MGRAG: Semantic Subgraph Matching and Graph-Aware Caching for Multimodal Retrieval-Augmented Generation

Yubo Wang
The Hong Kong University of Science and Technology
Hong Kong SAR, China
ywangnx@connect.ust.hk

Haoyang Li*
The Hong Kong Polytechnic University
Hong Kong SAR, China
haoyang-comp.li@polyu.edu.hk

Lei Chen
The Hong Kong University of Science and Technology (Guangzhou)
Guangzhou, China
leichen@hkust-gz.edu.cn

ABSTRACT

Answering complex queries over large and heterogeneous multimodal document corpora is a central challenge in data management, requiring fine-grained, entity-level evidence retrieval and efficient context serving. Graph-based Retrieval-Augmented Generation (RAG) systems achieve promising effectiveness by organizing multimodal documents as knowledge graphs (KGs); however, they still face three limitations: (1) query-agnostic KG construction, where corpus-wide graphs overwhelm query-relevant entities with irrelevant noise; (2) inflexible graph matching, which relies on rigid topological matching and misses path-level semantic equivalences; (3) event-agnostic KV re-computation, which scores tokens independently of graph topology, failing to preserve event-level semantic structure. To address these issues, we propose MGRAG. First, MGRAG incrementally builds query-specific KGs on demand via a lazy, top-down construction strategy. Second, we formulate graph retrieval as a path-based semantic subgraph matching problem, prove it NP-hard, and design an efficient greedy algorithm for flexible, semantics-aware retrieval. Third, MGRAG employs an event-aware KV caching mechanism to selectively recompute tokens critical to query-related events. Experiments on seven real-world multimodal QA datasets show that MGRAG achieves superior effectiveness and efficiency compared to state-of-the-art RAG, subgraph matching, and KV caching baselines.

PVLDB Reference Format:
Yubo Wang, Haoyang Li, and Lei Chen. MGRAG: Semantic Subgraph Matching and Graph-Aware Caching for Multimodal Retrieval-Augmented Generation. PVLDB, 19(9): 2480 - 2493, 2026.
doi:10.14778/3819518.3819565

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/Wyb0627/MGRAG.

1 INTRODUCTION

Answering complex queries over large, heterogeneous multimodal document corpora is a central challenge in data management [46, 47, 49, 53, 55]. Given a query and a corpus containing textual and image documents, the system is expected to retrieve the minimal set of fine-grained, entity-level evidence that is both sufficient and precise for answering the query, and serve the retrieved context to a generative model efficiently, especially when many queries share overlapping evidence. The first step is fundamentally a query-retrieval problem over structured or semi-structured data, closely related to subgraph matching [30, 66] and query processing over knowledge graphs [32] studied extensively in the database community. The second is a cache-management problem for intermediate computation states, akin to buffer and view management in query processing systems [33, 59]. While modern large language models (LLMs) such as GPT4 [4] and Qwen2.5 [8] serve as powerful generation backends, the quality of their answers is ultimately bounded by the quality of the retrieved evidence [18, 36, 38, 65]. This motivates the design of more principled retrieval and caching mechanisms as core data-management contributions, with LLMs positioned as the downstream consumer of the retrieved results.

Multimodal retrieval-augmented generation (RAG) [15, 19, 20, 23–25] provides a general framework for this task: an indexing module organizes the corpus, a retrieval module selects partial content across modalities, and an LLM generates the final answer. A key data-management challenge is to retrieve information that is both fine-grained enough to support multi-hop reasoning and compact enough to avoid the noise and latency penalties of overly lengthy contexts [31, 40, 43, 65]. Depending on their indexing and retrieval strategies, current multimodal RAG systems fall into two categories. *Similarity-based RAG* systems [20, 42, 62] encode multimodal data into a shared vector space using encoders such as CLIP [45] and retrieve the top-k most similar chunks. However, representing each chunk as a single vector obscures fine-grained entity-level information, leading to coarse-grained retrieval. To achieve entity-level granularity, *Graph-based RAG* systems [9, 15, 23–26, 63] use LLMs to extract entities and relations from multimodal documents, organize them as nodes and edges in a knowledge graph (KG), and retrieve relevant graph components at query time. However, from a data-management perspective, existing graph-based RAG approaches still face three fundamental challenges in their indexing, retrieval, and caching stages.

Challenge 1: Query-agnostic KG construction. Existing graph-based approaches [15, 23, 25] construct a single, corpus-wide KG offline by extracting all entities and relations from every document, without considering the focus of any specific query. This is analogous to materializing a universal view without knowing the query workload: query-relevant entities, especially fine-grained visual events extracted from images, are overwhelmed by irrelevant noise, and information critical for a specific query may never be

*Corresponding author

Proceedings of the VLDB Endowment, Vol. 19, No. 9 ISSN 2150-8097.
doi:10.14778/3819518.3819565

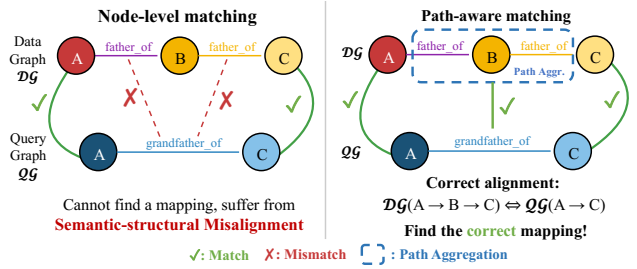


Figure 1: A motivating example of our path-based semantic subgraph matching. The query's `grandfather_of` edge matches a two-hop `father_of` path in the data graph. Node-wise matching (left) misses this alignment, while our method (right) aggregates paths to align them.

extracted at all. The challenge is to design an adaptive, query-driven graph construction strategy that incrementally builds a compact, query-specific KG from the most relevant documents, balancing construction cost against retrieval recall.

Challenge 2: Inflexible graph matching. Once a KG is available, the retrieval step should identify the subgraph that best answers the query. Current approaches either retrieve individual nodes or triples by vector similarity with a fixed result size k [23], or apply rigid neighborhood expansion [25], both of which ignore path-level semantic equivalences. As shown in Figure 1, consider a data graph $\mathcal{DG}$ containing the two-hop path $A \xrightarrow{\texttt{father\_of}} B \xrightarrow{\texttt{father\_of}} C$, and a query graph $\mathcal{QG}$ containing the single edge $A \xrightarrow{\texttt{grandfather\_of}} C$. Traditional node-level matching fails because no single edge in $\mathcal{DG}$ corresponds to `grandfather_of`, resulting in an inflexible graph matching. This kind of mismatch arises frequently in multimodal KGs, where a single query edge about a visual event may correspond to a multi-hop path through fine-grained entities extracted from images.

Challenge 3: Event-agnostic KV cache management. In multi-user serving scenarios, current RAG serving systems [11, 33, 41, 59, 68] cache the key-value (KV) states of retrieved contexts to accelerate inference for subsequent queries that reuse the same chunks. Existing methods, such as CacheBlend [59], decide which tokens to selectively recompute based solely on token-level cross-attention scores. However, in multimodal contexts, particularly when processing images, each image is represented as a sequence of discrete tokens, fragmenting holistic visual events into isolated segments. Token-level scoring fails to capture the event-level structural and semantic relationships among these tokens, which were naturally encoded in the KG topology.

Our approach. To address these three data-management challenges, we propose MGRAG, a graph-based multimodal RAG framework with three core stages. *First*, MGRAG uses dense encoders to rank multimodal chunks by semantic relevance to the query, then incrementally constructs a query-specific multimodal KG by extracting entities and relations with an LLM in a query-aware, top-down manner. Following the database principle of lazy view materialization, MGRAG builds only the graph fragment needed for the current query, with subgraph-matching-driven early termination. *Second*, to enable adaptive and semantically flexible graph retrieval, we formulate a Path-based Semantic Subgraph Matching problem that generalizes classical subgraph isomorphism [13, 21] by relaxing topological constraints with path-level semantic aggregation. We prove this problem is NP-hard and design an efficient greedy algorithm based on semantic path growing and VF2-based isomorphism enumeration [13], whose practical cost is modest on the small, query-specific graphs constructed by MGRAG. *Third*, MGRAG introduces an event-aware KV caching mechanism that incorporates four graph centrality measures: degree, betweenness, eigenvector, and closeness. MGRAG utilizes these features to guide selective re-computation of cached states, bridging the gap between token-level attention scores and event-level structural importance and preserving critical semantic relationships during inference.

We summarize our contributions, which center on graph construction, query retrieval, and cache management:

- We propose MGRAG, a multimodal graph-based RAG system that incrementally builds query-specific KGs on demand via a lazy, top-down construction strategy, avoiding the cost and quality drawbacks of corpus-wide graph indexing.
- We formulate graph retrieval as a new path-based semantic subgraph matching problem, prove it NP-hard, and design an efficient greedy algorithm to solve it.
- We introduce a cache re-computation policy guided by graph centrality, enabling the caching mechanism to preserve semantic coherence in multimodal contexts.
- Experiments on seven real-world multimodal QA datasets demonstrate that MGRAG achieves superior effectiveness and efficiency compared to state-of-the-art RAG, subgraph matching, and KV caching baselines.

2 PRELIMINARY AND RELATED WORKS

This section reviews related work on three core aspects of multimodal RAG data management: multimodal RAG pipelines (Section 2.1), subgraph matching (Section 2.2), and KV cache management (Section 2.3). Key notations are summarized in Table 1.

2.1 Multimodal RAG

Although LLMs excel at multimodal tasks, they often lack up-to-date or domain-specific knowledge and struggle to recall long-tail facts, leading to hallucinated responses [36, 38, 65]. Multimodal RAG systems [3, 5, 15, 20, 23–25, 42, 48, 54, 61, 64, 67] address these gaps by retrieving up-to-date, relevant multimodal content, such as text and images, from large external sources. Formally, given a user query q, a multimodal document corpus $\mathcal{D} = \mathcal{D}_{txt} \cup \mathcal{D}_{img}$ consisting of textual documents $\mathcal{D}_{txt}$ and image documents $\mathcal{D}_{img}$, and an LLM with multimodal capabilities, a retrieval module $\Omega(\cdot)$ first retrieves a set of query-relevant information contexts C_q from the multimodal chunk set C, which is generated from the original corpus $\mathcal{D}$ by an indexing module Γ: $C_q = \Omega(C, q), C = \Gamma(\mathcal{D})$. Next, the LLM generates an answer A_q conditioned on the query and the retrieved contexts C_q: $\mathcal{A}_q = \text{LLM}(q, C_q)$.

Depending on the technique, multimodal RAG approaches [5, 15, 20, 23–25, 42, 48, 54, 61, 64] fall into two categories: similarity-based and graph-based. In similarity-based systems [5, 20, 42, 48, 54, 64], Γ

encodes queries and documents into a shared embedding space, and Ω returns the top-k most similar chunks or images. For example, VanillaRAG [20] encodes text chunks and images with pre-trained multimodal encoders (e.g., CLIP [45]) and retrieves by cosine similarity. While straightforward, this coarse-grained retrieval cannot capture fine-grained, entity-level semantics that span multiple chunks, nor does it provide explicit relational reasoning to the LLM about why each chunk was retrieved. On the other hand, graph-based systems [15, 23–25, 61] address this by having Γ construct a knowledge graph from multimodal documents and having Ω retrieve structurally relevant subgraphs. GraphRAG [15] partitions the graph into communities and generates community-level summaries, but incurs high costs for community detection and summarization. HippoRAG [24] scores nodes via Personalized PageRank and retrieves chunks associated with top-scoring nodes, reducing context length. HippoRAG2 [25] further improves graph connectivity by indexing entire passages as nodes.

Despite recent advances, existing graph-based systems have three main limitations: (1) query-agnostic graph construction leads to high indexing costs and noisy, less accurate retrieval; (2) retrieval is limited to node or edge matching, missing multi-hop path semantics; (3) a fixed retrieval budget k cannot flexibly support queries needing variable evidence.

2.2 Subgraph Matching

To adaptively retrieve query-relevant information from the external KG, subgraph matching methods can identify the most relevant subgraphs that align with the query graph, directly affecting both the effectiveness and efficiency of query retrieval in the RAG data-management pipeline. Classical methods can be grouped into three categories: exact solvers [13, 27], approximate graph-matching algorithms [12, 17, 34, 35], and semantic graph-alignment methods [51, 52, 57].

Exact solvers [13, 27] enumerate all subgraphs of $\mathcal{KG}$ isomorphic to $\mathcal{QG}$. VF2 [13] uses depth-first backtracking with feasibility pruning on degrees, neighborhoods, and labels, while ISMAGS [27] extends it with symmetry-aware search to avoid redundant embeddings. However, exact solvers require strict topological isomorphism, making them unable to address the inflexible graph matching issue in Figure 1. Also, approximate graph-matching algorithms relax isomorphism constraints into optimization over soft correspondence matrices. SM [34] uses eigen-decomposition of an affinity matrix for soft scores, RRWM [12] interprets matching as a reweighted random walk with Sinkhorn normalization, VJ [17] expands seeded correspondences via bipartite matching, and IPFP [35] solves a quadratic assignment relaxation via integer-projected fixed-point iteration. These methods tolerate structural noise but still operate at individual node or edge level without aggregating path-level semantics. Lastly, semantic graph-alignment methods operate in embedding spaces derived from node/edge attributes. SLOTAlign [51] jointly refines graph structure and computes an optimal-transport alignment between attributed graphs, while FGWAlign [52] employs fused Gromov-Wasserstein distances to jointly model attribute and topological similarity. Despite leveraging semantic embeddings, these methods still match at the node or edge level without modeling path-level, query-aware semantics.

Table 1: Summary of important notations.

Notations	Meanings
q	The input user query.
$\mathcal{D}$	The multimodal document corpus.
Ω	The multimodal retrieval module.
$\mathcal{D}_{txt}$, $\mathcal{D}_{img}$	The set of textual and image documents in $\mathcal{D}$.
Γ, Ω	The indexing and retrieval module.
$\mathcal{C}$	The set of multimodal chunks generated from $\mathcal{D}$.
$\mathcal{T}$	The set of text chunks split from $\mathcal{D}_{txt}$.
c	A multimodal chunk (text or image).
$\overrightarrow{\mathbf{h}_*}$	The vector encoding of the respective component.
$\mathcal{C}_q^{rank}$	The coarse-grained ranked chunks regarding query q.
$\mathcal{KG}_q(\mathcal{V}_q, \mathcal{E}_q)$	The query-specific multimodal KG.
$\mathcal{V}_q$, $\mathcal{E}_q$	The set of entities and relations in $\mathcal{KG}_q$.
$\mathcal{QG}_q(\hat{\mathcal{V}}_q, \hat{\mathcal{E}}_q)$	The query graph constructed from query q.
$\hat{\mathcal{V}}_q$	Entities (including placeholders) in the query graph.
$\hat{\mathcal{E}}_q$	Relations in the query graph.
$\mathcal{R}$	The candidate matched subgraph set
res	The final matched subgraph.
S_r	Matched subgraph r's serialized string sequence.
$\oplus(\cdot)$	The path aggregation function.
θ	Semantic similarity threshold for subgraph matching.
$\phi(t)$	The KV recompute score for token t.
GF_t^q	The query specific graph topological score for token t.

2.3 KV Cache Management

To mitigate the high computational cost of autoregressive decoding in LLMs, key-value (KV) caching is widely adopted during inference. The core idea is to reuse previously computed key and value vectors in the attention mechanism, avoiding redundant recomputation across decoding steps [37]. Formally, consider an LLM with L transformer layers. At each layer ℓ, for an input token sequence $\mathbf{x} = [x_1, \ldots, x_n]$, the self-attention mechanism computes query $\mathbf{Q}^\ell$, key $\mathbf{K}^\ell$, and value $\mathbf{V}^\ell$ matrices as: $\mathbf{Q}^\ell = \mathbf{X}\mathbf{W}_Q^\ell, \mathbf{K}^\ell = \mathbf{X}\mathbf{W}_K^\ell, \mathbf{V}^\ell = \mathbf{X}\mathbf{W}_V^\ell$, where $\mathbf{X}$ is the token embedding matrix, and $\mathbf{W}_Q^\ell, \mathbf{W}_K^\ell, \mathbf{W}_V^\ell$ are learnable projection matrices. The attention output is then:

$$\text{Attention}(\mathbf{Q}^\ell, \mathbf{K}^\ell, \mathbf{V}^\ell) = \text{softmax}\left(\frac{\mathbf{Q}^\ell (\mathbf{K}^\ell)^\top}{\sqrt{d_k}}\right)\mathbf{V}^\ell. \tag{1}$$

During autoregressive generation, previously computed key/value (KV) matrices for tokens $x_1, \ldots, x_n$ can be cached and reused, since they depend only on past tokens. This caching speeds up inference. However, in Retrieval-Augmented Generation (RAG), the retrieved context C_q and its integration (e.g., via positional encoding or cross-attention) can vary per query. Thus, blindly reusing cached KV for identical context chunks across queries may harm generation quality, as their representation can differ by query. Proper cache management is necessary.

Definition 1 (KV Cache Management). *Given a user query q, a set of multimodal contexts C_q, and a transformer-based LLM, KV cache management accelerates autoregressive decoding by reusing previously computed key-value (KV) states for tokens in $C_q \cup \{q\}$, while selectively recomputing KV states for a subset of tokens critical to generation quality. Formally, let $\mathcal{T} = Tokenize(q \,\|\, C_q)$ denote the*

full input token sequence, with $\|$ indicating concatenation. For each token $t \in \mathcal{T}$, the system computes a recompute score $\phi(t)$ reflecting its importance. At each transformer layer, only the top-b% tokens with highest $\phi(t)$ values are recomputed, while the KV states of the remaining tokens are reused from cache:

$$KV(t) = \begin{cases} Recompute(t), & \text{if } \phi(t) \text{ is in top } b\% \\ Cache(t), & \text{otherwise} \end{cases} \quad (2)$$

where $Cache(t)$ denotes the cached KV state from a previous inference, and $Recompute(t)$ denotes freshly computed KV projections.

Current KV cache mechanisms are classified as Position Dependent Caching and Position Independent Caching. Position dependent approaches, such as vLLM [33], reuse KV matrices for tokens at the same positions, which improves efficiency for repeated prompts but provides limited acceleration due to short shared subsequences. Position independent approaches [6, 59] reuse KV caches based on token importance, such as feature distances [6] or attention scores [59], allowing broader reuse and greater speedup. However, these methods do not consider the structural relationships among tokens, like events in image data, which restricts their effectiveness.

3 METHODOLOGY

We introduce the overall MGRAG framework, detail its three core stages (Coarse-grain Ranking, Fine-grain Retrieval, Graph-Aware KV Cache), and deliver the complexity analysis.

3.1 Framework Overview

To address *query-agnostic KG construction*, *inflexible graph matching*, and *event-agnostic KV re-computation* issues of current multimodal RAG systems. We propose MGRAG, as shown in Figure 2, MGRAG consists of 3 stages, i.e., Coarse-grain Ranking, Fine-grain Retrieval, and Graph-Aware KV Caching.

Step 1: Coarse-grain Ranking. MGRAG first encodes source multimodal documents in $\mathcal{D} = \{\mathcal{D}_{txt}, \mathcal{D}_{img}\}$ to prepare for retrieval. Text documents in $\mathcal{D}_{txt}$ are split into overlapping chunks and encoded, while image captions C are used as vector representations for images in $\mathcal{D}_{img}$, forming the multimodal chunks C. MGRAG then ranks multimodal chunks C regarding their relevance to the query q, obtaining ranked chunks C_q^{rank}.

Step 2: Fine-grain Retrieval. In this step, MGRAG iterates the ranked multimodal chunks C top-down to construct or enrich the query q specific multimodal KG $\mathcal{KG}_q$. The iteration terminates when a semantically matched subgraph is found between the query graph $\mathcal{QG}_q$ and the multimodal KG $\mathcal{KG}_q$, or when the iteration limit μ is reached. We formalize subgraph matching as a *Path-based Semantic Subgraph Matching* problem, which we prove to be NP-hard. In this formulation, a path P in the query graph $\mathcal{QG}_q$ is mapped to a subgraph r in the multimodal KG $\mathcal{KG}_q$. To solve this problem, we design a greedy algorithm.

Step 3: Graph-Aware KV Cache. Here, MGRAG selectively recomputes the KV cache of each token t based on a cache recompute score $\phi(t)$, which comprises three components: (1) the cross-attention score between token t and the query q, (2) the cross-attention score between token t and the serialized graph sequence S_r, and (3) the graph feature score GF_t^q. To capture topological properties in small graphs, GF_t^q incorporates four key graph topological measures of the triplet fact on graph that associated with token t: degree, betweenness, closeness, and eigenvector centrality. MGRAG will recompute the KV matrix for b% of the tokens in each LLM layer.

Table 2: The scoring function of KV caching methods, where Prefix denotes the tokens in the input prefix, $KV^{full}[t]$ and $KV[t]$ denotes the fully computed, and previously cached KV matrices of token t, respectively.

Method	Feature	Scoring Function $\phi(t)$
vLLM [33]	Position Dependent	$1[t \in \text{Prefix}]$
CacheBlend [59]	Position Independent	$\|\|KV[t] - KV^{full}[t]\|\|_2$
MGRAG	Graph Structure Dependent	$\sum_{j=1}^{\|q\|} a_{tj} + \sum_{k=1}^{\|S_r\|} a_{tk} + GF_t^q$

At last, MGRAG would input the matched subgraph sequence S_r, the multimodal chunks C_q used to construct the multimodal KG $\mathcal{KG}_q$, and query q together into the LLM for final query answering.

3.2 Step 1: Coarse-grain Ranking

We first encode the source multimodal document set $\mathcal{D}$ for retrieval, where $\mathcal{D} = \mathcal{D}_{txt} \cup \mathcal{D}_{img}$, with text document set $\mathcal{D}_{txt}$ and image set $\mathcal{D}_{img}$. Specifically, for textual documents in $\mathcal{D}_{txt}$, we will split them into overlapping text chunks $\mathcal{T}$, with a fixed chunk token length l_c, and an overlapping token length l_o: $\mathcal{T} = \text{Split}(\mathcal{D}_{txt}, l_c, l_o)$, where we set l_c and l_o to 256 and 32, respectively, consistent with the best empirically findings in existing works [24, 25].

We combine all text chunks $\mathcal{T}$ and image documents $\mathcal{D}_{img}$ to form the multimodal chunk set $C = \mathcal{T} \cup \mathcal{D}_{img}$. Each chunk $c \in C$ is encoded with a pretrained encoder $\mathcal{M}$ to obtain its embedding $\overrightarrow{\mathbf{h}_c} \in \mathbb{R}^d$ for coarse-grained ranking: $\overrightarrow{\mathbf{h}_c} = \mathcal{M}(c)$, where d is the embedding dimension. Following [22], we represent images using their captions, enabling unified text-based encoding for both modalities. We use bge-base-en-v1.5 [57] as our encoder. If c is an image ($c \in \mathcal{D}_{img}$), we encode its caption, either provided by the dataset or generated by an LLM, as its embedding.

Given a query q that may consist of both text q_{txt} and image q_{img} modalities, our goal is to rank the most relevant multimodal chunks in $C = \{c_1, c_2, \ldots, c_{|C|}\}$. Each chunk c_j is text or image chunk. For a given query q, if it contains only a single modality (either text or image), we use the corresponding encoder to obtain the query embedding $\overrightarrow{\mathbf{h}_q}$. If q consists of both text and image modalities, we compute its representation $\overrightarrow{\mathbf{h}_q}$ by averaging the embeddings of its textual component and the caption q_{cap} of the querying image:

$$\overrightarrow{\mathbf{h}_q} = \frac{\mathcal{M}(q_{txt}) + \mathcal{M}(q_{cap})}{2}. \quad (3)$$

Next, we evaluate the distance between the query and each chunk using the Euclidean distance in the embedding space:

$$\text{distance}(\overrightarrow{\mathbf{h}_q}, \overrightarrow{\mathbf{h}_c}) = \left\| \overrightarrow{\mathbf{h}_q} - \overrightarrow{\mathbf{h}_c} \right\|_2, \quad (4)$$

where a smaller distance indicates a higher degree of semantic similarity between the query and the chunk. This metric enables an effective comparison across modalities by leveraging the strengths of the multimodal encoder.

With all distance scores computed, we rank the entire set of candidate chunks for each query by sorting them in ascending

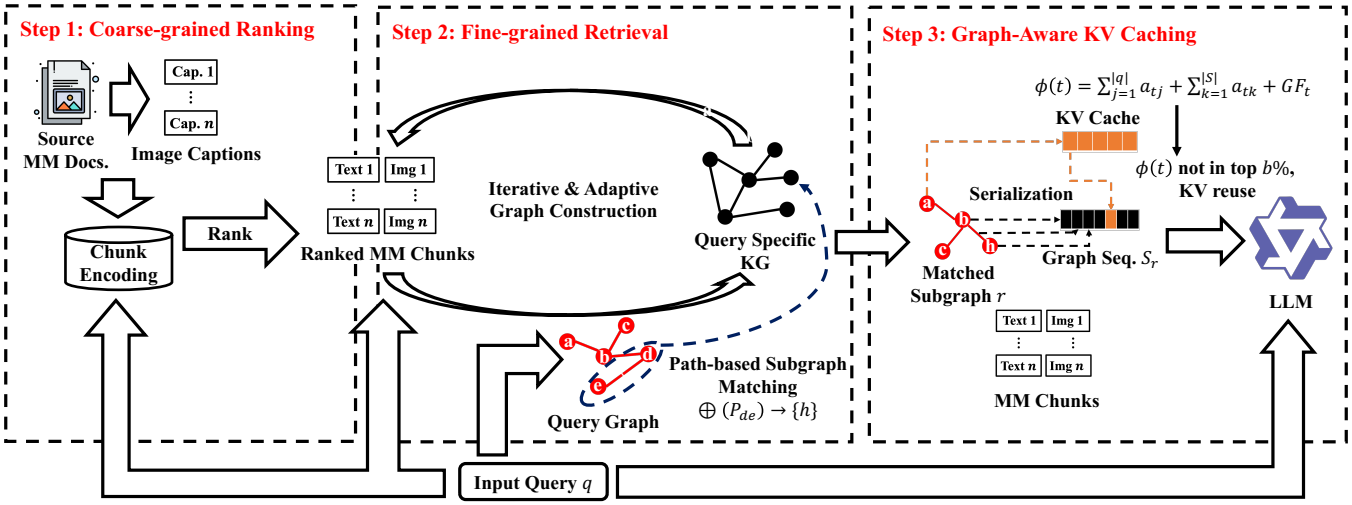


Figure 2: Our MGRAG consists of 3 steps. In Step 1, we rank the multimodal chunks regarding their relevance to the query. In Step 2, we iteratively construct a query-specific KG with the ranked multimodal chunks in a top-down manner. The iterations stop when the query's semantically aligned subgraph is found in KG via a Path-based Subgraph Matching mechanism. In Step 3, We selectively reuse the LLM's KV cache from Step 2, scoring each token t by its topological importance GF_t^q in KG and cross-attention weights a, and only recompute tokens with recompute score $\phi(t)$ in top b%.

order of their Euclidean distance to the query representation, this rank results C_q^{rank} of coarse-grained chunks C for the query q is:

$$C_q^{\text{rank}} = \left(c_i \in C \,\middle|\, i \in \text{argsort}_{\uparrow} \left(\left[\text{distance}(\overrightarrow{\mathbf{h}_q}, \overrightarrow{\mathbf{h}_{c_j}}) \right]_{j=1}^{|C|} \right) \right). \quad (5)$$

3.3 Step 2: Fine-grain Retrieval

Here, we iteratively select chunks c from the query q's ranked chunks C_q^{rank} in descending order of similarity to the query. Each selected chunk is used to extract entities and relations, thereby incrementally enriching the query-specific multimodal KG $\mathcal{KG}_q$. The iteration terminates when our *Path-based Semantic Subgraph Matching* algorithm identifies a matched subgraph $r \subseteq \mathcal{KG}_q$, or reach the maximum iteration μ.

Firstly, we construct the query graph $\mathcal{QG}_q(\hat{\mathcal{V}}_q, \hat{\mathcal{E}}_q)$ regarding the query q. Here, we extract key entities $\hat{\mathcal{V}}_q$ and relations $\hat{\mathcal{E}}_q$ from the query similarly to Equation (6). When extracting entities and relations, as shown in Figure 3, to represent the key missing entities and relations in the query, which are usually highly related to the query answer, we ask the LLM to use placeholders that are semantically close to those entities or relations, such as *Person*, or *Year*, to represent them, and serve as entities or relations on the query graph. These placeholders are later used for calculating semantic similarity with components on the multimodal KG, to retrieve the answer-related information.

Next, we construct or enrich the query-specific KG $\mathcal{KG}_q$. In each iteration, we first apply LLM (e.g., Qwen-2.5-VL [8]) to extract entities $\mathcal{V}_q^c$ and relations $\mathcal{E}_q^c$ from each chunk $c \in C_q^{\text{rank}}$ at the current iteration regarding query q:

$$\mathcal{V}_q^c, \mathcal{E}_q^c = \text{LLM}(c, q, p_{extraction}), \quad (6)$$

where $\text{LLM}(\cdot)$ is the extraction model and $p_{\text{extraction}}$ is the extraction prompt. In p_{ext}, we extract query-related entities and relations as key events from multimodal chunks, enabling query-aware KG construction and addressing the query-agnostic issue. The original multimodal chunks c are added to $\mathcal{V}_q^c$ as passage nodes, yielding the final entity set $\mathcal{V}_q$. We then connect the passage node c to entity v extracted from it with pseudo relation $e_{(c,v)}$, with relation *consists_in*, obtaining the final relation set $\mathcal{E}_q$:

$$\mathcal{V}_q = \bigcup_{c \in C_q^{\text{rank}}} \mathcal{V}_q^c \cup \{c\}, \quad \mathcal{E}_q = \bigcup_{c \in C_q^{\text{rank}}} (\mathcal{E}_q^c \cup \bigcup_{v \in \mathcal{V}_q^c} f(c, v)), \quad (7)$$

where $f(c, v) = \{e(c, \textit{consists\_in}, v)\}$ connects each passage node c to entities v extracted from it. With the entity set $\mathcal{V}_q$ and relation set $\mathcal{E}_q$, we are able to construct the multimodal KG $\mathcal{KG}_q(\mathcal{V}_q, \mathcal{E}_q)$ specific to query q. Then, given an edge $e(u, r, v) \in \mathcal{E}_q$, where u and v are entities and r is the relation, we obtain their representations using a text encoder $\mathcal{M}$ (e.g., BGE [57]) as follows:

$$\overrightarrow{\mathbf{h}}_u = \mathcal{M}(u), \quad \overrightarrow{\mathbf{h}}_v = \mathcal{M}(v), \quad \overrightarrow{\mathbf{h}}_r = \mathcal{M}(r). \quad (8)$$

Also, given two entities u and v, there may exist multiple paths from u to v, denoted as $\mathcal{P}(u, v)$. A single path $P_i(u, v) \in \mathcal{P}(u, v)$ consisting of k consecutive triples is defined as:

$$P_i(u, v) = \big(e_1(u, r_1, z_1),\ e_2(z_1, r_2, z_2),\ \ldots,\ e_k(z_{k-1}, r_k, v)\big),$$

where $z_1, \ldots, z_{k-1}$ are intermediate entities, $r_1, \ldots, r_k$ are the corresponding relations, and the path length is $|P(u, v)| = k$. We define the embedding of path $P(u, v)$ as the aggregation over its relation embeddings:

$$\overrightarrow{\mathbf{h}}_{P_i(u,v)} = \bigoplus_{j=1}^{|P_i(u,v)|} \overrightarrow{\mathbf{h}}_{r_j}, \quad (9)$$

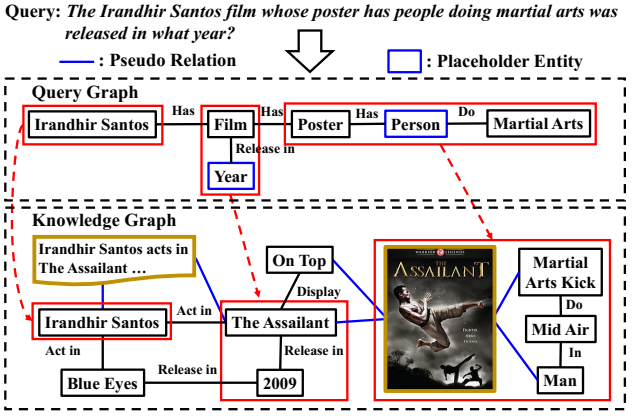


Figure 3: An example of our path-based semantic subgraph matching. The blue edges denote pseudo relations that connect multimodal chunks (passage nodes) to entities extracted from them. The blue boxes denote placeholder entities, representing unknown query knowledge.

where $\overrightarrow{\mathbf{h}}_{r_j}$ is the semantic embedding of relation r_j, and $\bigoplus$ is a customized aggregation function, such as mean pooling.

After the graph construction or enrichment in each iteration, a straightforward approach is to apply a graph matching algorithm based on each node/relation's pairwise semantic similarity. Concretely, given the query graph $\mathcal{QG}_q$, if a matched subgraph can be identified in $\mathcal{KG}_q$, we consider the available knowledge to be sufficient, terminate the iterative graph construction, and return the matched subgraph as the graph context to aid LLM query answering. However, this matching process faces two main challenges. First, as shown in Figure 3, semantically equivalent subgraphs may differ in topology, resulting in a semantic-structural mismatch that standard graph matching cannot resolve. Second, since MGRAG constructs a query-specific KG for each query, it is impractical to train a dedicated subgraph matching model for every KG. This limits the applicability of existing methods [30, 32, 66] in our setting. To address both challenges, we formulate this problem as the *Path-based Semantic Subgraph Matching Problem.*

Definition 2 (Path-based Semantic Subgraph Matching Problem). *Given a query graph $\mathcal{QG}_q(\hat{\mathcal{V}}_q, \hat{\mathcal{E}}_q)$, a multimodal KG $\mathcal{KG}_q(\mathcal{V}_q, \mathcal{E}_q)$, and a similarity threshold $\theta \in [0, 1]$, the Path-based Semantic Subgraph Matching Problem is to find an optimal matched subgraph $\mathcal{SG}(\mathcal{V}_q, \mathcal{E}_q) \subseteq \mathcal{KG}_q$, induced by a mapping $I : \hat{\mathcal{V}}_q \to \mathcal{V}_q$, to maximize the overall similarity:*

$$\max \ Sim(\mathcal{SG}) = \sum_{e(u,r,v)\in\hat{\mathcal{E}}_q} \max_{P\in\mathcal{P}(I(u),I(v))} \cos\left(\overrightarrow{\mathbf{h}}_r, \overrightarrow{\mathbf{h}}_P\right) \quad (10)$$

$$\text{s.t.} \begin{cases} \cos\left(\overrightarrow{\mathbf{h}}_u, \overrightarrow{\mathbf{h}}_{I(u)}\right) > \theta, \cos\left(\overrightarrow{\mathbf{h}}_v, \overrightarrow{\mathbf{h}}_{I(v)}\right) > \theta, & \forall u, v \in \hat{\mathcal{V}}_q \\ \max_{P\in\mathcal{P}(I(u),I(v))} \cos\left(\overrightarrow{\mathbf{h}}_r, \overrightarrow{\mathbf{h}}_P\right) > \theta, & \forall e(u,r,v) \in \hat{\mathcal{E}}_q \end{cases}$$

where $\overrightarrow{\mathbf{h}}_P$ is the aggregated path embedding for $P \in \mathcal{P}(I(u), I(v))$ as defined in Equation (9)*. If no such subgraph exists, return $\varnothing$.*

Algorithm 1: Path-based Semantic Subgraph Matching

Input: Query graph $\mathcal{QG}_q(\hat{\mathcal{V}}_q, \hat{\mathcal{E}}_q)$, anchor entity $\gamma \in \hat{\mathcal{V}}_q$, multimodal KG $\mathcal{KG}_q(\mathcal{V}_q, \mathcal{E}_q)$, threshold θ
Output: Highest-scoring matched subgraph $r \subseteq \mathcal{KG}_q$ or $\varnothing$

$\mathcal{R} \leftarrow \emptyset$;
for *each* $v \in \mathcal{V}_q$ *with* $\cos(\overrightarrow{\mathbf{h}}_v, \overrightarrow{\mathbf{h}}_\gamma) > \theta$ **do**
  $\mathcal{I} \leftarrow$ SubgraphIsomorphism$(\mathcal{QG}_q, \mathcal{KG}_q, v)$;
  for *each mapping* $I \in \mathcal{I}$ **do**
    valid $\leftarrow$ true, $s \leftarrow 0$;
    $\mathcal{V}_{\text{sub}} \leftarrow \{I(x) \mid x \in \hat{\mathcal{V}}_q\}$;
    $\mathcal{E}_{\text{sub}} \leftarrow \emptyset$;
    for *each* $\hat{e}(u, \hat{r}, v) \in \hat{\mathcal{E}}_q$ **do**
      (found, P) $\leftarrow$ GreedyPathSearch$(I(u), I(v), \overrightarrow{\mathbf{h}}_{e(u,r,v)}, \mathcal{KG}_q, \theta)$;
      if *not found* **then**
        valid $\leftarrow$ false;
        break;
      $s \leftarrow s + \cos(\overrightarrow{\mathbf{h}}_P, \overrightarrow{\mathbf{h}}_{e(u,r,v)})$;
      $\mathcal{E}_{\text{sub}} \leftarrow \mathcal{E}_{\text{sub}} \cup \{e \in P\}$;
      $\mathcal{V}_{\text{sub}} \leftarrow \mathcal{V}_{\text{sub}} \cup \{\text{nodes in } P\}$;
    if *valid* **then**
      $s \leftarrow s/|\hat{\mathcal{E}}_q|$;
      $\Lambda \leftarrow (\mathcal{V}_{\text{sub}}, \mathcal{E}_{\text{sub}})$;
      $\mathcal{R} \leftarrow \mathcal{R} \cup \{(\Lambda, s)\}$;
return top-scoring subgraph $r \in \mathcal{R}$, or $\varnothing$ if $\mathcal{R} = \emptyset$;

Theorem 1. *The Path-based Semantic Subgraph Matching Problem is NP-hard.*

Proof Sketch. We establish NP-hardness through a polynomial-time reduction from the Subgraph Isomorphism problem by constructing an instance with uniform embedding vectors and unit path lengths. Due to the space limitation, we put our proof in our technical report [1]. □

Since this problem is NP-hard, it is infeasible to obtain the optimal result in polynomial time. Therefore, to solve this problem, we propose an efficient greedy algorithm based on semantic path growing and subgraph isomorphism enumeration, as shown in Algorithm 1, which can find a subgraph within the multimodal KG that semantically matches the query graph, but not necessarily identical in graph topology. Given a query graph $\mathcal{QG}_q$, we first select its anchor entity γ to the multimodal KG $\mathcal{KG}_q$, specifically, we compute the pairwise cosine similarities between the embeddings of all entities in $\mathcal{QG}_q$ and $\mathcal{KG}_q$, and select the pair (γ, v) with the highest similarity, where $\gamma \in \hat{\mathcal{V}}_q$ and $v \in \mathcal{V}_q$. If the algorithm cannot find a valid match starting with γ, it will try other anchor entities until a valid match is found. After that, we consider γ as the anchor entity in the query graph and v as its candidate match entity in the multimodal KG, and calculate the *Path-based Subgraph Isomorphism*. In this paper, we substitute the VF2 [13] algorithm's exact matching mechanism with similarity-based matching. This

similarity-based matching uses a similarity threshold θ to calculate the Path-based Subgraph Isomorphism in line 3 of Algorithm 1.

Then, for each subgraph isomorphism $I : \hat{\mathcal{V}}_q \to \mathcal{V}_q$ that satisfies $I(\gamma) = v$, we try to find a matched subgraph edge by edge. To begin with, for every relation $\hat{e}(u, \hat{r}, v)$ in the relation set $\hat{\mathcal{E}}_q$ of the query graph $\mathcal{QG}_q$, we grow a path in the multimodal KG $\mathcal{KG}_q$ starting from entity u's isomorphic node $I(u)$ in the multimodal KG by greedily selecting entities x within the neighbor entity set $\mathcal{N}(P)$ of the currently generated path P on the multimodal KG.

Then, we greedily expand the path P by evaluating the neighbors of the current end node x of path P. We select the neighbor entity $y \in \mathcal{N}(x)$ and its connecting relation $e(x, r, y) \in \mathcal{E}_q$ that maximize the cosine similarity between the newly aggregated path embedding and the edge embedding $\overrightarrow{\mathbf{h}}_{\hat{e}(u,\hat{r},v)}$ of the query relation $\hat{e}(u, \hat{r}, v)$:

$$\arg \max_{y \in \mathcal{N}(x), e(x,r,y) \in \mathcal{E}_q} \cos\left(\overrightarrow{\mathbf{h}}_P \oplus \overrightarrow{\mathbf{h}}_{e(x,r,y)} \oplus \overrightarrow{\mathbf{h}}_y, \overrightarrow{\mathbf{h}}_{\hat{e}(u,\hat{r},v)}\right) \quad (11)$$

where $\bigoplus$ denotes the path aggregation function defined in Equation (9). Once the optimal relation $e(x, r, y)$ are identified, we append them to the current path P:

$$P = P \circ e(x, r, y), \quad (12)$$

where $\circ$ denotes the path concatenation function, and update $x \leftarrow y$ to continue the path expansion from the new endpoint. The path expansion continues until the path reaches the target node $I(v)$, or terminates if no neighbor yields an aggregated cosine similarity above θ. In our experiments, for efficiency, we set a maximum path length l_p to prevent excessively long paths.

The path expansion terminates when no qualifying neighbor remains. The aggregated path embedding is then validated against the query edge embedding; the mapping is accepted only when all query edges pass. Valid matches are added to $\mathcal{R}$, and the highest-scoring subgraph $r \in \mathcal{R}$ is returned. If $\mathcal{R} = \emptyset$, the next chunk in C_q^{rank} is consumed to enrich $\mathcal{KG}_q$.

Next, we prove our Path-based Semantic Subgraph Matching algorithm has an approximation ratio bounded by the similarity threshold θ via ∞-Wasserstein Bottleneck [10].

Theorem 2. *The Algorithms 1 achieve a θ-bounded quality guarantee for the Path-based Semantic Subgraph Matching Problem: every edge in the matched subgraph has cosine similarity at least θ, if a valid matched subgraph is found.*

Proof Sketch. By framing the graph matching process as a constrained optimal transport problem, we show that the local thresholding in our greedy algorithm bounds the ∞-Wasserstein (bottleneck) distance and achieves θ-approximation. Due to the space limitation, we put our proof in our technical report at [1]. □

After obtaining the matched subgraph r, to convert the graph into an LLM-compatible string format, we serialize it into a string sequence S_r of triplet facts, with [SEP] as the split marker between different facts:

$$S_r = e(u_1, r_1, v_1) \,||\, [\text{SEP}] \cdots [\text{SEP}] \,||\, e(u_n, r_n, v_n), \quad (13)$$

where $u_1 \cdots u_n, v_1 \cdots v_n \in \mathcal{V}_q$ denote entity nodes in graph r, $e(u_1, r_1, v_1) \cdots e(u_n, r_n, v_n) \in \mathcal{E}_q$ denotes relation edges in graph r, $||$ denotes the string concatenation operation.

Algorithm 2: Greedy Path Search (GreedyPathSearch)

Input: Start node v_{start}, target node v_{target}, target joint embedding $\overrightarrow{\mathbf{h}}_{\text{target}}$, data graph $\mathcal{KG}_q(\mathcal{V}_q, \mathcal{E}_q)$, similarity threshold θ

Output: Tuple (found, P) where found is Boolean and P is the matched path

$P \leftarrow (v_{\text{start}})$, $x \leftarrow v_{\text{start}}$;
while $x \neq v_{target}$ **do**
  $\mathcal{N}_{\text{valid}} \leftarrow \emptyset$;
  for *each neighbor* $y \in \mathcal{N}(x)$ *and edge* $e(x, r, y) \in \mathcal{E}_q$ **do**
    $\overrightarrow{\mathbf{h}}_{\text{tmp}} \leftarrow \overrightarrow{\mathbf{h}}_P \oplus \overrightarrow{\mathbf{h}}_{e(x,r,y)} \oplus \overrightarrow{\mathbf{h}}_y$;
    if $\cos(\overrightarrow{\mathbf{h}}_{tmp}, \overrightarrow{\mathbf{h}}_{target}) > \theta$ **then**
      $\mathcal{N}_{\text{valid}} \leftarrow \mathcal{N}_{\text{valid}} \cup \{(y, e(x, r, y), \cos(\overrightarrow{\mathbf{h}}_{\text{tmp}}, \overrightarrow{\mathbf{h}}_{\text{target}}))\}$;
  if $\mathcal{N}_{valid} == \emptyset$ **then**
    return (false, $\emptyset$);
  $(y^*, e^*, \text{score}) \leftarrow \arg\max_{(y,e,s) \in \mathcal{N}_{\text{valid}}} s$;
  $P \leftarrow P \circ e^*$;
  $x \leftarrow y^*$;
if $x == v_{target}$ **then**
  return (true, P);
else
  return (false, $\emptyset$);

In Step 2, the entities and relations of graph r are all extracted by the LLM to form the query-specific KG $\mathcal{KG}_q$, and will be input to the LLM again in graph sequence S_r for query answering. According to Figure 4, the green token sequence was input to the LLM for graph construction, and can be partially reused by tokens in the blue graph sequence $\mathcal{S}_r$, as both of them are tokens of entities and relations in the KG. Hence, we propose a Graph-Aware KV Caching mechanism in Step 3, to better reflect the importance of these graph components' respective tokens, regarding the KG's topological structure.

3.4 Step 3: Graph-Aware KV Caching

Here, the LLM KV cache is stored in a fixed-size memory, and we evict them with the LRU (Least Recently Used) manner. Based on the cache recompute score $\phi(\cdot)$, we will only selectively recompute the $b\%$ most important tokens' LLM KV matrices, and reuse other tokens' LLM KV matrices that were stored in Step 2. To be specific, we calculate the cache recompute score $\phi(t)$ for each token t regarding three components: (1). Its topological score GF_t^q regarding its respective entity or relation's topological feature in the query q specific multimodal KG $\mathcal{KG}_q$. (2). Its cross attention score $\sum_{j=1}^{|q|} a_{tj}$ with tokens of query q. (3). Its cross attention score $\sum_{k=1}^{|S_r|} a_{tk}$ with tokens of other entity and relation in $\mathcal{KG}_q$.

When processing images, the key events in LLM's image inputs can be fragmented, making the cross-attention score applied by current KV caching mechanisms unable to reflect the importance of image tokens. Hence, we construct a KG $\mathcal{KG}_q$ to contain the query-related events. As a result, the topological information of $\mathcal{KG}_q$ can

be crucial for reflecting the importance of image tokens regarding certain events. Hence, the first part of our cache recompute score $\phi(\cdot)$ is the graph's topological feature score GF_t^q of token t. As the $\mathcal{KG}_q$ is constructed with the entities and relations that adaptively extracted from multimodal chunks related to query q, they are usually small graphs with an average of fewer than 20 entities (see our technical report at [1]).

As a result, random walk-based graph scoring algorithms such as PageRank [58] are prone to overfitting and result in suboptimal performance. Consequently, we select the four most commonly used graph statistical features to reflect the topological importance of graph components. Here, we denote $\mathcal{V}_q$ and $\mathcal{E}_q$ as the entity and relation set of multimodal KG $\mathcal{KG}_q$, $\mathcal{N}(\cdot)$ denotes the neighbor node set of a node or a subgraph on $\mathcal{KG}_q$:

- **Degree Centrality** (dc): For $v \in \mathcal{V}_q$, degree centrality is $dc(v) = \frac{\deg(v)}{|\mathcal{V}_q|-1}$ where $\deg(v)$ is the degree of v in the multimodal KG $\mathcal{KG}_q$. Higher $dc(v)$ means greater topological importance.
- **Betweenness Centrality** (bc): For any $v \in \mathcal{V}_q$, we define its betweenness centrality $bc(v)$ as:

$$bc(v) = \frac{2}{(|\mathcal{V}_q|-1)(|\mathcal{V}_q|-2)} \sum_{x \neq y \neq v, x, y \in \mathcal{V}_q} \frac{|P^v_{(x,y)}|}{|P_{(x,y)}|},$$

  Here, $P_{(x,y)}$ is the set of all paths from node x to y, and $P^v_{(x,y)} \subseteq P_{(x,y)}$ is the subset passing through node v. This metric measures a node's control over information flow and its role as a bridge in the network; higher values indicate greater importance for graph connectivity.
- **Eigenvector Centrality** (ec): Let $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_n)^\top$ be the principal eigenvector of the adjacency matrix of $\mathcal{KG}_q$, and let λ_v denote its component for node v. The eigenvector centrality is: $ec(v) = \frac{\lambda_v}{\max(\lambda)}$. Unlike degree centrality, $ec(v)$ reflects both direct and indirect influence, recursively rewarding connections to highly central neighbors.
- **Closeness Centrality** (cc): For $v \in \mathcal{V}_q$, $cc(v) = \frac{|\mathcal{V}_q|-1}{\sum_{u \in \mathcal{V}_q, u \neq v} \text{dist}(u,v)}$, where $\text{dist}(u,v)$ is the shortest path length between u and v. Higher $cc(v)$ means v is more centrally located in the graph.

With all these feature scores, we denote their average as the graph topology $f(u)$ score for node u: $f(u) = \frac{dc(u)+bc(u)+ec(u)+cc(u)}{4}$. After obtaining the graph topology score $f(\cdot)$ for each graph node, we map each token t to triplet facts. Here, we denote $(u, e_{(u,v)}, v)$ as the respective triplet fact that token t involved in, and map t to the triplet fact as follow: $t \in \psi(u)||\psi(e_{(u,v)})||\psi(v)$, where $\psi(\cdot)$ denotes the mapping function from entity or relation to their respective tokens, $||$ denotes the token concatenation operation, entity nodes $u, v \in \mathcal{V}_q$, relation edges $e_{(u,v)} \in \mathcal{E}_q$. Then token t's graph topological feature GF_t^q is calculated as the average topological feature score of the two nodes involved in its respective triplet fact: $GF_t^q = \frac{f(u)+f(v)}{2}$. Similar to CacheBlend [59], we compute cross-attention scores for our KV cache: specifically, the cross-attention between token t and the KV cache of query q, as well as between t and the KV cache of other tokens.

After obtaining the matched subgraph r's graph sequence S_r, we compute the cross attention score a in a same manner with CacheBlend [59], and obtain the final KV recompute score $\phi(t)$ of

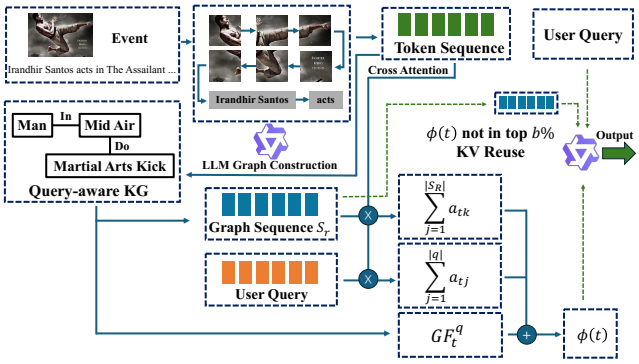


Figure 4: An example of our event-agnostic KV recomputation, where we utilize the graph topological features of the event KG as supplementary knowledge to help mitigate the effects of attention calculation, making the KV Cache Score better reflect the original importance of image tokens.

token t in query q as:

$$\phi(t) = \sum_{j=1}^{|q|} a_{tj} + \sum_{k=1}^{|S_r|} a_{tk} + GF_t^q, \tag{14}$$

where $|q|$ and $|S_r|$ denotes the number of tokens in query q and graph sequence S_r, respectively, and a_{tj} denotes the cross attention score between token t and token j.

3.5 Complexity Analysis

Denote the multimodal chunk set as $|C|$, the chunk length as l_c, the embedding dimension as d_{emb}, the node set of query graph and multimodal KG as $\hat{\mathcal{V}}$ and $\mathcal{V}$, the number of edges in query graph as $\hat{\mathcal{E}}$, the maximum path context length as l_p, the query token length as l_q, the number of chunks used for graph construction as k, the LLM's hidden states as d_{LLM}, the number of nodes and edges in multimodal KG as $\mathcal{V}$ and $\mathcal{E}$, the token length of the matched subgraph sequence and query as $|S_r|$ and $|q|$. The total offline time complexity of MGRAG is:

$$O(|C| d_{emb} l_c^2),$$

and online time complexity is:

$$O(d_{emb} k l_c^2 + l_p |\hat{\mathcal{E}}| |\mathcal{V}|^{|\hat{\mathcal{V}}|+1} + |\mathcal{V}||\mathcal{E}| + d_{LLM}(|S_r|^2 + |q|^2)).$$

Although the time complexity is exponential in the worst case, the average number of nodes and edges in the query graph and multimodal KG is less than 20, so MGRAG's time cost remains comparable to other baselines. Detailed complexity analysis is available in our technical report at [1].

4 EXPERIMENTS

In this section, we outline our experimental setup in Section 4.1, including details on datasets, and evaluation metrics, and report our experimental results in Section 4.2 and Section 4.3 for effectiveness and efficiency, respectively. At last, we conduct the ablation study in Section 4.6 and parameter sensitivity study in Section 4.7.

4.1 Experiment Settings

4.1.1 ***Datasets.*** We select the Arxiv, Manual, Web, Wiki, Wit, and Recipe dataset from the MRAMG benchmark [60], as well as the MMQA dataset [50] to test MGRAG's performance. The dataset statistics are shown in Table 3, where the MMQA dataset has a large number of source documents, and its queries can consist of image and text pairs, making it a challenging dataset for evaluating RAG models. The MRAMG benchmark is the most up-to-date benchmark of the multimodal QA task, and its six datasets are built upon source data from various domains, compared to the MMQA dataset, it has a much smaller number of source documents, and only consists of textual queries.

4.1.2 ***Evaluation Metrics.*** We employ four metrics to evaluate the performance of MGRAG and baselines via the RAGAS evaluation framework [16], as follows:

(1) **Semantic Similarity (SIM.)**: Measures the vector's cosine similarity between the generated answer and the gold answer, we apply the bge-base-en-v1.5 [57] for vector encoding.
(2) **Accuracy (ACC.)**: Measures the proportion of correct answers among all answers generated by a model.
(3) **ROUGE-L (ROG.)** [39]: Evaluates the quality of generated text by comparing the longest common subsequence (LCS) between the generated answer G and the ground-truth answer Y.
(4) **BLEU-1 (BLU.)** [44]: Measures the quality of the generated answer G by computing the modified unigram precision with respect to the ground-truth answer Y.

4.1.3 ***Baselines.*** As shown in Table 4 and Table 7, we compare MGRAG against a broad set of retrieval, subgraph matching, KG construction, and KV cache baselines.

- **RAG Systems**: We compare MGRAG's performance with 5 RAG systems in Table 4. Among these systems, VanillaRAG [20] is a similarity-based RAG system that retrieves top-k semantic-nearest chunks with a dense encoder and directly feeds them to the LLM. R1-Searcher [48] fine-tunes the retrieval and reasoning pipeline of VanillaRAG with reinforcement learning to improve multi-step question answering. GraphRAG [15] constructs a global knowledge graph from the corpus, performs community detection, and generates community summaries as graph-level contexts for QA. LightRAG [23] simplifies GraphRAG by skipping community construction and instead retrieving high-scoring nodes and their neighborhoods to reduce retrieval cost. HippoRAG2 [25] further improves graph-based retrieval by indexing full passages as nodes and using Personalized PageRank to select a fixed number of text or image chunks.
- **KG Construction methods**: We compare MGRAG's performance with 2 KG construction methods in Table 7. Stanford OpenIE (O.IE) [7] extracts entities and relations using pattern-based information extraction without LLMs, while STAT [56] builds KGs via term-frequency based statistical entity extraction.
- **Subgraph Matching methods**: We compare MGRAG's performance with 9 subgraph matching methods in Table 7. Among these systems, VF2 [13] is a classic exact subgraph-isomorphism solver based on depth-first backtracking with strong feasibility pruning, and ISMAGS [27] extends VF2 with symmetry-aware search to avoid redundant isomorphic embeddings. SM [34] relaxes discrete matching to a spectral optimization over an affinity matrix, VJ [17] initializes from seeded correspondences and iteratively expands them, RRWM [12] performs reweighted random walks on the association graph to obtain robust soft correspondences, and IPFP [35] solves a quadratic assignment relaxation via an integer-projected fixed-point iteration. BGE [57] measures graph similarity in a learned embedding space by encoding entities and relations into dense vectors, SLOTAlign [51] jointly refines graph structure and computes an optimal-transport based alignment between attributed graphs, and FGWAlign [52] uses a fused Gromov-Wasserstein distance to couple node attributes and topology for graph edit distance style alignment.
- **KV Caching mechanisms**: We compare MGRAG's performance with 2 KV Cache mechanisms in Table 6. vLLM [33] provides prefix-based paged-attention KV caching to accelerate decoding, while CacheBlend [59] is a position-independent method that scores tokens by cross-attention importance and recomputes only the most salient ones without exploiting graph structure.

Table 3: The statistic of MMQA [50], and 7 sub dataset of the MRAMG [60] benchmark. T and I in Query Modality denotes the dataset contains textual and image queries, respectively.

Statistic	MMQA	Wiki	Wit	Recipe	Arxiv	Manual	Web
Query Modality	T / I	T	T	T	T	T	T
Queries	951	500	600	2,360	200	390	750
Texts	218,285	538	639	1,528	101	40	1,500
Images	57,058	538	639	8,569	337	2,607	1,500
Corpus Size	22.5M	680k	478k	1.1M	136k	420k	215k
Query Hops	1-2	2-5	2-4	1-6	3-4	2-6	2-3
Avg Query Hops	1.5	3.0	2.8	3.3	3.1	3.7	2.4

For VanillaRAG, we implement it with the langchain framework and FAISS vector database [14]. For GraphRAG, we use the implementation from [2] which optimizes its original code and achieves better time efficiency while not affect the performance; For all other baselines, we use their official implementations. For all similarity-based retrieval scenarios, we encode all images' captions to represent them for similarity based retrieval as in [22], for achieving better performance. We deploy the Qwen-2.5-VL-7B-Instruct backbone with vLLM [33], and use vLLM [33] to accelerate all tested baselines other than MGRAG.

4.1.4 ***Hyperparameter and Hardware Settings.*** We set the retrieval top-k value for all RAG models to 5, aligning with the settings of HippoRAG2 [25]. We apply the hybrid search and local model for LightRAG [23] and GraphRAG [15], respectively, as they are the most complete search mode and achieve best performance. We apply LightRAG's local search mode for the text classification task, as it achieves the best effectiveness on the two datasets tested. For HippoRAG2 [25], we set its hyperparameters to default, aligning with its original paper and code. For our MGRAG, the text chunks' length is set to 256, and token overlap between chunks to 32, to align with the setting of HippoRAG2. All our experiments were carried out and evaluated with Qwen2.5-7B-VL-Instruct [8] LLM backbone, as Qwen is the best open-sourced multimodal LLM model according to huggingface LLM-leaderboard [28] We apply the Contriever

Table 4: The experiment result on MMQA [50], and 7 sub dataset of the MRAMG [60] benchmark, all experiments were carried out with Qwen2.5-VL-7B-Instruct backbone. "-" denotes the experiment cost longer than 3 days.

Dataset	Metric	Similarity-based		Graph-based			
		Vanilla.	R1.	Graph.	Light.	Hippo.	**MGRAG**
MMQA	SIM.	0.8362	0.5478	-	-	0.8381	**0.8533**
	ACC.	0.7053	0.3567	-	-	0.6335	**0.7442**
	ROG.	0.5942	0.1041	-	-	0.6659	**0.6763**
	BLEU	0.6645	0.0817	-	-	0.5917	**0.7215**
Arxiv	SIM.	0.8664	0.7051	0.7943	0.7404	0.8293	**0.8760**
	ACC.	0.6638	0.3775	0.2350	0.3813	0.4688	**0.6800**
	ROG.	0.2713	0.1189	0.1553	0.1580	0.2331	**0.2726**
	BLEU	0.2452	0.1092	0.1690	0.1370	0.1281	**0.2498**
Manual	SIM.	0.8166	0.6046	0.5453	0.7255	0.6852	**0.8221**
	ACC.	0.5881	0.2071	0.1436	0.4096	0.2641	**0.6173**
	ROG.	0.3509	0.1078	0.0974	0.1595	0.2047	**0.3839**
	BLEU	0.2767	0.0740	0.0718	0.1385	0.1314	**0.3093**
Web	SIM.	0.6937	0.6620	0.5470	0.5513	0.5642	**0.7009**
	ACC.	0.4703	0.4200	0.1053	0.2000	0.3607	**0.5123**
	ROG.	0.1508	0.1142	0.1133	0.0905	0.0616	**0.1664**
	BLEU	0.0326	0.0785	0.1031	0.0206	0.0133	**0.1456**
Wiki	SIM.	0.8824	0.5478	0.6966	0.7517	0.8204	**0.8838**
	ACC.	0.7800	0.3567	0.2000	0.3660	0.6500	**0.7845**
	ROG.	0.3559	0.1418	0.1361	0.2008	0.3042	**0.3579**
	BLEU	0.3053	0.1146	0.1007	0.1863	0.1851	**0.3174**
Wit	SIM.	0.7851	0.6110	0.6211	0.6973	0.7230	**0.8033**
	ACC.	0.6675	0.6017	0.1500	0.2483	0.5138	**0.6967**
	ROG.	0.3255	0.1307	0.1319	0.2416	0.2712	**0.3589**
	BLEU	0.2095	0.0934	0.1142	0.1928	0.1200	**0.2382**
Recipe	SIM.	0.8599	0.6155	0.8166	0.8118	0.7792	**0.8673**
	ACC.	0.6525	0.2180	0.2771	0.3334	0.4917	**0.6814**
	ROG.	0.3714	0.1065	0.1745	0.2128	0.3303	**0.4220**
	BLEU	0.2599	0.0773	0.1527	0.1648	0.1941	**0.3102**

[29] embedding model for HippoRAG2's dense passage retrieval procedure, and bge-base-en-v1.5 [57] embedding model for other baselines' LLM embedding, align with HippoRAG2's [25] setting. All experiments are conducted on an Intel(R) Xeon(R) Gold 5220R @ 2.20GHz CPU and a single NVIDIA A100-SXM4-40GB GPU. Other hyperparameters of models were set as default by their codes.

4.2 Effectiveness Evaluation

As shown in Table 4 and Table 6, MGRAG achieves the overall best performance over all 7 datasets, surpasses all tested baselines. Based on the experiment results, we can have the following observations:

Firstly, compared to the previous state-of-the-art systems, MGRAG achieves a larger effectiveness improvement (5.5% on ACC.) on the large MMQA dataset than on other datasets. This is because, large external multimodal sources means a large amount of noise for each retrieval, and an accurate and query-aware graph construction is crucially important in this scenario.

Secondly, although lacking the ability of fine-grained retrieval, the Similarity-based RAG systems can still achieve comparable or better performance on certain tested datasets. This is because the

Table 5: MGRAG's breakdown efficiency for each module. Where KGC denotes multimodal KG construction time (in minutes) and token cost, Match denotes subgraph matching time cost, QA denotes LLM QA time and token cost, and E2E denotes end-to-end time and token cost.

Dataset	# Query	KGC	Match	QA	E2E
MMQA	951	193.0 / 3,085	0.14	8.9 / 1,431	248.1 / 4,516
Arxiv	200	61.9 / 3,946	0.23	16.2 / 2,944	83.9 / 6,890
Manual	390	76.1 / 4,285	0.04	23.9 / 2,556	109.7 / 6,841
Web	750	143.8 / 4,623	0.27	42.9 / 3,101	196.5 / 7,723
Wiki	500	164.2 / 6,629	0.30	52.1 / 6,267	241.5 / 12,896
Wit	600	140.6 / 7,660	0.21	42.6 / 6,539	196.5 / 14,199
Recipe	2,360	525.7 / 4,351	1.74	191.3 / 3,173	808.7 / 7,525

Table 6: Efficiency and effectiveness comparison between MGRAG and other KV cache mechanisms on MMQA dataset. Where V. and H. denote these systems cope with similarity-based VanillaRAG and graph-based HippoRAG2 framework.

System	SIM.	ACC.	ROG.	BLEU	TTFT	Time Cost
vLLM (V.)	0.8315	0.6593	0.6277	0.5867	3.2 s	4.5 h
CacheBlend (V.)	0.8315	0.6601	0.6417	0.5964	**3.1 s**	4.5 h
vLLM (H.)	0.8381	0.6335	0.6659	0.5917	4.6 s	8.8 h
CacheBlend (H.)	0.8415	0.6267	0.6636	0.5917	4.4 s	8.3 h
MGRAG	**0.8533**	**0.7442**	**0.6763**	**0.7215**	3.2 s	**4.3 h**

Graph-based RAG baselines' graph construction steps are query-agnostic, which limits their ability to utilize the fine-grained multimodal source information, leading to high information loss. Consequently, as they lose the fine-grained multimodal information early in the graph construction step, their later graph retrieval step is also affected, leading to suboptimal performance. For our MGRAG, the graphs constructed are smaller but query-aware and more tailored for the respective query, enabling the system to better utilize the fine-grained multimodal source information.

Lastly, according to the experiment results on different hop queries of MRAMG benchmark in Table 8, MGRAG achieves better performance on long-hop queries. This demonstrates that MGRAG's subgraph matching mechanism can be redundant and costly when the query only requires simple retrieval, and crucial for handling complex queries.

4.3 Efficiency Evaluation

We evaluate the efficiency of MGRAG in comparison to both similarity-based and graph-based RAG baselines, using the challenging MMQA dataset. Results are summarized in Table 6.

We add fine-grained breakdown time and token cost statistics of MGRAG's each parts in Table 5. According to the results, the repeated LLM calls for entity and relation extraction is the key efficiency bottleneck of our MGRAG model. But in return, by adaptively building compact, relevant knowledge graphs using only the most pertinent multimodal chunks, MGRAG avoids the costly and redundant graph expansion that plagues other graph-based approaches and achieves better effectiveness.

4.4 Subgraph Matching Experiments

Table 7: Experiments of MGRAG with 2 multimodal KG construction and 5 subgraph matching baselines, all experiments were carried out with Qwen2.5-VL-7B-Instruct backbone.

Dataset	Metric	MMKGC		Subgraph Matching									Ours
		Non LLM		Exact Match		Approximate Match				Semantic Match			
		O.IE	STAT	VF2	ISMAGS	SM	VJ	RRWM	IPFP	BGE	SLOT	FGW	
MMQA	SIM.	0.642	0.655	0.851	0.833	0.832	0.812	0.838	0.773	0.832	0.838	0.837	**0.853**
	ACC.	0.571	0.602	0.735	0.706	0.702	0.715	0.698	0.633	0.703	0.700	0.697	**0.744**
	ROG.	0.510	0.519	0.673	0.638	0.632	0.646	0.647	0.601	0.635	0.650	0.647	**0.676**
	BLEU	0.483	0.500	0.620	0.590	0.582	0.606	0.599	0.585	0.586	0.602	0.598	**0.722**
Arxiv	SIM.	0.618	0.640	0.871	0.868	0.866	0.872	0.874	0.814	0.874	0.872	0.868	**0.876**
	ACC.	0.526	0.576	0.650	0.643	0.670	0.667	0.675	0.581	0.668	0.653	0.651	**0.680**
	ROG.	0.177	0.184	0.270	0.266	0.263	0.269	0.257	0.212	0.270	0.269	0.264	**0.273**
	BLEU	0.138	0.145	0.239	0.231	0.244	0.245	0.243	0.203	0.244	0.233	0.230	**0.250**
Manual	SIM.	0.520	0.509	0.817	0.817	0.813	0.808	0.811	0.800	0.815	0.813	0.820	**0.822**
	ACC.	0.459	0.443	0.612	0.612	0.614	0.608	0.614	0.594	0.615	0.611	0.600	**0.617**
	ROG.	0.245	0.234	0.377	0.382	0.378	0.372	0.377	0.369	0.375	0.375	0.374	**0.384**
	BLEU	0.209	0.182	0.306	0.307	0.303	0.298	0.303	0.289	0.303	0.301	0.292	**0.309**
Web	SIM.	0.466	0.485	0.677	0.675	0.676	0.679	0.675	0.666	0.687	0.676	0.676	**0.701**
	ACC.	0.290	0.298	0.507	0.509	0.507	0.508	0.503	0.493	0.510	0.498	0.480	**0.512**
	ROG.	0.052	0.101	0.156	0.155	0.156	0.160	0.157	0.149	0.156	0.154	0.147	**0.166**
	BLEU	0.020	0.095	0.088	0.079	0.056	0.056	0.066	0.055	0.095	0.116	0.096	**0.146**
Wiki	SIM.	0.504	0.525	0.877	0.879	0.878	0.873	0.878	0.869	0.878	0.879	0.878	**0.884**
	ACC.	0.498	0.510	0.752	0.754	0.750	0.751	0.743	0.735	0.746	0.749	0.747	**0.785**
	ROG.	0.230	0.253	0.352	0.355	0.353	0.349	0.355	0.348	0.355	0.356	0.352	**0.358**
	BLEU	0.200	0.218	0.300	0.303	0.298	0.297	0.301	0.296	0.301	0.301	0.300	**0.317**
Wit	SIM.	0.492	0.489	0.799	0.797	0.784	0.786	0.795	0.792	0.789	0.793	0.794	**0.803**
	ACC.	0.465	0.453	0.693	0.694	0.686	0.691	0.679	0.678	0.689	0.693	0.691	**0.697**
	ROG.	0.233	0.205	0.354	0.355	0.353	0.356	0.347	0.346	0.356	0.351	0.349	**0.359**
	BLEU	0.210	0.198	0.234	0.231	0.229	0.227	0.230	0.231	0.230	0.227	0.223	**0.238**
Recipe	SIM.	0.551	0.545	0.859	0.861	0.862	0.857	0.852	0.847	0.864	0.861	0.860	**0.867**
	ACC.	0.425	0.401	0.679	0.668	0.678	0.672	0.670	0.661	0.674	0.675	0.677	**0.681**
	ROG.	0.211	0.198	0.413	0.407	0.411	0.413	0.406	0.395	0.415	0.412	0.409	**0.422**
	BLEU	0.202	0.172	0.304	0.302	0.305	0.295	0.298	0.221	0.308	0.306	0.305	**0.310**

To compare MGRAG's path-based semantic subgraph matching method with other applicable subgraph matching methods, we conduct extensive experiments on MMQA dataset in Table 7. As the query graph and data graph are all query specific in our subgraph matching scheme, training-based methods are not applicable, hence we only select exact matching algorithms VF2 [13], and ISMAGS [27], the approximate matching algorithms SM [34], VJ [17], RRWM [12], and IPFP [35], and the semantic matching algorithms BGE [57], SLOTAlign [51], FGWAlign [52] for comparison. Based on the results in Table 7, we can have the following observations:

Firstly, exact matching methods achieve suboptimal performance, demonstrating that they ignores the semantic correlation between the query and the KG, which leads to inaccurate graph matching. Secondly, on most tested datasets, semantic matching methods achieve the second-best performance, which demonstrates the importance of semantic correlation between the query and the KG. However, they also achieve worse effectiveness than MGRAG, as they ignore the path-based feature during node matching, which is crucial for accurate semantic subgraph matching in our scenario.

4.5 KV Cache Experiments

To systematically assess the efficiency and effectiveness of the proposed graph-aware KV cache mechanism, we compare the mechanism against two representative KV cache baselines, vLLM and CacheBlend, each equipped with both similarity-based (VanillaRAG) and graph-based (HippoRAG2) retrieval modules. The results are summarized in Table 6, where all experiments are conducted on the MMQA dataset under the same hardware and model backbone (Qwen2.5-VL-7B-Instruct), ensuring fair and consistent evaluation. Table 6 summarizes the results across metrics of semantic similarity (SIM.), accuracy (ACC.), ROUGE-L (ROG.), BLEU, the time-to-first-token (TTFT), and total time cost.

Table 8: Performance comparison between VanillaRAG and MGRAG on different hop queries from the MRAMG benchmark. Times are in minutes.

# Hop	# Q	VanillaRAG				MGRAG			
		Time	ACC.	ROG.	BLEU	Time	ACC.	ROG.	BLEU
1-hop	201	**0.13**	0.613	0.379	0.240	0.35	**0.624**	**0.425**	**0.286**
2-hop	1,237	**0.10**	0.588	0.304	0.193	0.37	**0.601**	**0.335**	**0.230**
3-hop	2,357	**0.16**	0.633	0.329	0.238	0.41	**0.659**	**0.355**	**0.260**
4-hop	499	**0.19**	0.697	0.338	0.259	0.44	**0.712**	**0.373**	**0.293**
5-hop	286	**0.18**	0.678	0.319	0.205	0.45	**0.733**	**0.392**	**0.282**
6-hop	220	**0.18**	0.661	0.305	0.176	0.43	**0.738**	**0.377**	**0.249**

Table 9: Ablation experiment of MGRAG on MMQA.

Model	SIM.	ACC.	ROG.	BLEU	Time Cost
MGRAG w.o. GM	0.8362	0.7260	0.6657	0.5774	4.4 h
MGRAG w. MM Encoder	0.8513	0.7376	0.6744	0.7008	4.5 h
Vanilla MGRAG w. k=5	0.8260	0.6887	0.6266	0.5773	4.7 h
MGRAG	**0.8533**	**0.7442**	**0.6763**	**0.7215**	**4.3 h**

According to the results in Table 6, MGRAG consistently achieves superior efficiency and answer quality compared to all vLLM and CacheBlend variants. While vLLM and CacheBlend are fast when equipped with both similarity-based VanillaRAG, they incur significant overhead and reduced answer quality when applied to graph-based retrieval. In contrast, MGRAG's event-aware, graph-structure-guided KV caching enables both fast inference and high answer quality across all metrics and retrieval paradigms.

These results highlight the advantage of our approach: by leveraging the structural and semantic properties of the constructed knowledge graph, MGRAG's KV cache mechanism delivers state-of-the-art performance.

4.6 Ablation Study

4.6.1 Model ablation. To analyze MGRAG's performance, we conduct an extensive ablation study on the MMQA dataset, the ablation models tested are listed as follows:

- **MGRAG w.o. GM**: We remove MGRAG's semantic path-based subgraph matching algorithm, and construct query-specific graph with a fixed number (the maximum number of coarse-grained retrieval) of multimodal chunks.
- **MGRAG w. MM Encoder**: We directly use CLIP [45] to encode multimodal contents, not using image captions as a proxy for multimodal input in coarse-grained ranking.
- **Vanilla MGRAG**: We directly use our path-based semantic subgraph matching method and the graph-based KV cache mechanism with the VanillaRAG pipeline, and set the retrieval amount $k = 5$ (same as our MGRAG's coarse-grained retrieval amount).

Based on the results in Table 9, we can have the following observations regarding MGRAG: Firstly, **Vanilla MGRAG** achieves the worst effectiveness, which demonstrates that our MGRAG's coarse-to-fine-grained adaptive retrieval pipeline can reduce the retrieval noise by avoiding a large amount of noisy retrieval. Secondly, **MGRAG w.o. GM** achieves the second worst effectiveness, which demonstrates that using a fixed number of multimodal chunks to construct KG can introduce noise or insufficient knowledge. Lastly,

Table 10: The inverse ablation of MGRAG's path-based semantic subgraph matching (PSGM) and graph-based KV (GKV) cache mechanism, with two best-performing RAG system HippoRAG2 [25] and VanillaRAG [20] on MMQA.

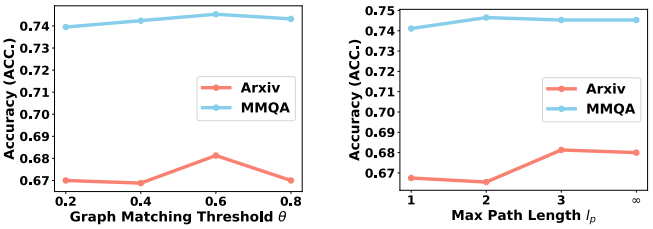
Model	SIM.	ACC.	ROG.	BLEU	Time
Hippo.	0.8381	0.6335	0.6659	0.5917	7.5 h
Hippo.+GKV	0.8392	0.6855	0.6651	0.7013	7.6 h
Hippo.+PSGM	0.8416	0.7372	0.6720	0.7128	4.1 h
Hippo.+PSGM+GKV	**0.8533**	**0.7442**	**0.6763**	**0.7215**	4.3 h
Vanilla.	0.8362	0.7053	0.5942	0.6645	**2.6 h**
Vanilla.+PSGM	0.8460	0.7175	0.6466	0.6973	4.7 h
Vanilla.+PSGM+GKV	0.8485	0.7232	0.6563	0.7001	4.8 h
MGRAG	**0.8533**	**0.7442**	**0.6763**	**0.7215**	4.3 h

aligned with the finding in [22], **MGRAG w. MM Encoder** achieves worse performance than MGRAG, which uses image caption as a proxy for multimodal input. This demonstrates the limited encoding ability of current multimodal encoders, which causes more information loss than converting the image to text and using text encoders to generate representation vectors.

4.6.2 Inverse Ablations. We carry out inverse ablations of MGRAG's three stages on the MMQA (the easiest) dataset in Table 10, where we integrate our path-based semantic subgraph matching (Namely PSGM) and graph-based KV cache (Namely GKV) mechanism with the previously best-performed graph-based RAG model HippoRAG2 [25], and non graph-based model VanillaRAG [20]. According to the results, each of our proposed path-based semantic subgraph matching (PSGM) and graph-based KV cache mechanism (GKV) can further improve the performance of these two tested baselines, and achieve the best performance when combined, demonstrating the effectiveness and generalizability of our proposed mechanisms. Moreover, it is also worth noting that the performance of HippoRAG2+PSGM+GKV is the same as our MGRAG, since while sharing the same LLM-based entity extraction step, adding the PSGM and GKV mechanisms to HippoRAG2 makes the retrieval and generation setting identical to our MGRAG.

4.7 Parameter Sensitivity

In this subsection, we analyze the parameter sensitivity of MGRAG's unique hyperparameter, the graph matching similarity threshold $0 < \theta < 1$, and the max semantic path length l_p, for which we test values from $\{1, 2, 3, \infty\}$. For MGRAG's other parameters, we align them with our tested baselines. According to Figure 5, we can have the following observations: Firstly, for the graph similarity threshold θ, when the threshold is to 0.6, MGRAG tends to achieve better performance. This demonstrates that a moderate similarity threshold can help MGRAG to better balance the precision and recall of the subgraph matching, which is crucial for accurate query-graph mapping and consequently better retrieval performance. Secondly, for the maximum path length l_p, when l_p is set to 3, MGRAG achieves the best performance, which demonstrates that considering paths with a maximum length of 3 can effectively capture the relevant semantic relationships in the knowledge graph without introducing excessive noise or computational overhead.



(a) Similarity threshold θ (b) Max path length l_p

Figure 5: Parameter sensitivity of θ and l_p.

5 CONCLUSION

We present MGRAG, a graph-based multimodal RAG framework that tackles limitations in query awareness, flexible graph matching, and KV cache management. By integrating query-driven KG construction, path-based semantic subgraph matching, and graph-aware KV caching, MGRAG achieves higher accuracy and comparable efficiency than existing baselines, especially on complex multi-hop queries. Our results demonstrate the value of principled graph methods for retrieval-augmented generation with large language models. Specifically, MGRAG attains the largest gains on large-scale and multi-hop queries while roughly halving the end-to-end latency of the strongest graph-based baseline, and inverse ablations confirm that the proposed path-based subgraph matching and graph-aware KV caching can be plugged into other RAG pipelines and consistently improve their effectiveness. In future work, we plan to explore cross-query KG reuse and lighter-weight entity extraction to further enhance serving throughput.

6 ACKNOWLEDGEMENTS

Lei Chen's work is partially supported by National Key Research and Development Program of China Grant No. 2023YFF0725100, National Science Foundation of China (NSFC) under Grant No. U22B2060, Guangdong-Hong Kong Technology Innovation Joint Funding Scheme Project No. 2024A0505040012, the Hong Kong RGC GRF Project 16213620, RIF Project R6020-19, AOE Project AoE/E-603/18, Theme-based project TRS T41-603/20R, CRF Project C2004-21G, Key Areas Special Project of Guangdong Provincial Universities 2024ZDZX1006, Guangdong Province Science and Technology Plan Project 2023A0505030011, Guangzhou municipality big data intelligence key lab, 2023A03J0012, Hong Kong ITC ITF grants MHX/078/21 and PRP/004/22FX, Hong Kong ITC TC-SKLCRCC26EG01, Zhujiang scholar program 2021JC02X170, Microsoft Research Asia Collaborative Research Grant, HKUST-Webank joint research lab, 2025 HKUST Shenzhen-Hong Kong Collaborative Innovation Institute Green Sustainability Special Fund from Shui On Xintiandi and the InnoSpace GBA, and HKUST(GZ) - CMCC(Guangzhou Branch) Metaverse Joint Innovation Lab under Grant No. P00659. Haoyang Li's work is partially supported by grant Guangdong Basic and Applied Basic Research Foundation (Project No. 2026A1515010227), No. PolyU P0052504, PolyU-Huawei P0053707, PolyU-Minshang P0057770, PolyU TDG25-28/TBP/11-IICA.

REFERENCES

[1] [n.d.]. https://github.com/Wyb0627/MGRAG/blob/main/tech_report.pdf

[2] [n.d.]. GitHub - gusye1234/nano-graphrag: A simple, easy-to-hack GraphRAG implementation — github.com. https://github.com/gusye1234/nano-graphrag. [Accessed 17-10-2025].

[3] Mohammad Mahdi Abootorabi, Amirhosein Zobeiri, Mahdi Dehghani, Mohammadali Mohammadkhani, Bardia Mohammadi, Omid Ghahroodi, Mahdieh Soleymani Baghshah, and Ehsaneddin Asgari. 2025. Ask in any modality: A comprehensive survey on multimodal retrieval-augmented generation. *arXiv preprint arXiv:2502.08826* (2025).

[4] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).

[5] Omar Adjali, Olivier Ferret, Sahar Ghannay, and Hervé Le Borgne. 2024. Multi-level information retrieval augmented generation for knowledge-based visual question answering. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 16499–16513.

[6] Shubham Agarwal, Sai Sundaresan, Subrata Mitra, Debabrata Mahapatra, Archit Gupta, Rounak Sharma, Nirmal Joshua Kapu, Tong Yu, and Shiv Saini. 2025. Cache-craft: Managing chunk-caches for efficient retrieval-augmented generation. *Proceedings of the ACM on Management of Data* 3, 3 (2025), 1–28.

[7] Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 344–354.

[8] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. 2025. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923* (2025).

[9] Yukun Cao, Zengyi Gao, Zhiyang Li, Xike Xie, and S Kevin Zhou. 2025. LEGO-GraphRAG: Modularizing Graph-based Retrieval-Augmented Generation for Design Space Exploration. *Proceedings of the VLDB Endowment* 18 (2025).

[10] Thierry Champion, Luigi De Pascale, and Petri Juutinen. 2008. The inf-Wasserstein Distance: Local Solutions and Existence of Optimal Transport Maps. *SIAM Journal on Mathematical Analysis* 40, 1 (2008), 1–20.

[11] Yihua Cheng, Yuhan Liu, Jiayi Yao, Yuwei An, Xiaokun Chen, Shaoting Feng, Yuyang Huang, Samuel Shen, Kuntai Du, and Junchen Jiang. 2025. LMCache: An Efficient KV Cache Layer for Enterprise-Scale LLM Inference. *arXiv preprint arXiv:2510.09665* (2025).

[12] Minsu Cho, Jungmin Lee, and Kyoung Mu Lee. 2010. Reweighted random walks for graph matching. In *European conference on Computer vision*. Springer, 492–505.

[13] Luigi P Cordella, Pasquale Foggia, Carlo Sansone, and Mario Vento. 2004. A (sub) graph isomorphism algorithm for matching large graphs. *IEEE transactions on pattern analysis and machine intelligence* 26, 10 (2004), 1367–1372.

[14] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The Faiss library. (2024). arXiv:2401.08281 [cs.LG]

[15] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130* (2024).

[16] Shahul Es, Jithin James, Luis Espinosa Anke, and Steven Schockaert. 2024. Ragas: Automated evaluation of retrieval augmented generation. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*. 150–158.

[17] Stefan Fankhauser, Kaspar Riesen, and Horst Bunke. 2011. Speeding up graph edit distance computation through fast bipartite matching. In *International Workshop on Graph-Based Representations in Pattern Recognition*. Springer, 102–111.

[18] Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. 2024. Detecting hallucinations in large language models using semantic entropy. *Nature* 630, 8017 (2024), 625–630.

[19] Sensen Gao, Shanshan Zhao, Xu Jiang, Lunhao Duan, Yong Xien Chng, Qing-Guo Chen, Weihua Luo, Kaifu Zhang, Jia-Wang Bian, and Mingming Gong. 2025. Scaling Beyond Context: A Survey of Multimodal Retrieval-Augmented Generation for Document Understanding. arXiv:2510.15253 [cs.CL] https://arxiv.org/abs/2510.15253

[20] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997* (2023).

[21] M.R. Garey and D.S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman.

[22] Ankush Gola. 2024. Multi-modal rag on Slide Decks. https://blog.langchain.com/multi-modal-rag-template/

[23] Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. 2025. LightRAG: Simple and Fast Retrieval-Augmented Generation. In *Proceedings of the 2025 conference on empirical methods in natural language processing*.

[24] Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2024. HippoRAG: Neurobiologically Inspired Long-Term Memory for Large Language Models. *arXiv preprint arXiv:2405.14831* (2024).

[25] Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. 2025. From RAG to Memory: Non-Parametric Continual Learning for Large Language Models. *Forty-second International Conference on Machine Learning* (2025).

[26] Haoyu Han, Yu Wang, Harry Shomer, Kai Guo, Jiayuan Ding, Yongjia Lei, Mahantesh Halappanavar, Ryan A Rossi, Subhabrata Mukherjee, Xianfeng Tang, et al. 2024. Retrieval-augmented generation with graphs (graphrag). *arXiv preprint arXiv:2501.00309* (2024).

[27] Maarten Houbraken, Sofie Demeyer, Tom Michoel, Pieter Audenaert, Didier Colle, and Mario Pickavet. 2014. The Index-based Subgraph Matching Algorithm with General Symmetries (ISMAGS): exploiting symmetry for faster subgraph enumeration. *PloS one* 9, 5 (2014), e97896.

[28] huggingface. 2025. *llm-leaderboard*. https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard#/

[29] Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. [n.d.]. Unsupervised Dense Information Retrieval with Contrastive Learning. *Transactions on Machine Learning Research* ([n. d.]).

[30] Xun Jian, Zhiyuan Li, and Lei Chen. 2023. Suff: Accelerating subgraph matching with historical data. *Proceedings of the VLDB Endowment* 16, 7 (2023), 1699–1711.

[31] Huiqiang Jiang, Qianhui Wu, , Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2024. LongLLMLingua: Accelerating and Enhancing LLMs in Long Context Scenarios via Prompt Compression. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.). Association for Computational Linguistics, Bangkok, Thailand, 1658–1677. https://aclanthology.org/2024.acl-long.91

[32] Jinhao Jiang, Kun Zhou, Xin Zhao, and Ji-Rong Wen. 2023. UniKGQA: Unified Retrieval and Reasoning for Solving Multi-hop Question Answering Over Knowledge Graph. In *The Eleventh International Conference on Learning Representations*.

[33] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*. 611–626.

[34] Marius Leordeanu and Martial Hebert. 2005. A spectral technique for correspondence problems using pairwise constraints. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, Vol. 2. IEEE, 1482–1489.

[35] Marius Leordeanu, Martial Hebert, and Rahul Sukthankar. 2009. An integer projected fixed point method for graph matching and map inference. *Advances in neural information processing systems* 22 (2009).

[36] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems* 33 (2020), 9459–9474.

[37] Haoyang LI, Yiming Li, Anxin Tian, Tianhao Tang, Zhanchao Xu, Xuejia Chen, Nicole HU, Wei Dong, Li Qing, and Lei Chen. 2025. A Survey on Large Language Model Acceleration based on KV Cache Management. *Transactions on Machine Learning Research* (2025). https://openreview.net/forum?id=z3JZzu9EA3

[38] Zihao Li. 2023. The dark side of chatgpt: Legal and ethical challenges from stochastic parrots and hallucination. *arXiv preprint arXiv:2304.14347* (2023).

[39] Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*. Association for Computational Linguistics, Barcelona, Spain, 74–81. https://aclanthology.org/W04-1013/

[40] Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the Middle: How Language Models Use Long Contexts. *Transactions of the Association for Computational Linguistics* 11 (2024), 157–173.

[41] Yuhan Liu, Hanchen Li, Yihua Cheng, Siddhant Ray, Yuyang Huang, Qizheng Zhang, Kuntai Du, Jiayi Yao, Shan Lu, Ganesh Ananthanarayanan, et al. 2024. Cachegen: Kv cache compression and streaming for fast large language model serving. In *Proceedings of the ACM SIGCOMM 2024 Conference*. 38–56.

[42] Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. 2023. Chameleon: Plug-and-play compositional reasoning with large language models. *Advances in Neural Information Processing Systems* 36 (2023), 43447–43478.

[43] Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Ruhle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Dongmei Zhang. 2024. LLMLingua-2: Data Distillation for Efficient and Faithful Task-Agnostic Prompt Compression. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.). Association for Computational Linguistics, Bangkok, Thailand, 963–981. https://aclanthology.org/2024.findings-acl.57

[44] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the*

40th annual meeting of the Association for Computational Linguistics. 311–318.

[45] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*. PmLR, 8748–8763.

[46] Ritesh Sarkhel and Arnab Nandi. 2019. Visual segmentation for information extraction from heterogeneous visually rich documents. In *Proceedings of the 2019 international conference on management of data*. 247–262.

[47] Ritesh Sarkhel and Arnab Nandi. 2021. Improving information extraction from visually rich documents using visual span representations. *Proceedings of the VLDB Endowment* 14, 5 (2021).

[48] Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. 2025. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. *arXiv preprint arXiv:2503.05592* (2025).

[49] Jun Song, Jingyi Ding, Irshad Kandy, Yanghao Lin, Zhongjia Wei, Zilong Zhou, Zhiwei Peng, Jixi Shan, Hongyue Mao, Xiuqi Huang, et al. 2025. Magnus: A Holistic Approach to Data Management for Large-Scale Machine Learning Workloads. *Proceedings of the VLDB Endowment* 18, 12 (2025), 4964–4977.

[50] Alon Talmor, Ori Yoran, Amnon Catav, Dan Lahav, Yizhong Wang, Akari Asai, Gabriel Ilharco, Hannaneh Hajishirzi, and Jonathan Berant. 2021. Multimodalqa: Complex question answering over text, tables and images. *International Conference on Learning Representations* (2021).

[51] Jianheng Tang, Weiqi Zhang, Jiajin Li, Kangfei Zhao, Fugee Tsung, and Jia Li. 2023. Robust attributed graph alignment via joint structure learning and optimal transport. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 1638–1651.

[52] Jianheng Tang, Xi Zhao, Lemin Kong, Xiaofang Zhou, and Jia Li. 2025. Fused Gromov-Wasserstein Alignment for Graph Edit Distance Computation and Beyond. *Proceedings of the VLDB Endowment* 18, 10 (2025), 3641–3654.

[53] Matthias Urban and Carsten Binnig. 2024. Eleet: Efficient learned query execution over text and tables. *Proceedings of the VLDB Endowment* 17, 13 (2024), 4867–4880.

[54] Mengzhao Wang, Xiangyu Ke, Xiaoliang Xu, Lu Chen, Yunjun Gao, Pinpin Huang, and Runkai Zhu. 2024. Must: An effective and scalable framework for multimodal search of target modality. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. IEEE, 4747–4759.

[55] Mengzhao Wang, Haotian Wu, Xiangyu Ke, Yunjun Gao, Xiaoliang Xu, and Lu Chen. 2024. An Interactive Multi-Modal Query Answering System with Retrieval-Augmented Large Language Models. *Proceedings of the VLDB Endowment* 17, 12 (2024), 4333–4336.

[56] Yubo Wang, Haoyang Li, Fei Teng, and Lei Chen. 2026. AGRAG: Advanced Graph-based Retrieval-Augmented Generation for LLMs. *2026 IEEE 42nd International Conference on Data Engineering (ICDE)* (2026).

[57] Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. C-Pack: Packaged Resources To Advance General Chinese Embedding. arXiv:2309.07597 [cs.CL]

[58] Mingji Yang, Hanzhi Wang, Zhewei Wei, Sibo Wang, and Ji-Rong Wen. 2024. Efficient algorithms for personalized pagerank computation: A survey. *IEEE Transactions on Knowledge and Data Engineering* 36, 9 (2024), 4582–4602.

[59] Jiayi Yao, Hanchen Li, Yuhan Liu, Siddhant Ray, Yihua Cheng, Qizheng Zhang, Kuntai Du, Shan Lu, and Junchen Jiang. 2025. CacheBlend: Fast large language model serving for RAG with cached knowledge fusion. In *Proceedings of the Twentieth European Conference on Computer Systems*. 94–109.

[60] Qinhan Yu, Zhiyou Xiao, Binghui Li, Zhengren Wang, Chong Chen, and Wentao Zhang. 2025. MRAMG-Bench: A Comprehensive Benchmark for Advancing Multimodal Retrieval-Augmented Multimodal Generation. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 3616–3626.

[61] Xu Yuan, Liangbo Ning, Wenqi Fan, and Qing Li. 2025. mKG-RAG: Multimodal Knowledge Graph-Enhanced RAG for Visual Question Answering. *arXiv preprint arXiv:2508.05318* (2025).

[62] Haoyu Zhang, Jun Liu, Zhenhua Zhu, Shulin Zeng, Maojia Sheng, Tao Yang, Guohao Dai, and Yu Wang. 2024. Efficient and effective retrieval of dense-sparse hybrid vectors using graph-based approximate nearest neighbor search. *arXiv preprint arXiv:2410.20381* (2024).

[63] Qinggang Zhang, Shengyuan Chen, Yuanchen Bei, Zheng Yuan, Huachi Zhou, Zijin Hong, Junnan Dong, Hao Chen, Yi Chang, and Xiao Huang. 2025. A survey of graph retrieval-augmented generation for customized large language models. *arXiv preprint arXiv:2501.13958* (2025).

[64] Tao Zhang, Ziqi Zhang, Zongyang Ma, Yuxin Chen, Zhongang Qi, Chunfeng Yuan, Bing Li, Junfu Pu, Yuxuan Zhao, Zehua Xie, et al. 2024. mR^2AG: Multimodal Retrieval-Reflection-Augmented Generation for Knowledge-Based VQA. *arXiv preprint arXiv:2411.15041* (2024).

[65] Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, et al. 2023. Siren's song in the AI ocean: a survey on hallucination in large language models. *arXiv preprint arXiv:2309.01219* (2023).

[66] Zhijie Zhang, Yujie Lu, Weiguo Zheng, and Xuemin Lin. 2024. A comprehensive survey and experimental study of subgraph matching: trends, unbiasedness, and interaction. *Proceedings of the ACM on Management of Data* 2, 1 (2024), 1–29.

[67] Ruochen Zhao, Hailin Chen, Weishi Wang, Fangkai Jiao, Do Xuan Long, Chengwei Qin, Bosheng Ding, Xiaobao Guo, Minzhi Li, Xingxuan Li, et al. [n.d.]. Retrieving Multimodal Information for Augmented Generation: A Survey. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

[68] Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Livia Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. 2024. Sglang: Efficient execution of structured language model programs. *Advances in neural information processing systems* 37 (2024), 62557–62583.