GEAR: Graph-Efficient Augmented Retrieval via Adaptive Knowledge-Path Fusion

Yongkang Zhou*,Xingrun Quan†, Yuxuan Hou†,Junjie Yao†‡, Gang Xu†‡, and Jiangtao Wang†‡
* School of Computer Science and Technology, East China Normal University.
† Software Engineering Institute, East China Normal University.
‡ Shanghai Key Laboratory of Trustworthy Computing
gabbyzhou0512@gmail.com, junjie.yao@sei.ecnu.edu.cn

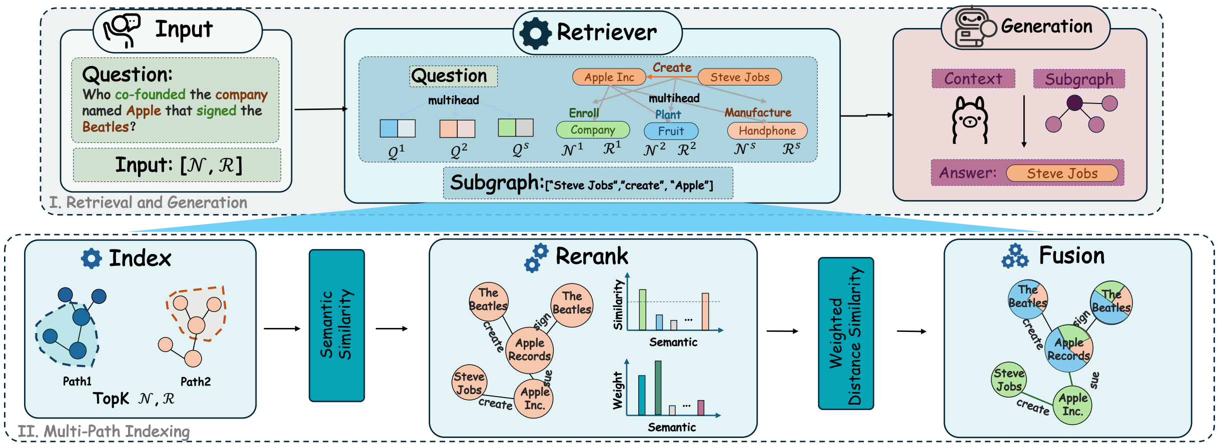

Fig. 1: Adaptive Knowledge-Path Fusion in Graph Retrieval Pipeline

***Abstract*—Retrieval-Augmented Generation (RAG) seeks to mitigate hallucinations in Large Language Models (LLMs) by grounding responses in external knowledge. Graph structures serve as a particularly potent knowledge source, offering rich semantic and structural context. However, the efficacy of existing graph-based RAG systems is often compromised by imperfect and inefficient retrievals. These drawbacks stem from two key challenges: the semantic gap, where textual similarity metrics fail to capture multifaceted query aspects, and inter-module misalignment, where independently optimized components disrupt the coherence of the retrieval pipeline.**

To address these limitations, we introduce Graph-Efficient Augmented Retrieval (GEAR), a novel framework for graph structure indexing and fusion. GEAR utilizes a multi-head architecture to interpret queries across diverse semantic spaces, facilitating a more nuanced retrieval process. It then performs fine-grained fusion and re-ranking of retrieved paths to construct subgraphs precisely aligned with user intent. This jointly optimized design enables the retriever to effectively leverage both nodal text attributes and subgraph topology, significantly enhancing generation quality and efficiency. The implementation and datasets are available: https://github.com/gabbyzyk/GEAR .

Extensive experiments on multiple benchmark datasets reveal that GEAR substantially outperforms state-of-the-art baselines in both accuracy and reliability. Further ablation studies validate the critical contributions of each component in our design, confirming its effectiveness in achieving robust and efficient graph-based retrieval.

***Index Terms*—Retrieval Augmented Generation, Large Language Models, Graph Representation, Path Indexing.**

I. Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities in natural language understanding and generation, yet they are constrained by significant limitations, including a propensity for factual hallucinations and a reliance on static, outdated internal knowledge [1]. These shortcomings hinder their deployment in knowledge-intensive, real-world applications. Retrieval-Augmented Generation(RAG) has emerged as a compelling paradigm to address these issues by grounding LLM responses in external, verifiable, and up-to-date knowledge sources [2]. Within this domain, there is a growing momentum toward leveraging structured knowledge, particularly knowledge graphs, to enrich the retrieval process [3]. Among these, graph structure are especially powerful, as they synergize structural relations with rich textual descriptions, providing a semantically expressive context ideal for complex reasoning tasks. By enabling multi-hop inference across the graph topology and integrating descriptive node and edge attributes, graph offers a contextual depth that far surpasses linear text corpora.

Recent efforts have sought to overcome these limitations through various architectural innovations, yet these often introduce their own trade-offs between expressive power and computational cost. Some approaches employ Graph Neural Networks (GNNs) to encode structural semantics [4] or explore explicit reasoning paths [3], [5], [6]. While powerful, these methods can be computationally expensive, introducing significant retrieval latency due to complex message-passing or a combinatorial explosion in path exploration. Other strategies focus on linearizing graph structures into text that LLMs can directly process [7] or constructing hypergraphs to capture higher-order dependencies [8]. However, linearization can lead to excessively long input sequences, increasing token costs and losing vital structural priors, while hypergraph construction adds another layer of computational overhead.

Despite these advancements, the integration of graph structures into RAG frameworks introduces its own set of formidable challenges. A primary difficulty lies in effectively navigating the complexity of graph data to extract precisely relevant subgraphs. Existing graph-RAG pipelines often consist of modular components—such as a retriever, a re-ranker, and a generator—that are developed and optimized in isolation. This fragmented design leads to critical inefficiencies and semantic misalignments, including representation mismatches and suboptimal inter-module handoffs. For instance, a retriever might identify relevant entities, but the subsequent subgraph construction module may fail to capture the precise relational context needed to answer a query, ultimately undermining the quality of the generated response and leading to hallucinations or incomplete answers. This issue highlights a critical *semantic gap* and *query efficiency* between the user's multi-faceted intent and the system's uni-dimensional understanding, a problem that remains even if efficiency concerns are set aside.

To address these challenges, we introduce **GEAR (Graph-Efficient Augmented Retrieval)**, a novel, end-to-end framework that re-engineers the graph-RAG pipeline around a multi-head architecture. GEAR's central premise is to deconstruct complex queries into their constituent semantic facets and pursue them in parallel across the knowledge graph. As illustrated in Figure 1, our approach models both the query and the graph elements across multiple, independent semantic spaces. Instead of collapsing all information into a single, dense vector, GEAR employs the distinct attention heads of a Transformer-based encoder to project the query and graph nodes into several specialized representations. Each head learns to capture a unique aspect; for instance, given a query involving "Michael Jordan" one head might focus on the "computer scientist" context while another captures the "basketball player" context. This multi-perspective encoding allows for fine-grained retrieval tailored to the specific nuances of the user's intent, effectively bridging the *semantic gap* by disentangling polysemous entities at the representational level before retrieval even begins.

A core innovation of GEAR is its unified design, which resolves the *inter-module misalignment* that plagues conventional pipelines by ensuring semantic coherence from retrieval to generation. The multi-head representation is not a one-off feature of the retrieval stage; it serves as a persistent "semantic signature" that guides the entire workflow. This process begins with *multi-space retrieval*, where each head retrieves a distinct set of candidate nodes and paths corresponding to its specialized semantic view. Subsequently, in the *adaptive fusion* stage, GEAR dynamically weighs the importance of each retrieved path based on its relevance to the overall query, intelligently merging the parallel streams of evidence. This fused representation then directs a *structure-aware subgraph construction* module, ensuring that the final graph context is not just a loose collection of relevant nodes but a connected and coherent explanation. By maintaining the query's nuanced, multifaceted interpretation from start to finish, GEAR constructs a subgraph that is both structurally sound and semantically aligned with the user's precise intent.

As depicted in Figure 2, this "fusion-path" strategy overcomes the pitfalls of single-path methods, which often retrieve locally optimal but globally incomplete evidence. For instance, a single-path approach might identify a strong link between "Steve Jobs" and "Apple Inc." but fail to incorporate the context about "The Beatles" and "Apple Records". GEAR's fusion mechanism, in contrast, synthesizes these disparate but essential pieces of evidence into a single, comprehensive subgraph. This holistic context generation leads directly to superior end-to-end performance and greater efficiency by focusing computational resources on promising intersections of evidence rather than exhaustively exploring single, isolated paths.

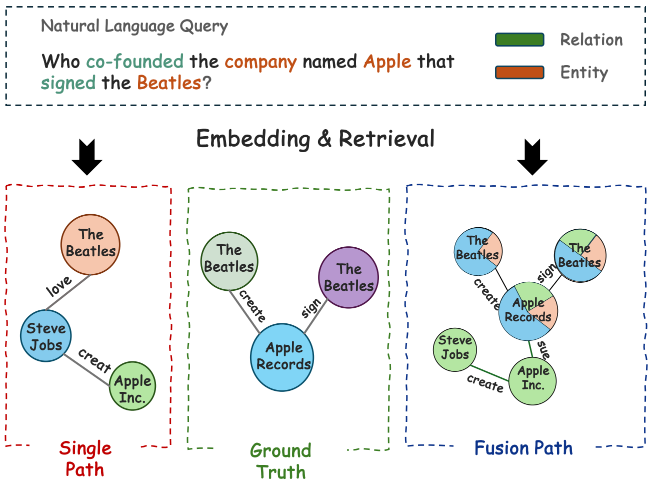

Fig. 2: Adaptive Knowledge-Path Fusion.

In summary, our main contributions are as follows:

- We identify and analyze two fundamental, interrelated challenges in graph-based RAG: the *semantic gap* caused by single-space semantic modeling and the *inter-module misalignment* stemming from fragmented pipeline design, which together impair performance on complex, multi-aspect queries.
- We propose GEAR, a novel end-to-end framework featuring a multi-head architecture that facilitates adaptive knowledge-path fusion. This unified design spans retrieval, re-ranking, and subgraph construction, ensuring semantic fidelity across the entire RAG pipeline.
- We conduct comprehensive experiments and ablation studies on multiple benchmark datasets, demonstrating

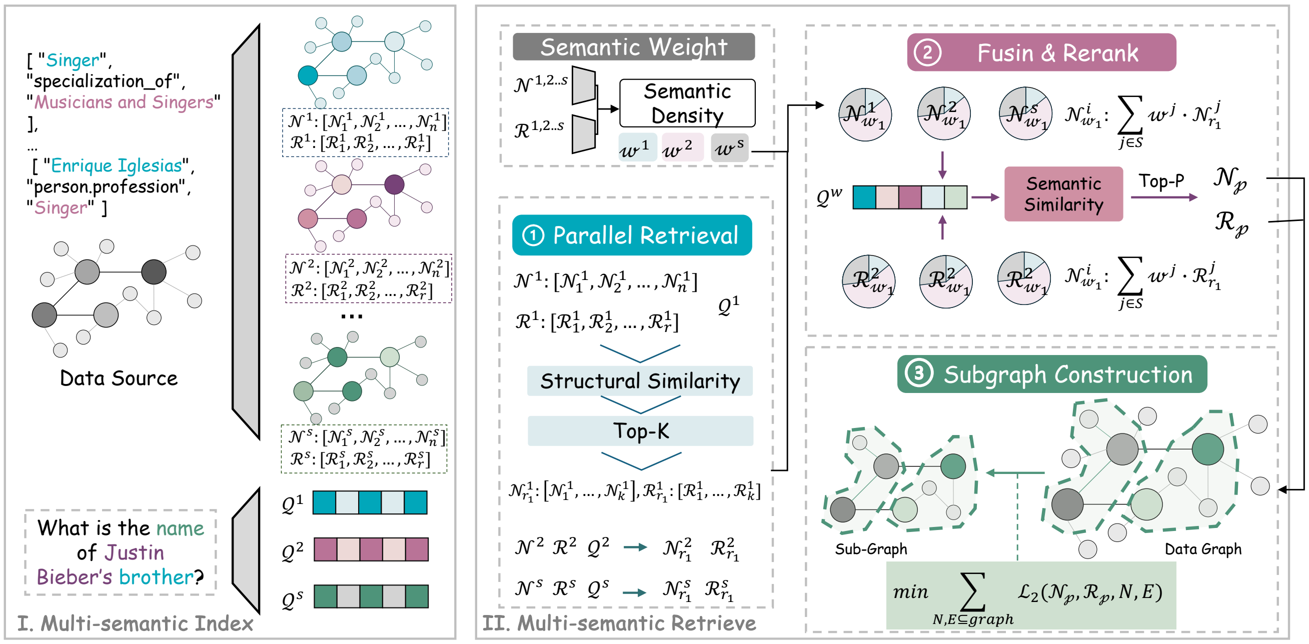

Fig. 3: Framework of GEAR

that GEAR's path fusion approach provides a more robust, efficient, and accurate solution for semantically rich retrieval in complex graph-based RAG settings.

II. Preliminary

We consider the problem of answering natural language queries using structured knowledge encoded in a *text-attributed graph* (TAG), denoted as $G = (V, E)$. Each node $v \in V$ and each edge $e \in E \subseteq V \times V$ is associated with a natural language description, such as entity names, attribute values, or relational phrases. This representation allows graph reasoning to be conducted entirely in natural language space, enabling seamless integration with pretrained language models.

a) Task Definition.: Formally, let $G = (V, E)$ be a text-attributed graph. We define H distinct retrieval paths $\{P_1, \ldots, P_H\}$, where each path P_i provides a unique semantic or structural lens to interpret and retrieve nodes and edges. These paths may correspond to relation types, logical chains, or embedding subspaces. Our goal is to construct a query-specific subgraph $G_{\text{sub}} = (V_{\text{sub}}, E_{\text{sub}})$ that satisfies three key properties: it should exhibit **path relevance**, meaning the subgraph includes nodes and edges that are meaningfully related to the query based on multi-view semantic matching; provide **structural support**, by capturing graph paths and neighborhoods that enable multi-hop reasoning or contextual understanding; and ensure **compactness**, maintaining a bounded size that facilitates efficient processing by large language models (LLMs).

The constructed G_{sub} is fed into an LLM alongside the query q to produce an answer. Unlike schema-driven knowledge retrieval or symbolic querying over graphs, our framework requires no predefined ontology or formal query language; instead, the entire pipeline operates over text-annotated graph structures and unstructured natural language input/output.

b) Problem Setting.: We adopt a retrieval-augmented strategy for subgraph construction, which comprises the following core stages:

1) **Path-Based Indexing:** Each graph element (node or edge) and the query are encoded under multiple latent structural paths, capturing diverse relational perspectives across the graph.
2) **Multi-Path Retrieval:** For each path, the system retrieves the top-k relevant elements (nodes or edges), forming multiple parallel candidate sets.
3) **Cross-Path Fusion:** A re-ranking module aggregates relevance scores across all paths using a weighted voting mechanism, yielding a unified candidate list.
4) **Subgraph Construction:** The final subgraph G_{sub} is assembled using a structure-aware construction module over candidates of high confidence.

Our objective is to construct G_{sub} in a way that maximizes the probability that a pre-trained LLM produces the correct answer:

$$\arg\max_{G_{\text{sub}} \subseteq G} \text{LLM}(q, G_{\text{sub}}) \tag{1}$$

This formulation allows our system to leverage both semantic representations and structural signals during subgraph construction, enabling generalizable reasoning over natural language graphs.

III. Approach

In this section, we detail the components of GEAR. Given a query and a large text-attributed graph, our system first

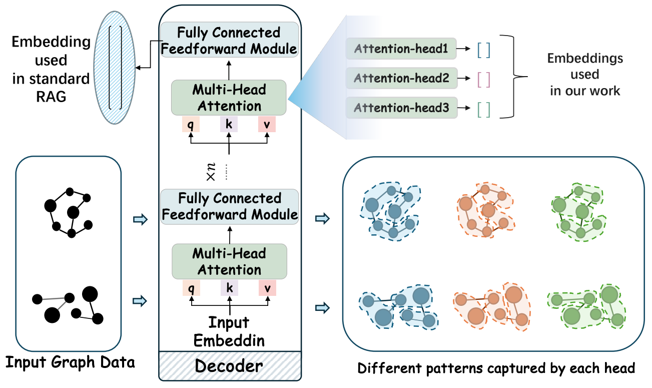
Embedding used in standard RAG
Fully Connected Feedforward Module
Multi-Head Attention
q
k
v
Attention-head1
Attention-head2
Attention-head3
Embeddings used in our work
Fully Connected Feedforward Module
Multi-Head Attention
Input Embeddin
Decoder
Input Graph Data
Different patterns captured by each head

Fig. 4: Multi-Path Index

constructs multiple path-aware vector indices to support diverse relational retrievals. It then retrieves candidate elements through multi-path search, re-ranks them based on path importance, and assembles a compact subgraph for downstream generation.

We begin by introducing how the input graph is indexed under multiple structural perspectives to enable efficient path-based retrieval.

A. Path-based Indexing

To enable path retrieval from diverse relational perspectives, we construct a set of parallel indexing structures, each corresponding to a distinct structural projection of the graph. As illustrated in Figure 4, we leverage the final layer attention heads of a Transformer decoder to define H latent paths $P_1, P_2, \dots, P_H$ over graph elements.

For each path P_i, the query q is encoded into a vector representation $\mathbf{q}^{(i)} \in \mathbb{R}^d$, while each node or edge $x \in V \cup E$ is encoded into $\mathbf{x}^{(i)} \in \mathbb{R}^d$ using the same attention head h_i. This yields H independent embedding paths $\{\mathbf{x}^{(i)}\}_{i=1}^{H}$, where each path captures a different structural intuition—e.g., entity-centric, path-centric, or context-centric relevance.The detailed procedure is formally summarized in Algorithm 1.

These multiple embedding path offer threefold advantages that are critical for structured retrieval. First, we introduce path-level diversity: each embedding head specializes in capturing distinct textual cues and relational semantics, enabling the system to model heterogeneous path views without reliance on predefined schema or manual supervision. Second, the decoupling across embedding path allows for parallel retrieval, where candidate selection can be performed independently in each path, yielding significant scalability gains. Third, by projecting both queries and graph elements into multiple latent views, GEAR supports fine-grained matching, capturing subtle alignment signals that may be missed under a single representation—especially in cases of ambiguous or context-sensitive relevance.

We build H vector indices using the above embeddings, each supporting efficient similarity search via approximate nearest neighbor methods. These indices form the foundation for the multi-path retrieval process described in the next section.

Algorithm 1 Path-based Indexing

```
1: Input: Query $q$; Graph $G = (V, E)$ with node texts $T_V$
   and edge texts $T_E$; number of semantic paths $H$
2: Output: Multi-Path index $\mathcal{I} = \{G^{(1)}, G^{(2)}, \dots, G^{(H)}\}$
3: for $h = 1$ to $H$ do
4:     $\mathbf{q}^{(h)} \leftarrow Encoder_h(q)$
5:     $\mathbf{T}_V^{(h)} \leftarrow Encoder_h(T_V)$
6:     $\mathbf{T}_E^{(h)} \leftarrow Encoder_h(T_E)$
7:     Construct graph view $G^{(h)}$ based on encoded path
8: end for
9: return $\mathcal{I}$
```

B. Fusion Retriever

We design the Fusion Retriever as a two-stage framework to balance retrieval diversity and candidate precision. In the first stage, we perform path-specific retrieval under multiple relational views, capturing semantically complementary evidence. In the second stage, we re-rank the retrieved elements across paths by assessing the informativeness of each path and the quality of individual candidates. We elaborate on both stages below, as illustrated in Figure 5.

Multi-Path Retrieval. Given the multi-path vector index constructed in the previous stage, we now retrieve graph elements relevant to the input query from multiple structural perspectives. This retrieval is performed independently across H latent path spaces, each capturing distinct relational signals embedded in the graph.

Formally, for each path P_i ($i = 1, \dots, H$), we compute the top-k most similar nodes and edges to the query representation $\mathbf{q}^{(i)}$ via a nearest neighbor search over the corresponding vector space:

$$L_i = \underset{x \in V \cup E}{\mathrm{TopK}} \ \mathrm{sim}\big(\mathbf{q}^{(i)}, \mathbf{x}^{(i)}\big) \tag{2}$$

This results in H candidate lists $\{L_1, L_2, \dots, L_H\}$, where each L_i contains the highest-scoring candidates under path P_i. Intuitively, each list offers a complementary perspective on which parts of the graph are relevant to the query, depending on the structural paths captured by P_i.

This stage ensures that our system can recall a diverse set of potentially useful candidates, even when they are salient only under specific relational views. However, not all paths contribute equally to a given query; thus, we proceed to selectively re-rank and merge these results based on their structural importance.

Cross-Path Re-Ranking. To consolidate the diverse retrieval results into a unified ranking, we introduce a cross-path re-ranking stage that adaptively weights candidates based on the informativeness of their associated paths.

We first estimate the importance of each path P_i through a scalar score s_i, computed as:

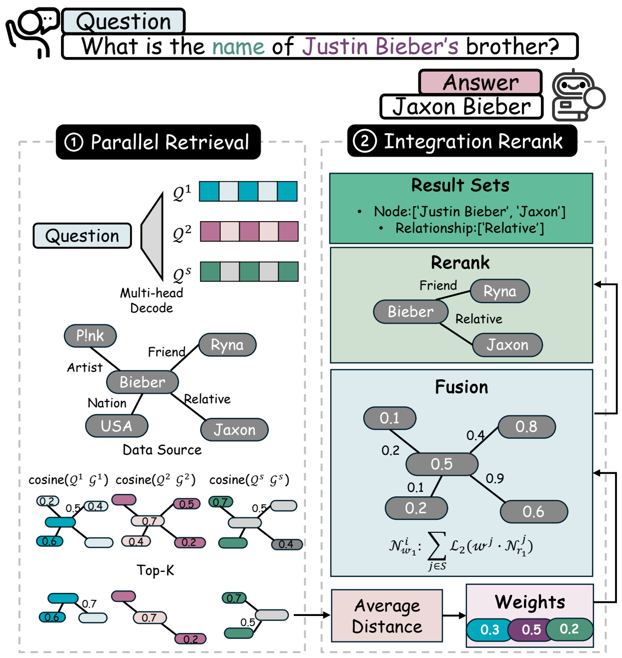

Fig. 5: Fusion Retriever

Algorithm 2 Fusion Retriever

```
1:  Input: Encoded query $\{\mathbf{q}^{(h)}\}_{h=1}^{H}$; graph representations $\{\mathbf{T}_V^{(h)}, \mathbf{T}_E^{(h)}\}_{h=1}^{H}$; retrieval size $k$
2:  Output: Candidate nodes $\hat{V}$ and edges $\hat{E}$

    I. Multi-Path Retrieval
3:  for $h = 1$ to $H$ do
4:      $TopK\_V^{(h)} \leftarrow Sim(\mathbf{q}^{(h)}, \mathbf{T}_V^{(h)}, k)$
5:      $TopK\_E^{(h)} \leftarrow Sim(\mathbf{q}^{(h)}, \mathbf{T}_E^{(h)}, k)$
6:  end for

    II. Cross-Path Re-Ranking
7:  for $h = 1$ to $H$ do
8:      Compute importance weight: $\alpha^{(h)} = a^{(h)} \cdot b^{(h)}$
9:  end for
10: $\mathcal{S}_V \leftarrow WeightedRerank(\{TopK\_V^{(h)}\}, \{\alpha^{(h)}\})$
11: $\mathcal{S}_E \leftarrow WeightedRerank(\{TopK\_E^{(h)}\}, \{\alpha^{(h)}\})$
12: $\hat{V} \leftarrow TopK(\mathcal{S}_V, k)$
13: $\hat{E} \leftarrow TopK(\mathcal{S}_E, k)$
14: return $\hat{V}, \hat{E}$
```

$$s_i = a_i \cdot b_i \tag{3}$$

Here, a_i is the average L2 norm of all retrieved embeddings $\{\mathbf{x}^{(i)}\}$ in L_i, which reflects the overall attention strength in that path space. b_i is the average pairwise cosine distance among these embeddings, capturing path dispersion. Together, s_i quantifies how informative and structurally diverse the retrieved results are under path P_i.

We then assign a path-weighted score to each candidate $x \in L_i$ based on both its position and the path's importance:

$$w(x) = s_i \cdot 2^{-p} \tag{4}$$

where p is the rank position of x in L_i. This formulation prioritizes high-quality results from high-importance paths, while exponentially penalizing lower-ranked entries.

Finally, we merge all lists $\{L_1, \dots, L_H\}$ and select the top-k candidates by their re-weighted scores $w(x)$ to form the initial seed set $\hat{V}, \hat{E}$ for subgraph construction.

This two-stage strategy: (1) path-specific retrieval followed by (2) cross-path re-ranking — allows us to balance between diversity and precision. The first stage ensures broad coverage across multiple relational views, enabling recall of heterogeneous but potentially relevant candidates. The second stage filters and prioritizes the most informative signals, ensuring that only structurally salient and semantically coherent candidates are preserved. This approach avoids early pruning errors and provides strong seeds for subgraph assembly in the next stage.

C. Subgraph Construction via Path-Weighted Expansion

After multi-path retrieval and cross-path re-ranking, we obtain a candidate set of nodes $\hat{V}$ and edges $\hat{E}$ that are highly relevant to the query under diverse relational paths. The goal of this module is to assemble a connected and expressive subgraph from these ranked candidates, forming the structural context for downstream generation.

We propose a default subgraph construction strategy named **Fusion-Union**, which unifies semantically relevant signals from multiple paths into a structurally coherent context. Unlike purely retrieval-based methods that may yield sparse or fragmented results, Fusion-Union explicitly ensures semantic alignment, path diversity, and structural connectivity.

While the top-k nodes and edges identified from re-ranking already reflect query salience under different paths, they may not form a well-connected or semantically complete subgraph. Some relevant intermediate nodes may not receive high similarity scores individually but are necessary to bridge disparate high-scoring nodes. Our method compensates for this by fusing local structure with global path importance.

Let $\hat{G} = (\hat{V}, \hat{E})$ denote the initial seed graph. We expand this graph into $\tilde{G}$ using a neighborhood expansion strategy that integrates three complementary criteria: (1) Relational adjacency, which requires candidate nodes to be direct neighbors of existing nodes in $\hat{G}$ to preserve structural continuity; (2) path alignment, which restricts expansion to neighbors exhibiting high cosine similarity to the query embedding $\mathbf{q}^{(i)}$ in each path-specific path $\mathbf{x}^{(i)}$; and (3) path selectivity, which assigns higher expansion priority to candidates associated with paths of larger normalized importance weights $\{\alpha^{(i)}\}_{i=1}^{H}$.

Algorithm 3 Subgraph Construction and Answer Generation

```
1: Input: Query $q$; Candidate nodes and edges $(\hat{V}, \hat{E})$; Fusion weights $\{\alpha^{(h)}\}_{h=1}^{H}$
2: Output: Predicted answer $\hat{a}$
3: Step 1: Initialize subgraph
4: Initialize seed subgraph $\hat{G} \leftarrow (\hat{V}, \hat{E})$
5: Step 2: Identify expansion candidates
6: $N \leftarrow$ set of nodes adjacent to $\hat{V}$ in the global graph $G$
7: for each $x \in N$ do
8:     Compute path-aware similarity score:
           score$(x) \leftarrow \sum_{h=1}^{H} \alpha^{(h)} \cdot \text{sim}(\mathbf{q}^{(h)}, \mathbf{x}^{(h)})$
9: end for
10: Select top-$m$ scored neighbors $N^*$ and corresponding edges
11: Step 3: Expand subgraph
12: Update $\tilde{V} \leftarrow \hat{V} \cup N^*$
13: Update $\tilde{E} \leftarrow \hat{E} \cup$ edges connecting $\tilde{V}$
14: Construct expanded subgraph $\tilde{G} \leftarrow (\tilde{V}, \tilde{E})$
15: Step 4: Linearize input for LLM
16: Generate subgraph description Desc$(\tilde{G})$ via BFS traversal
17: Retrieve salient candidates from re-ranking: TopK$(\hat{V}, \hat{E})$
18: Concatenate input sequence:
        $\mathbf{x} \leftarrow$ [BOS], $q$, TopK$(\hat{V}, \hat{E})$, Desc$(\tilde{G})$, [EOS]$_{\text{USER}}$
19: // Step 5: Generate answer with LLM
20: $\hat{a} \leftarrow$ LLM$(\mathbf{x})$
21: return $\hat{a}$
```

Formally, we score each neighbor candidate x using:

$$\text{score}(x) = \sum_{i=1}^{H} \alpha^{(i)} \cdot \text{sim}(\mathbf{q}^{(i)}, \mathbf{x}^{(i)}) \tag{5}$$

where $\mathbf{q}^{(i)}$ is the query representation in path P_i, and $\mathbf{x}^{(i)}$ is the corresponding embedding of node or edge x under the same path.

Top-scoring neighbors are iteratively incorporated into the graph:

$$\tilde{G} = \text{Expand}(\hat{G}; \{\alpha^{(i)}\}_{i=1}^{H}) \tag{6}$$

until a size threshold or convergence criterion is met.

In practice, we first identify all nodes adjacent to $\hat{V}$ and compute their scores using Equation 5. We then retain edges between any pair of nodes selected into the final graph, ensuring that the resulting subgraph is connected. Algorithm 3 outlines this process in detail.

Fusion-Union is designed as a general-purpose method for subgraph construction that maximizes semantic recall and path coherence. However, for different use cases—e.g., high precision, low redundancy, or stronger topological priors—alternative construction paradigms may be preferred. We introduce such variants in Section IV-A, including Personalized PageRank and combinatorial optimization via PCST, to meet these diverse needs.

D. Answer Generation

The final subgraph $\tilde{G}$, produced through multi-path retrieval and guided expansion, provides a structured and context-rich foundation for answer generation. Given a natural language query q and the optimized subgraph $\tilde{G}$, our goal is to generate an answer that is both accurate and grounded in the retrieved graph content.

To accomplish this, we follow a *textual linearization* strategy: we convert the selected subgraph and re-ranked nodes and edges into natural language spans, which are concatenated with the query and fed directly into a pretrained autoregressive language model (e.g., T5[9], LLaMA[10]). This approach enables us to exploit powerful off-the-shelf LLMs without requiring architectural modification or soft prompt tuning.

$$\mathbf{X} = [\text{Query}] \parallel [\text{Candidate Entity}] \parallel [\text{Subgraph}]$$

Firstly, the Query is appended at the end of the sequence, then the Subgraph is generated by breadth-first linearization of $\tilde{G}$, converting each node and edge into a textual span that reflects both its content and partial structural role. Optionally, entity types and relation labels may be added to improve the semantic expressiveness of the input. Following subgraph construction, the Candidate Entity component highlights the most salient elements as identified during the re-ranking stage, serving as focused context to strengthen the alignment between subgraph content and query intent. , followed by an answer placeholder such as [Answer]:, which signals the LLM to begin generation.

We compute the language modeling loss only over the answer span:

$$\mathcal{L}_{\text{LM}} = -\sum_{t \in \text{Answer}} \log P(y_t \mid y_{<t}, \mathbf{X}) \tag{7}$$

This generation design ensures that the model conditions its output on a query-aligned, semantically weighted, and structurally guided subgraph, yielding more accurate and interpretable responses compared to full-graph or retrieval-only baselines.

Importantly, the generation module remains *fully modular*: it operates solely on the textualized output of prior stages, allowing seamless adaptation to various LLM backbones and domains without entangling retrieval-specific components.

IV. Subgraph Construction Variants

A. Fusion Personalized PageRank

While the base subgraph construction strategy in Section III effectively combines semantic relevance with path-based guidance, it may not always align with different application constraints or structural priors. In particular, tasks may vary in their preference for coverage versus compactness, or require integration with external optimization tools. To enhance flexibility and robustness, we propose two complementary subgraph construction variants, shown in Figure 6. Both variants build upon the same ranked retrieval output (top-k nodes and edges), but differ in how they expand or filter these candidates into a final subgraph.

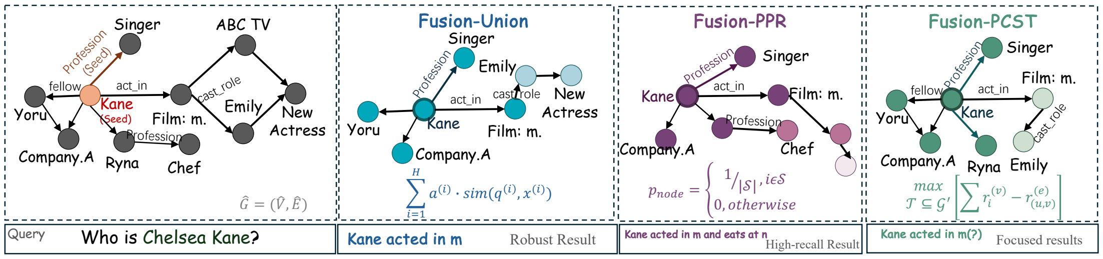

Fig. 6: Different Subgraph Construction Methods

The first variant adopts a diffusion-based strategy that expands from high-scoring entities via personalized propagation. This method favors broader contextual coverage and is suitable for recall-oriented scenarios.

Specifically, we apply the *Generalized Diffusion Convolution* (GDC) operator, configured to approximate personalized PageRank (PPR) over the input graph. Let $\mathcal{V}_{\text{top}}$ and $\mathcal{E}_{\text{top}}$ denote the top-k retrieved nodes and edges. We define the seed node set $\mathcal{S}$ as:

$$\mathcal{S} = \mathcal{V}_{\text{top}} \cup \{u \mid (u,v) \in \mathcal{E}_{\text{top}}\} \cup \{v \mid (u,v) \in \mathcal{E}_{\text{top}}\} \tag{8}$$

We then construct a personalization vector $\boldsymbol{p} \in \mathbb{R}^{|\mathcal{V}|}$ over all nodes, where:

$$p_i = \begin{cases} \frac{1}{|\mathcal{S}|}, & \text{if } i \in \mathcal{S} \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

The GDC operator yields the stationary distribution:

$$\boldsymbol{\pi} = (1-\alpha) \cdot (\mathbf{I} - \alpha\mathbf{P})^{-1} \cdot \boldsymbol{p} \tag{10}$$

where α is the teleport probability and $\mathbf{P}$ is the normalized transition matrix of the graph.

We retain nodes with non-zero entries in $\boldsymbol{\pi}$, and select all edges connecting them. This results in a contextually extended subgraph guided by both semantic relevance and structural proximity.

B. Fusion PCST

The second variant takes a different perspective, framing subgraph construction as a constrained optimization problem[4]. It aims to select a compact subgraph that maximizes reward while minimizing edge cost, making it particularly suitable for precision-focused tasks.

We model this problem via the *Prize-Collecting Steiner Tree* (PCST) framework, where nodes and edges receive rewards based on semantic alignment across multiple heads.

Let the query be encoded as $\boldsymbol{q} \in \mathbb{R}^{h\times d}$ using h semantic heads. The reward scores are computed as:

$$r_i^{(v)} = \sum_{k=1}^{h} \cos(\boldsymbol{q}_k, \boldsymbol{x}_{i,k}) \cdot w_k, \quad \forall i \in \mathcal{V} \tag{11}$$

$$r_j^{(e)} = \sum_{k=1}^{h} \cos(\boldsymbol{q}_k, \boldsymbol{e}_{j,k}) \cdot w_k, \quad \forall j \in \mathcal{E} \tag{12}$$

where $\boldsymbol{x}_{i,k}$ and $\boldsymbol{e}_{j,k}$ are the k-th path embeddings for node i and edge j, respectively, and w_k is the learned importance weight of path k.

We retain top-k nodes and edges by reward, and transform the graph into a PCST instance. Each edge (u,v) with reward greater than a base cost c is split into two edges via an auxiliary node, redistributing excess reward to encourage its inclusion.

Let $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ denote the augmented graph, and $r^{(v)}$, $r^{(e)}$ the reward and cost vectors. We solve:

$$\max_{\mathcal{T}\subseteq\mathcal{G}'} \left[\sum_{i\in\mathcal{V}_\mathcal{T}} r_i^{(v)} - \sum_{(u,v)\in\mathcal{E}_\mathcal{T}} r_{(u,v)}^{(e)} \right] \tag{13}$$

The resulting tree $\mathcal{T}$ is mapped back to the original graph, yielding a semantically coherent and cost-efficient subgraph.

These two variants embody different structural philosophies: the PPR-based approach favors broad diffusion and high recall, while the PCST variant prioritizes sparsity and precision. By integrating both, our framework supports a wide spectrum of applications and retrieval conditions. We evaluate their performance in Section V.

V. EXPERIMENTS

A. Experimental Settings

Running Environment: To comprehensively evaluate the effectiveness and generalizability of our proposed approach, we establish a controlled experimental environment with public datasets, competitive baselines, and rigorous evaluation protocols. All experiments are conducted on a high-performance server equipped with one NVIDIA A100 GPU under Ubuntu 24.04.

To ensure reproducibility and fairness, all random seeds are fixed across runs, and each experiment is repeated five times with the average result reported.

Benchmarks: We evaluate our method on three representative and publicly available graph dataset benchmarks: ExplaGraphs[11], SceneGraphs[12], and WebQSP [13]. These datasets span diverse domains and pose distinct reasoning challenges, making them well-suited for a comprehensive evaluation of graph-augmented LLM reasoning systems.

TABLE I: Linguistic and textual properties of the datasets, analyzed on a per-query basis.

Dataset	Avg. Q. Length	Avg. Constraints	Avg. Tokens
ExplaGraphs[11]	12.1	1.8	30
SceneGraphs[12]	9.7	1.2	1396
WebQSP[13]	8.5	2.1	100,627

TABLE II: Graph structural properties for Each Dataset.

Dataset	Avg. Nodes	Avg. Edges
ExplaGraphs[11]	5.2	4.3
SceneGraphs[12]	19	85.5
WebQSP[13]	1371	4402.0

ExplaGraphs features explanation-seeking queries over knowledge graphs, often requiring long-form, constraint-rich reasoning grounded in multiple relations. SceneGraphs are constructed from visual scene annotations and emphasize spatial and compositional reasoning among objects. WebQSP-focuses on Freebase-based factual question answering, where precise multi-hop reasoning over concise knowledge subgraphs is essential.

Table I presents key linguistic characteristics across datasets, including average query length, number of logical constraints, and the number of tokens in the full graph context. Notably, ExplaGraphs exhibits the highest linguistic complexity, while WebQSPoffers the most compact formulations with dense reasoning signals.

Table II summarizes the average structural complexity of the input graphs. Again, ExplaGraphs contains significantly larger graphs in terms of nodes and edges, introducing challenges in subgraph construction and input serialization for language models. These variations in textual and structural properties motivate the need for effective multi-path pruning and subgraph construction strategies to isolate the most relevant context from large graph neighborhoods.

Baselines: To ensure a fair and insightful comparison, we implemented several baseline methods that cover both traditional retrieval-augmented generation and advanced graph-based retrieval paradigms. These baselines include:

- **Zero-shot**: The language model directly generates answers based on its internal knowledge, without retrieval or reasoning prompts.
- **Chain-of-Thought (CoT)**[14]: Enhances zero-shot performance by prompting the model to generate intermediate reasoning steps before the final answer.
- **Think-on-Graph (ToG)**[15]: Constructs local subgraphs around entities in the query via heuristic expansion, serializes them, and appends to the prompt for structured reasoning.
- **Vanilla RAG**: Retrieves top-k relevant text passages using dense retrievers, concatenates them with the query, and feeds the combined input into the language model.
- **RAPTOR**[16]: Builds on Vanilla RAG by introducing graph-based reranking signals (e.g., centrality, connectivity) to refine retrieved text passages.
- **Multi-Head RAG**[17]: Performs semantic retrieval in multiple representation spaces via head-wise matching, then fuses the results for joint reasoning with LLMs.
- **GraphToken**[7]: Serializes graph structures into token sequences using structural markers, allowing standard LLMs to process graph data without architectural changes.
- **GraphRAG**[3]: Constructs query-centered subgraphs from knowledge graphs and augments the prompt with structural context for retrieval-augmented generation.
- **G-Retriever**[4]: Uses GNN-based retrieval by propagating query embeddings through graph edges, enabling multi-hop reasoning over relational structures.

Evaluation Metrics: To comprehensively assess the performance of all methods, we employed multiple metrics capturing accuracy, efficiency, and reasoning quality.

Accuracy Metrics: We evaluate answer accuracy across datasets using metrics best aligned with their respective task formats. For ExplaGraphs and SceneGraphs, we report accuracy, defined as the proportion of queries where the predicted answer exactly matches the ground truth. For WebQSP, we adopt the **Hit Rate** (HIT), following existing work [4], which measures whether the correct entity appears among the predicted candidates. In our ablation studies, we further report **F1 score**, computed at the entity level based on exact matches, to assess the impact of individual components on precision-recall balance.

Efficiency Metrics: We measure the average size of the retrieved subgraphs before and after the reranking stage, quantified both by the number of nodes and the total token count fed into the language model prompt. These metrics reflect the computational burden and input complexity for downstream inference. Moreover, end-to-end inference latency is recorded to evaluate practical runtime performance.

All metrics are averaged over the test set of each benchmark, and experiments are repeated across multiple independent seeds. Results report mean values alongside standard deviations to ensure statistical robustness.

B. Accuracy Evaluation

Q1:Does fusion retriever guided subgraph construction improve answer accuracy over standard and graph-based RAG methods?

We evaluate GEAR against a broad spectrum of baselines, organized into three paradigms that reflect varying degrees of graph integration in LLM reasoning: (i) *Inference-based methods* (e.g., Chain-of-Thought, Think-on-Graph) that guide LLMs through explicit intermediate reasoning or local prompt construction; (ii) *Textual RAG methods* (e.g., Vanilla RAG, RAPTOR) that retrieve textual context from external corpora, with RAPTOR incorporating graph signals in reranking; and (iii) (e.g., GraphToken, GraphRAG, G-Retriever) that more directly encode structural information via token serialization or graph-based retrieval.

TABLE III: Retrieval Accuracy Comparison.

Setting	**Method**	**ExplaGraphs**[11]	**SceneGraphs**[12]	**WebQSP**[13]
Inference Method	Zero-shot	0.515	0.397	41.06
	CoT [14]	0.570	0.526	51.30
	ToG [15]	0.535	0.473	54.80
RAG	Vanilla RAG	0.536	0.467	52.42
	RAPTOR [16]	0.622	0.437	52.64
	Multihead RAG [17]	0.645	0.575	<u>61.37</u>
Graph-based RAG	GraphToken [7]	<u>0.744</u>	0.483	47.82
	GraphRAG [3]	0.630	<u>0.631</u>	55.20
	G-Retriever$_{\text{Inference only}}$ [4]	0.574	0.617	51.73
GEAR	Fusion-Union	0.749	0.688	71.20
	Fusion-PCST	**0.753**	0.679	70.80
	Fusion-PPR	0.747	**0.699**	**72.36**
	Δ over best baseline	$\uparrow$ 1.21%	$\uparrow$ 10.78%	$\uparrow$ 17.96%

Table III reports the overall performance across three benchmarks. We observe several consistent patterns: First, Vanilla RAG underperforms all methods except the zero-shot baseline, suggesting that naive text-based retrieval without structured guidance provides limited utility for graph-based reasoning tasks. In contrast, RAPTOR shows strong improvements by incorporating structural signals in reranking, achieving the best results among existing RAG baselines. Among graph-enhanced methods, G-Retriever and GraphRAG demonstrate the benefit of incorporating relational structure. However, their performance is surpassed by our proposed methods across all datasets, confirming the effectiveness of multi-path guided subgraph construction.

Despite introducing no model training or fine-tuning, our variants—Fusion-PCST, Fusion-Union, and Fusion-PRR—consistently outperform all baselines. Notably, Union achieves stable gains across all datasets, while Union excels on ExplaGraphs, likely due to its capacity to retain sparse but semantically relevant paths. PRR performs best on WebQSP, where a greater number of dense entity connections facilitate richer LLM prompting. On the structurally harder SceneGraphs, our methods outperform GraphRAG, indicating that the benefit of additional structural pruning becomes limited when the query graphs exhibit higher semantic complexity. These results highlight the importance of context-aware subgraph construction, especially on benchmarks with complex structure and high query constraint density.

Takeaway 1: GEAR outperforms all baselines on every dataset, *without requiring any model fine-tuning or additional training*. This highlights that fusion-guided subgraph construction alone can yield more relevant and structurally aligned contexts for LLM reasoning.

TABLE IV: Inference Latency (in seconds).

Method	**SceneGraphs**[12]	**WebQSP**[13]
CoT[14]	2.89	3.85
RAPTOR[16]	3.84	3.51
G-Retriever[4]	1.83	2.17
GEAR	**0.66**	**1.32**

C. Efficiency Evaluation

Q2: Can GEAR construct more compact and efficient pipeline without sacrificing performance? We assess GEAR's efficiency in terms of subgraph size, token input length, and end-to-end inference latency. To this end, we compare average statistics before and after subgraph pruning across datasets. As shown in Table V, our path-based subgraph construction significantly reduces both node count and token length. Notably, token reduction often exceeds structural sparsification, reflecting the semantic compression achieved by GEAR. This compactness is obtained without multi-turn prompting or soft prompt tuning, ensuring efficient input encoding while preserving accuracy. Latency results in Table IV further demonstrate favorable inference speed: unlike CoT or RAPTOR, which incur overhead from multi-step reasoning or reranking, GEAR avoids expensive GNN operations and prompt chaining while retaining structural awareness, making it well-suited for real-time applications.

Latency analysis: We report the average end-to-end inference latency in Table IV. As expected, Vanilla RAG achieves the fastest response due to its lightweight retrieval-then-generation pipeline. In contrast, CoT incurs the highest latency, as it invokes the language model multiple times to generate intermediate reasoning steps. RAPTOR, while still grounded in retrieval, introduces a costly graph-based reranking procedure, resulting in inference times comparable to CoT. G-Retriever further increases latency due to the overhead

TABLE V: Efficiency Comparison

Dataset	Before Path (Avg.)			After Path (Avg.)		
	# Tokens	# Nodes	Edges	# Tokens	# Nodes	Edges
SceneGraphs[12]	1,396	19	86	433(↓69%)	8(↓58%)	22(↓74%)
WebQSP[13]	100,627	1,371	4402	81(↓99%)	16(↓99%)	26(↓99%)

of GNN-based message passing and node propagation. Our method achieves competitive latency by avoiding multi-turn prompting and heavy graph encodings, while still maintaining structural awareness via efficient subgraph pruning. This demonstrates favorable inference efficiency, particularly in real-time or resource-constrained scenarios.

Takeaway 2: GEAR reduces prompt tokens and graph size by up to 99% while maintaining accuracy, and achieves competitive latency without relying on multi-turn prompting or GNN inference.

D. Ablation Study

Q3: How do fusion retrieval and structure-aware construction respectively contribute to GEAR performance?

To better understand the contribution of each component in our pipeline, we conduct a series of controlled ablation studies across three datasets and three subgraph construction variants. Specifically, we compare performance with and without (i) the Fusion Retriever module, which performs multi-path semantic retrieval, and (ii) the Fusion-Based Subgraph Construction, which guides structural selection and expansion.

Figure 7 summarizes the results across six settings—each combining a dataset (ExplaGraphs, SceneGraphs, or WebQSP) with one of the two ablated components. For each setting, we visualize both efficiency (bar chart: lower is better) and answer accuracy (line chart: higher is better), with the horizontal axis representing the three subgraph construction variants (Union, PRR, PCST).

Impact of Fusion Retriever: Removing the fusion retriever consistently degrades answer accuracy across all datasets, while having limited effect on inference latency. This demonstrates that multi-path semantic retrieval plays a central role in improving retrieval relevance, yet remains efficient due to its lightweight design. Notably, on ExplaGraphs and SceneGraphs, accuracy drops sharply for all three subgraph methods without fusion, especially for Union, which depends heavily on rich retrieval signals for semantically grounded expansion.

Impact of Fusion-Based Subgraph Construction: Disabling the subgraph construction module results in moderate accuracy degradation on smaller graphs (SceneGraphs), but leads to more significant performance drops on WebQSP, where dense node connectivity and longer reasoning paths demand more precise graph pruning. This confirms that structure-aware selection contributes substantially in high-complexity settings. Importantly, latency reduction from this ablation remains marginal, as fewer edges are trimmed without substantial savings in token length or LLM input size.

Comparison of Subgraph Variants: Among the three variants, Fusion-Union consistently outperforms others in maintaining accuracy across all datasets. Its ability to expand from multiple seed paths makes it robust to both sparse and dense graph structures. In contrast, Fusion-PCST, which aggressively extends subgraphs by estimated path cost, achieves the most benefit on WebQSP, reflecting its advantage in deep graph contexts. Fusion-PRR shows its strongest contribution on SceneGraphs, where node-level path traversal is sufficient for reasoning. These observations suggest that different graph patterns exhibit different sensitivity to fusion components, and our modular design enables flexible adaptation.

Takeaway 3: Fusion retrieval improves accuracy across all datasets with minimal cost, while structure-aware construction is critical for handling dense or complex graphs, particularly on WebQSP.

E. Case Study

As illustrated in Figure 8, a direct traversal from the entity Chelsea Kane via the profession relation retrieves explicit roles such as Singer, Dancer, and Voice Actor. These results are faithful to the information explicitly recorded in the knowledge graph, but remain limited to surface-level labels.

In this example, further analysis reveals that semantic cues related to professions can emerge through indirect connections. For instance, via the `actor.film` relation, Chelsea Kane links to the film entity *m.09p2m7y*. While this link lacks explicit profession annotation, associated metadata—such as `tv_guest_role.actor` and cast information—enables the system to infer that she is an actress, even though this fact is not explicitly stated in the graph.

This case underscores the importance of integrating multiple evidence sources. By combining structured paths (e.g., `actor.film`) with lexical context (e.g., role labels and film metadata), the system can recover facts that traditional path-based retrieval may overlook, thereby mitigating sparsity and incomplete annotation in the knowledge graph.

VI. Related Work

A. Retrieval-Augmented Generation

Large Language Models (LLMs), despite their capabilities, often generate factually inaccurate content, referred to as "hallucinations", and their knowledge base is static and can quickly become outdated [1], [18]. Retrieval-Augmented

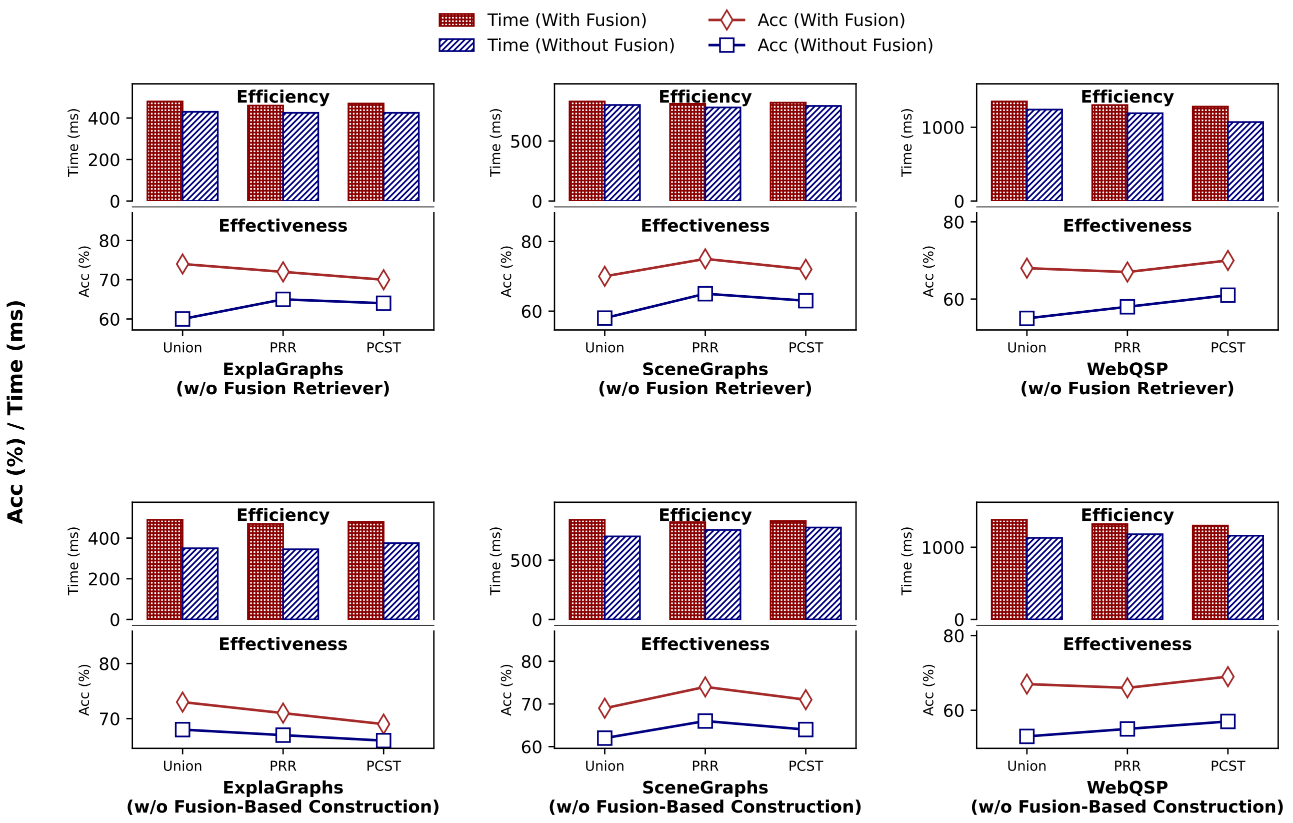

Fig. 7: Ablation Study on Components' Influence on Accuracy and Efficiency

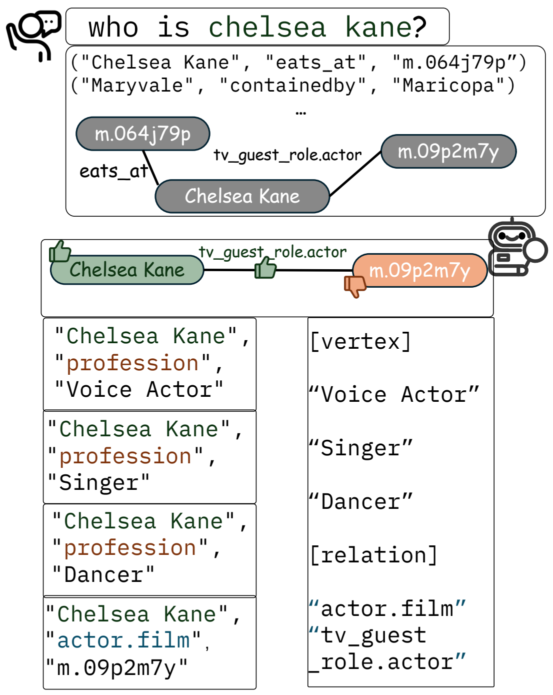
who is chelsea kane?
("Chelsea Kane", "eats_at", "m.064j79p")
("Maryvale", "containedby", "Maricopa")
...
m.064j79p
eats_at
tv_guest_role.actor
m.09p2m7y
Chelsea Kane
Chelsea Kane
tv_guest_role.actor
m.09p2m7y
"Chelsea Kane", "profession", "Voice Actor"
"Chelsea Kane", "profession", "Singer"
"Chelsea Kane", "profession", "Dancer"
"Chelsea Kane", "actor.film", "m.09p2m7y"
[vertex]
"Voice Actor"
"Singer"
"Dancer"
[relation]
"actor.film"
"tv_guest _role.actor"

Fig. 8: Case Study with GEAR

Generation (RAG) was introduced as a powerful paradigm to mitigate these issues by enabling LLMs to consult external, up-to-date knowledge sources before generating a response [2]. The overall performance of a RAG system, however, is heavily dependent on the quality and efficiency of its retrieval component.

To move beyond simple vector similarity, the field has rapidly evolved to explore more sophisticated methods. Advanced indexing structures, such as trees [16] and graphs [3], [5], have been developed to better represent and navigate complex information. Simultaneously, research has focused on refining the synergy between the retriever and the generator. Techniques such as iterative retrieval, which refines search queries over multiple steps [19], and self-correction mechanisms that allow the model to critique and improve upon the retrieved context [20], [21], have been proposed. Other efforts have focused on jointly training the retriever and generator to better align their objectives and improve the end-to-end performance [22].

Despite these innovations, a fundamental challenge persists. Many of these methods struggle to efficiently synthesize information from disparate knowledge fragments, particularly for complex queries that require reasoning over intricate entity relationships [21]. Furthermore, while iterative and self-reflective approaches can enhance output quality, they introduce significant computational overhead and increase latency, undermining the efficiency of large-scale RAG deployments [22][23].

B. Graph-Structured Knowledge in RAG

To better address queries requiring complex, multi-hop reasoning, there is a growing movement towards integrating structured knowledge, particularly knowledge graphs, into RAG frameworks. This approach, often called "GraphRAG"[24], [3], leverages the explicit relational structure of graphs to augment the semantic understanding of LLMs.

Current research in GraphRAG has progressed along several distinct paths. One prominent approach utilizes Graph Neural Networks (GNNs) to encode the graph's structural information, thereby guiding the retrieval of relevant subgraphs [4], [25]. G-Retriever, for instance, propagates query information through the graph to perform multi-hop retrieval of evidence nodes [4]. A second approach involves linearizing graph structures into text sequences that can be directly ingested by LLMs [7], [26]. While these methods are innovative, they each come with significant trade-offs. GNN-based retrieval can be computationally intensive and slow, posing a bottleneck for real-time applications and motivating research into specialized graph database optimizations [25]. Conversely, linearization risks losing critical structural information during the conversion to text and can result in excessively long input sequences, which increases the cost and processing burden on the LLM.

Other notable research aims to enhance the reasoning capabilities over graphs. For example, HopRAG is designed to maintain path coherence during multi-hop retrieval [6], while Hyper-RAG employs hypergraphs to capture more complex, higher-order relationships between entities [8]. However, a persistent limitation across most GraphRAG systems is their reliance on a single, monolithic semantic space for representing both queries and graph elements. This creates a "semantic gap", making it difficult to capture the multifaceted nature of complex queries and leading to inefficient or incomplete retrievals that fail to gather all necessary evidence.

C. Data-centric Optimization in RAG Pipelines

Recognizing the limitations of optimizing individual components in isolation, recent research from both the machine learning and data systems communities has shifted towards a more holistic, data-centric optimization of the entire RAG pipeline [27]. A fragmented pipeline, where the retriever, indexer, and generator are treated as separate problems, often suffers from "inter-module misalignment", which degrades overall performance and efficiency.

From the data systems perspective, researchers are architecting new systems designed to natively support the unique demands of RAG workloads. This involves moving beyond simple component-level fixes to rethink the end-to-end data flow[28]. For example, Chameleon proposes a heterogeneous and disaggregated hardware accelerator system specifically for RAG [27], while PipeRAG introduces adaptive pipeline parallelism to accelerate query processing by overlapping retrieval and generation stages [29].

Concurrently, more sophisticated data-centric strategies are developed to ensure semantic coherence throughout the pipeline. KET-RAG, for instance, proposes a cost-efficient, multi-granular indexing framework to better structure knowledge for retrieval [30]. Similarly, MultiRAG presents a knowledge-guided framework to mitigate hallucinations when retrieving from multiple knowledge sources, ensuring that the combined evidence is consistent and relevant [31]. This trend is further underscored by calls for a unified framework for GraphRAG to standardize evaluation and foster more integrated pipeline designs [24]. These efforts collectively signal a critical shift from component-centric tuning to data-centric, end-to-end pipeline optimization.

Our work is motivated by these persistent challenges. The dual problems of the "semantic gap" (a quality issue stemming from single-view representations) and "inter-module misalignment" (an efficiency and quality issue arising from fragmented pipelines) are central to the current limitations of GraphRAG. We argue that a holistic, end-to-end framework is required-one that maintains a multi-faceted understanding of a query's intent throughout the entire pipeline, from retrieval to subgraph construction, to simultaneously improve the quality and efficiency of graph-based RAG.

VII. Conclusion

This paper introduces GEAR (Graph-Efficient Augmented Retrieval), a novel framework designed to resolve the critical "semantic gap" and "inter-module misalignment" issues that currently hinder the performance of graph-based Retrieval-Augmented Generation (RAG) systems. By employing a multi-head architecture, GEAR effectively processes queries across various semantic dimensions, unifying the retrieval, re-ranking, and subgraph construction phases into a single, cohesive pipeline . This integrated approach ensures that the final subgraph presented to the Large Language Model (LLM) is precisely aligned with the user's intent .

A key advantage of GEAR is its significant efficiency. The framework achieves a reduction of up to 99% in both prompt tokens and graph size while maintaining high accuracy . This compactness is achieved without resorting to costly multi-turn prompting or GNN operations, leading to competitive end-to-end inference latency . Consequently, GEAR is well-suited for real-time applications, demonstrating superior performance and efficiency over state-of-the-art baselines across three distinct benchmark datasets.

Future work could explore more adaptive subgraph construction methods tailored to specific query types and investigate combining GEAR's multi-perspective retrieval with advanced RAG strategies like iterative or reflective generation . Furthermore, end-to-end training of the RAG components with the LLM could unlock deeper semantic integration and system-wide optimizations.

AI-Generated Content Acknowledgement

During the preparation of this manuscript, we utilized artificial intelligence (AI) systems to enhance the quality of the writing. Specifically, the language models DeepSeek and QWen were employed for grammar correction and text polishing across the entire article. The role of these AI systems was limited to improving the clarity, style, and grammatical accuracy of the content, which was conceived and written by the human authors. The AI was not used to generate scientific ideas, experimental results, or the core textual content.

References

[1] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, and P. Fung, "Survey of hallucination in natural language generation," *ACM computing surveys*, vol. 55, no. 12, pp. 1–38, 2023.

[2] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks," *Advances in neural information processing systems*, vol. 33, pp. 9459–9474, 2020.

[3] H. Han, Y. Wang, H. Shomer, K. Guo, J. Ding, Y. Lei, M. Halappanavar, R. A. Rossi, S. Mukherjee, X. Tang *et al.*, "Retrieval-augmented generation with graphs (graphrag)," *arXiv preprint arXiv:2501.00309*, 2024.

[4] X. He, Y. Tian, Y. Sun, N. Chawla, T. Laurent, Y. LeCun, X. Bresson, and B. Hooi, "G-retriever: Retrieval-augmented generation for textual graph understanding and question answering," *Advances in Neural Information Processing Systems*, vol. 37, pp. 132 876–132 907, 2024.

[5] D. Edge, H. Trinh, N. Cheng, J. Bradley, A. Chao, A. Mody, S. Truitt, D. Metropolitansky, R. O. Ness, and J. Larson, "From local to global: A graph rag approach to query-focused summarization," *arXiv preprint arXiv:2404.16130*, 2024.

[6] H. Liu, Z. Wang, X. Chen, Z. Li, F. Xiong, Q. Yu, and W. Zhang, "Hoprag: Multi-hop reasoning for logic-aware retrieval-augmented generation," *arXiv preprint arXiv:2502.12442*, 2025.

[7] B. Perozzi, B. Fatemi, D. Zelle, A. Tsitsulin, M. Kazemi, R. Al-Rfou, and J. Halcrow, "Let your graph do the talking: Encoding structured data for llms," *arXiv preprint arXiv:2402.05862*, 2024.

[8] Y. Feng, H. Hu, X. Hou, S. Liu, S. Ying, S. Du, H. Hu, and Y. Gao, "Hyper-rag: Combating llm hallucinations using hypergraph-driven retrieval-augmented generation," *arXiv preprint arXiv:2504.08758*, 2025.

[9] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *J. Mach. Learn. Res.*, vol. 21, pp. 140:1–140:67, 2020. [Online]. Available: http://jmlr.org/papers/v21/20-074.html

[10] A. Dubey, A. Grattafiori, A. Jauhri, A. Pandey, A. Kadian, and ..., "The llama 3 herd of models," 2024. [Online]. Available: https://arxiv.org/abs/2407.21783

[11] S. Saha, P. Yadav, L. Bauer, and M. Bansal, "Explagraphs: An explanation graph generation task for structured commonsense reasoning," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021, pp. 7716–7740.

[12] X. Chang, P. Ren, P. Xu, Z. Li, X. Chen, and A. Hauptmann, "A comprehensive survey of scene graphs: Generation and application," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 1, pp. 1–26, 2021.

[13] W.-t. Yih, M. Richardson, C. Meek, M.-W. Chang, and J. Suh, "The value of semantic parse labeling for knowledge base question answering," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2016, pp. 201–206.

[14] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in neural information processing systems*, vol. 35, pp. 24 824–24 837, 2022.

[15] J. Sun, C. Xu, L. Tang, S. Wang, C. Lin, Y. Gong, L. M. Ni, H.-Y. Shum, and J. Guo, "Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph," *arXiv preprint arXiv:2307.07697*, 2023.

[16] P. Sarthi, S. Abdullah, A. Tuli, S. Khanna, A. Goldie, and C. D. Manning, "Raptor: Recursive abstractive processing for tree-organized retrieval," in *The Twelfth International Conference on Learning Representations*, 2024.

[17] M. Besta, A. Kubicek, R. Niggli, R. Gerstenberger, L. Weitzendorf, M. Chi, P. Iff, J. Gajda, P. Nyczyk, J. Müller *et al.*, "Multi-head rag: Solving multi-aspect problems with llms," *arXiv preprint arXiv:2406.05085*, 2024.

[18] I. Augenstein, T. Baldwin, M. Cha, T. Chakraborty, G. L. Ciampaglia, D. Corney, R. DiResta, E. Ferrara, S. Hale, A. Halevy *et al.*, "Factuality challenges in the era of large language models," *arXiv preprint arXiv:2310.05189*, 2023.

[19] Z. Jiang, F. F. Xu, L. Gao, Z. Sun, Q. Liu, J. Dwivedi-Yu, Y. Yang, J. Callan, and G. Neubig, "Active retrieval augmented generation," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023, pp. 7969–7992.

[20] A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi, "Self-rag: Learning to retrieve, generate, and critique through self-reflection," in *International Conference on Learning Representations*, 2024.

[21] Y. Mao, X. Dong, W. Xu, Y. Gao, B. Wei, and Y. Zhang, "Fit-rag: Black-box rag with factual information and token reduction," *arXiv preprint arXiv:2403.14374*, 2024.

[22] W. Fan, Y. Ding, L. Ning, S. Wang, H. Li, D. Yin, T.-S. Chua, and Q. Li, "A survey on rag meeting llms: Towards retrieval-augmented large language models," in *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2024, pp. 6491–6501.

[23] H. Zhang, X. Ji, Y. Chen, F. Fu, X. Miao, X. Nie, W. Chen, and B. Cui, "Pqcache: Product quantization-based kvcache for long context llm inference," *arXiv preprint arXiv:2407.12820*, 2024.

[24] Y. Zhou, Y. Su, Y. Sun, S. Wang, T. Wang, R. He, Y. Zhang, S. Liang, X. Liu, Y. Ma, and Y. Fang, "In-depth analysis of graph-based rag in a unified framework," 2025. [Online]. Available: https://arxiv.org/abs/2503.04338

[25] D. Yu, R. Bao, G. Mai, and L. Zhao, "Spatial-rag: Spatial retrieval augmented generation for real-world spatial reasoning questions," *arXiv preprint arXiv:2502.18470*, 2025.

[26] Y. Hu, Z. Lei, Z. Zhang, B. Pan, C. Ling, and L. Zhao, "Grag: Graph retrieval-augmented generation," *arXiv preprint arXiv:2405.16506*, 2024.

[27] W. Jiang, M. Zeller, R. Waleffe, T. Hoefler, and G. Alonso, "Chameleon: A heterogeneous and disaggregated accelerator system for retrieval-augmented language models," *Proc. VLDB Endow.*, vol. 18, no. 1, p. 42–52, Sep. 2024.

[28] Y. Zhou, M. Yan, J. Yao, and G. Xu, "Prorag: Towards reliable and proficient aigc-based digital avatar," in *Proceedings of the 30th International Conference on Database Systems for Advanced Applications (DASFAA)*, 2025, accepted to appear.

[29] W. Jiang, S. Zhang, B. Han, J. Wang, B. Wang, and T. Kraska, "Piperag: Fast retrieval-augmented generation via adaptive pipeline parallelism," in *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1*, ser. KDD '25, 2025, p. 589–600.

[30] Y. Huang, S. Zhang, and X. Xiao, "Ket-rag: A cost-efficient multi-granular indexing framework for graph-rag," 2025. [Online]. Available: https://arxiv.org/abs/2502.09304

[31] W. Wu, H. Wang, B. Li, P. Huang, X. Zhao, and L. Liang, " MultiRAG: A Knowledge-Guided Framework for Mitigating Hallucination in Multi-Source Retrieval Augmented Generation ," in *2025 IEEE 41st International Conference on Data Engineering (ICDE)*, May 2025, pp. 3070–3083.