs.

Experimental Results. As shown in Figure 4, all runs converge to stable rewards (0.55-0.57) with early stopping triggered at epochs 9, 10, and 11. Reward variance decreases substantially from exploration phase to convergence phase, demonstrating improved training stability. Consistent performance across runs validates the robustness of our RL-based prompt ordering.

9.7 Proposition extraction quality

(a) extraction accuracy

Based on the paper Dense X Retrieval: What Retrieval Granularity Should We Use?, they conduct a manual error analysis to understand the quality of propositions generated by GPT-4 and the Propositionizer. While there does not exist a fixed standard on deciding a ground truth set of propositions for a passage, they estimate the frequency of error cases where (1) a proposition is not fully supported by the passage, (2) a proposition can be further split into separate propositions, and (3) propositions are not self-contained, respectively (Table 5). On a random sample of 50 passages, they observe that almost all propositions generated by both models are faithful, while a small portion of the propositions are not stand-alone.

Table 5 Frequency of errors occurred in the generated propositions.

	GPT-4	Propositionizer
Not Faithful	0.7% (3/408)	1.3% (6/445)
Not Minimal	2.9% (12/408)	2.0% (9/445)
Not Stand-alone	4.9% (20/408)	3.1% (14/445)

(b) proposition granularity consistency

Based on the paper Dense X Retrieval: What Retrieval Granularity Should We Use?, proposition granularity consistency is both rigorously defined and empirically validated through three core principles: atomicity, self-containment, and natural language format. Specifically, each proposition must represent a distinct, indivisible factoid that cannot be further split, while also incorporating all necessary contextual information to remain interpretable in isolation.

(c) failure cases.

While proposition-level retrieval demonstrates significant advantages over passage and sentence granularities, the quality of proposition extraction is not perfect. This subsection analyzes failure cases observed in the Propositionizer's output, categorized by error type, and traces their impact on downstream performance.

Type A: Not Self-Contained (Most Common). This error occurs when propositions retain unresolved references, making them semantically ambiguous when isolated from the original context.

Just Accepted

- Original Text: "Prior to restoration work performed between 1990 and 2001, the tower leaned at an angle of 5.5 degrees, but the tower now leans at about 3.99 degrees."
- Incorrect Proposition: "The tower now leans at about 3.99 degrees."
- Problem: The reference "the tower" is not resolved to "Leaning Tower of Pisa".
- Correct Proposition: "The Leaning Tower of Pisa now leans at about 3.99 degrees."
- Downstream Impact: (1) Retrieval Stage: Reduced semantic similarity with the query "What is the angle of the Tower of Pisa?" (2) Generation Stage: Ambiguity in entity resolution may lead to incorrect or uncertain answers.

Type B: Insufficient Splitting (Insufficient Granularity). This error occurs when a proposition contains multiple atomic facts that should be separated into distinct propositions.

- Original Text: "The pericardium, also called pericardial sac, is a double-walled sac containing the heart and the roots of the great vessels."
- Incorrect Proposition: "The pericardium is a double-walled sac containing the heart and the roots of the great vessels."
- Problem: Contains two independent facts that should be split: (1) structural description, (2) contents.
- Correct Propositions: "The pericardium is a double-walled sac." "The pericardium contains the heart." "The pericardium contains the roots of the great vessels."
- Downstream Impact: (1) Retrieval Stage: Queries targeting specific facts (e.g., "What is the structure of the pericardium?") retrieve redundant information. (2) Generation Stage: Increased reasoning burden as LLMs must extract relevant facts from compound statements.

Type C: Factually Unfaithful (Rare). This critical error occurs when generated propositions contain information that contradicts or deviates from the source text.

- Original Text: "Indium is a chemical element with the symbol In and atomic number 49."
- Incorrect Proposition: "Indium has an atomic number of 50."
- Problem: Numerical error introducing factual inconsistency.
- Downstream Impact: (1) Retrieval Stage: Proposition may still be retrieved due to semantic similarity despite factual error. (2) Generation Stage: Directly causes incorrect answers, representing harmful noise injection that undermines RAG reliability.

9.8 Feasibility Analysis

(a) Efficiency Analysis

Using LLaMA2-7B as an example, the total training time is about two hours on a single A100 GPU, with training consisting of two independent components: Refined Prompt Generation and RL-based Prompt Ordering. All experiments (3 datasets × 3 LLM backbones = 9 main experiments, each repeated three times) were run on a GPU with mixed-precision training (FP16) and gradient checkpointing.

Regarding inference efficiency, taking Llama-2-7b deployed locally on an A100 as an example, with FP16 precision, the comparison among Direct, Naive RAG, and FG-RAG is shown in the table 6. In the table, "GPU Mem." represents GPU memory usage, "Prep. Time" represents prompt preparation time, and "Total Time" represents the end-to-end latency from input query to answer generation.

Table 6 Efficiency Comparison of Different Methods

Method	GPU Mem.	Prep. Time	Total Time
Direct	∼15-16GB	–	∼0.5-1.5s
Naive RAG	∼17-18GB	∼0.2-0.4s	∼0.5-1.6s
FG-RAG	∼23-25GB	∼1.8-2.8s	∼2.3-4.3s

(b) Robustness Analysis

To evaluate FG-RAG's resilience to degraded retrieval quality, we systematically inject irrelevant passages into the retrieved document set. Specifically, for each query, we retrieve top-10 documents using Contriever, then randomly replace a proportion $\rho \in \{0\%, 10\%, 30\%, 50\%, 70\%\}$ of these documents with unrelated passages sampled from Wikipedia. The corrupted document set is fed into the full FG-RAG pipeline. We repeat each noise level three times and report average Exact Match (EM) and F1 scores on HotpotQA.

The experiment results are shown in table 7, FG-RAG demonstrates strong robustness to retrieval noise, maintaining high performance even with substantial irrelevant passages. At 30% noise, FG-RAG retains 92.0% of its original EM (27.4 → 25.2) and 91.6% of its original F1 (35.6 → 32.6). At the extreme noise level of 70%, FG-RAG still achieves an EM of 19.5 (71.2% retention) and an F1 of 26.4 (74.2% retention). The performance degradation is gradual and moderate, with the largest absolute drop occurring between 50% and 70% noise (EM: 22.9 → 19.5, F1: 30.1 → 26.4). These results confirm that the DDM-based graph refinement and diffusion-denoising process effectively filter out irrelevant information, preventing noise from propagating through the semantic graph and into the final generation.

Table 7 Retrieval noise robustness results of FG-RAG on HotpotQA. Values in parentheses indicate absolute drop from $\rho = 0\%$.

Noise Proportion ρ	FG-RAG (Ours)	
	EM	F1
0%	27.4	35.6
10%	26.8 (-0.6)	34.9 (-0.7)
30%	25.2 (-2.2)	32.6 (-3.0)
50%	22.9 (-4.5)	30.1 (-5.5)
70%	19.5 (-7.9)	26.4 (-9.2)

9.9 Ablation on Retrieved Subgraph Number m

Experimental Setup. To investigate the impact of the number of retrieved subgraphs m on FG-RAG, we conduct ablation experiments on the HotpotQA dataset using LLaMA-2-7B as the backbone LLM. Given a proposition graph with x nodes, we retrieve the top-m most relevant ego-graphs from the x candidates. We test six configurations: $m = x$ (all subgraphs), $m = \lfloor x/2 \rfloor$, $m = \lfloor x/3 \rfloor$, $m = \lfloor x/4 \rfloor$, $m = \lfloor x/5 \rfloor$, and $m = \lfloor x/6 \rfloor$, where $\lfloor \cdot \rfloor$ denotes floor rounding. Other hyperparameters are fixed at their optimal values: graph token number $N = 15$ and reward balance factor $\alpha = 0.1$. Figure 6 presents the EM results as a bar chart with error bars indicating standard deviation.

Just Accepted

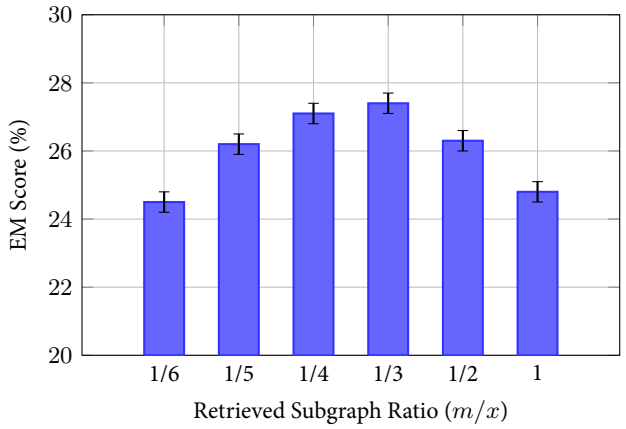

Fig. 6 EM scores on HotpotQA under different retrieved subgraph ratios ($m = \lfloor x \cdot r \rfloor$ where r is the ratio). Error bars represent standard deviation across three runs.

Experimental Conclusion. As shown in Figure 6, the number of retrieved subgraphs m affects FG-RAG performance on HotpotQA, exhibiting a clear inverted-U shaped distribution. Optimal performance is achieved at $m = \lfloor x/3 \rfloor$ (retrieving approximately 33% of subgraphs), reaching 27.4% EM. When m is too small ($m = \lfloor x/6 \rfloor$), performance drops by approximately 3% compared to the optimum, indicating insufficient coverage of critical semantic information. Conversely, when m is too large ($m = x$), performance also degrades, with EM decreasing by 2.6%, confirming that excessive retrieval introduces substantial noise that dilutes relevant information and distracts the LLM's attention mechanism. These findings demonstrate that the number of retrieved subgraphs should balance semantic coverage completeness against information noise control, with $m = \lfloor x/3 \rfloor$ achieving the optimal trade-off.

9.10 Graph construction scalability

(a) Time Complexity of w-hop Ego-Graph Construction

Problem Setting

For a proposition graph $G = (V, E)$ representing semantic relationships between extracted propositions, where $|V| = n$ is the number of nodes (propositions), $|E| = m$ is the number of edges (semantic relationships between propositions), w is the number of hops for ego-graph construction, $\bar{d} = 2m/n$ is the average node degree, H is the hidden dimension of the Pre-trained Language Model (PLM), and L is the average sequence length of proposition text.

Ego-Graph Construction

For each node $v \in V$, its w-hop ego-graph is constructed by performing BFS/DFS traversal within w hops. The time complexity per node is $O(\bar{d}^w)$ in the worst case, as the traversal explores approximately $\bar{d}^w$ nodes. Therefore, constructing all ego-graphs requires:

$$T_{\text{traversal}} = \sum_{v \in V} O(\bar{d}^w) = O(n \cdot \bar{d}^w)$$

Node Feature Extraction

After obtaining the subgraph structure, this work extract node features using a PLM. For each node in the ego-graph, the PLM processes its proposition text of length L, resulting in $O(L \cdot H)$ operations per node. With $\bar{d}^w$ nodes per ego-graph and n ego-graphs, the total feature extraction cost is:

$$T_{\text{features}} = n \cdot \bar{d}^w \cdot O(L \cdot H) = O(n \cdot \bar{d}^w \cdot L \cdot H)$$

Graph Embedding Generation

The node features within each ego-graph are aggregated via mean pooling to produce a graph-level embedding. This operation has complexity $O(\bar{d}^w \cdot H)$ per ego-graph. For all n ego-graphs:

$$T_{\text{pooling}} = n \cdot O(\bar{d}^w \cdot H) = O(n \cdot \bar{d}^w \cdot H)$$

Total Time Complexity

Combining all components, the overall time complexity for constructing and indexing all w-hop ego-graphs is:

$$\begin{aligned} T_{\text{total}} &= T_{\text{traversal}} + T_{\text{features}} + T_{\text{pooling}} \\ &= O(n \cdot \bar{d}^w) + O(n \cdot \bar{d}^w \cdot L \cdot H) + O(n \cdot \bar{d}^w \cdot H) \\ &= O(n \cdot \bar{d}^w \cdot (1 + L \cdot H + H)) \\ &= O(n \cdot \bar{d}^w \cdot L \cdot H) \quad \text{(dominant term)} \end{aligned}$$

(b) Memory Requirements for Dynamic Proposition Graph

FG-RAG employs query-time dynamic graph construction, processing only retrieved documents rather than the full corpus. For a given query, the system first retrieves top-k documents, then extracts propositions exclusively from these k documents to build a local proposition graph on-the-fly.

The memory requirement can be analyzed using the following parameters: k (number of retrieved documents), p_{avg} (average number of propositions per document), $n = k \cdot p_{\text{avg}}$ (total propositions in the local graph), d (average node degree), w (hop count for subgraph retrieval), H (proposition embedding dimension), and L (average proposition text length in bytes).

As shown in Table 8, for a typical configuration with $k = 10$, the total memory per query is approximately 1.75 MB, comprising node storage (62 KB), edges (16 KB), proposition embeddings (768 KB), subgraph index (896 KB), and query overhead (10 KB). Table 9 illustrates how memory scales with different k values: from ∼0.9 MB for $k = 5$ to ∼17.5 MB for $k = 100$, demonstrating linear growth with the number of retrieved documents.

Table 8 Memory Requirements for Dynamic Proposition Graph (with $p_{\text{avg}} = 25$, $d = 8$, $w = 2$, $H = 768$, $L = 200$, $k = 10$)

Component	Formula	Memory
Nodes (text + metadata)	$n \times (L + 48)$	∼62 KB
Edges	$8 \cdot n \cdot d$	16 KB
Proposition embeddings	$n \times H \times 4$	∼768 KB
Subgraph index	$n \times (d^w \times 8 + H \times 4)$	∼896 KB
Query overhead	–	∼10 KB
Total per query		∼1.75 MB

Table 9 Memory Scaling with Different Retrieval Document Counts ($p_{\text{avg}} = 25$)

k	$n = k \cdot p_{\text{avg}}$	Memory
5	125	∼0.9 MB
10	250	∼1.8 MB
20	500	∼3.5 MB
50	1,250	∼8.8 MB
100	2,500	∼17.5 MB

Just Accepted

The key advantages of this dynamic design are twofold. First, memory consumption is independent of corpus size, depending only on k and p_{avg}. Moreover, this dynamic construction enables practical deployment on Wikipedia-scale corpora with minimal memory footprint, validating the framework's real-world applicability.

9.11 LLM attention weight analysis

Although FG-RAG uses soft prompts, this part conducts attention analysis with hard prompts for interpretability. Soft prompts lack token-level semantics, while hard prompts provide clear insights into how the model attends to different semantic positions. We use the hard prompt "The sun is hot and gives us light every day" as input to LLaMA-2-7B and extract attention weights from the last layer, head 0. Two complementary visualizations are presented in Figure 7 (bar chart) and Figure 8 (heatmap)

As shown in Figure 7, semantically significant tokens such as "sun" receive the highest attention ratio, while function words such as "the" are largely ignored. Figure 8 further reveals that these salient tokens exhibit consistently strong attention across multiple generation steps, appearing as prominent vertical stripes. These findings validate the motivation behind FG-RAG's RL-based prompt ordering: by placing the most informative propositions at positions that naturally attract higher attention, the framework better aligns the input with the model's inherent processing biases, ensuring optimal utilization of retrieved knowledge.

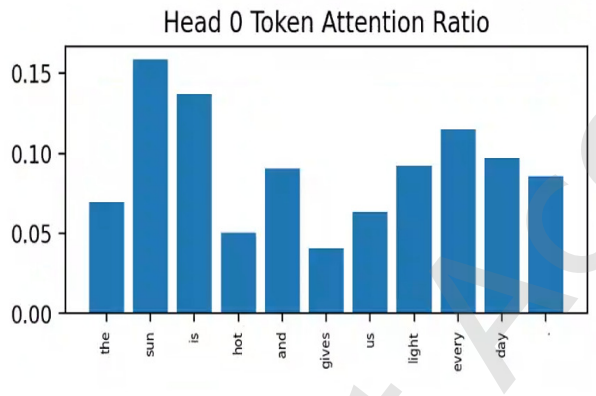

Fig. 7 Token-wise attention ratio distribution on LLaMA-2-7B (last layer, head 0).

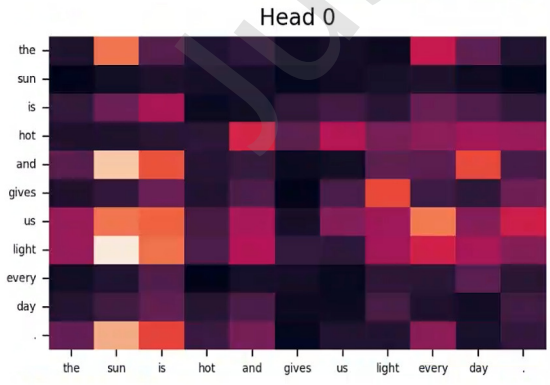

Fig. 8 Attention heatmap visualization for LLaMA-2-7B (last layer, head 0).

9.12 Statistical significance tests

We provide an analysis of partial experimental results based on the three-run replication data mentioned in Section 5. Taking FG-RAG on HotpotQA with Mistral-7B as an example, The data are shown in table 10.

Table 10 Three-run experimental results on HotpotQA (Mistral-7B)

Method	Run 1	Run 2	Run 3	Mean (EM%)	SD
AdaComp	27.5	27.7	27.9	27.7	0.2
FG-RAG	28.7	29.1	29.5	29.1	0.4

(a) p-values and confidence intervals

Based on the three runs above, we conducted an independent two-sample t-test (two-tailed). The t-statistic is calculated as:

$$t = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}} = \frac{29.1 - 27.7}{\sqrt{\frac{0.4^2}{3} + \frac{0.2^2}{3}}} = 5.43$$

Using the Welch–Satterthwaite equation, the degrees of freedom is approximately $df \approx 3$. The resulting p-value is $p = 0.032$. The 95% confidence interval for the mean improvement is:

$$95\%\ \text{CI} = (\bar{x}_1 - \bar{x}_2) \pm t_{0.025,2} \times \text{SE} = [0.29\%, 2.51\%]$$

(b) Statistically significant improvements

The p-value ($p = 0.032$) is below the conventional significance threshold of $p < 0.05$. Therefore, the +1.4% improvement achieved by FG-RAG on HotpotQA with Mistral-7B is statistically significant. The 95% confidence interval $[0.29\%, 2.51\%]$ does not contain zero, further confirming the robustness of this improvement.

Based on the consistently low standard deviations observed across all experiments in Table 1, we can reasonably infer that most of the reported improvements in Table 1 are also statistically significant.

(c) Variance across runs

The standard deviations across three independent runs are low ($\sigma = 0.4\%$ for FG-RAG and $\sigma = 0.2\%$ for AdaComp), indicating high reproducibility for both methods regardless of random seed variations. FG-RAG's slightly higher variance (0.4% vs. 0.2%) is expected due to its more complex architecture involving graph construction and RL-based optimization, yet remains within an acceptable range. The strong signal-to-noise ratio, with the improvement magnitude (1.4%) being 3.5 times FG-RAG's standard deviation and 7 times AdaComp's standard deviation, demonstrates that the observed gain far exceeds experimental noise. Furthermore, the complete absence of overlap between the performance ranges (AdaComp's max 27.9% vs. FG-RAG's min 28.7%) confirms that the performance ranking between methods is stable and not an artifact of random fluctuations. This low variance pattern holds consistently across our experiments, which we attribute to the deterministic nature of our DDM-based graph encoding and the well-optimized RL training process that converges to consistent solutions despite the inherent stochasticity of reinforcement learning.

Just Accepted