Use Graph When It Needs: Efficiently and Adaptively Integrating Retrieval-Augmented Generation with Graphs

Anonymous ACL submission

Abstract

Large language models (LLMs) often struggle with knowledge-intensive tasks due to hallucinations and outdated parametric knowledge. While Retrieval-Augmented Generation (RAG) addresses this by integrating external corpora, its effectiveness is limited by fragmented information in unstructured domain documents. Graph-augmented RAG (GraphRAG) emerged to enhance contextual reasoning through structured knowledge graphs, yet paradoxically underperforms vanilla RAG in real-world scenarios, exhibiting significant accuracy drops and prohibitive latency despite gains on complex queries. We identify the rigid application of GraphRAG to all queries, regardless of complexity, as the root cause. To resolve this, we propose an efficient and adaptive GraphRAG framework called EA-GraphRAG that dynamically integrates RAG and GraphRAG paradigms through syntax-aware complexity analysis. Our approach introduces: (i) a syntactic feature constructor that parses each query and extracts a set of structural features; (ii) a lightweight complexity scorer that maps these features to a continuous complexity score; and (iii) a score-driven routing policy that selects RAG for low-score queries, invokes graph-based retrieval for high-score queries, and applies complexity-aware reciprocal rank fusion to handle borderline cases. Extensive experiments on a comprehensive benchmark, consisting of two single-hop and two multi-hop QA benchmarks, demonstrate that our EA-GraphRAG significantly improves accuracy, reduces latency, and achieves state-of-the-art performance in handling mixed scenarios involving both simple and complex queries. Our code is available at https://anonymous.4open.science/r/EA_GraphRAG-2473/.

1 Introduction

Large language models (LLMs) have substantially advanced natural language processing, but they still suffer from hallucinations and outdated parametric knowledge in knowledge-intensive tasks. Retrieval-Augmented Generation (RAG) (Gao et al., 2023; Lewis et al., 2020) mitigates this issue by retrieving external evidence and conditioning generation on the retrieved context. By grounding responses in relevant passages, RAG improves factuality and has become a widely used framework for open-domain and domain-specific question answering.

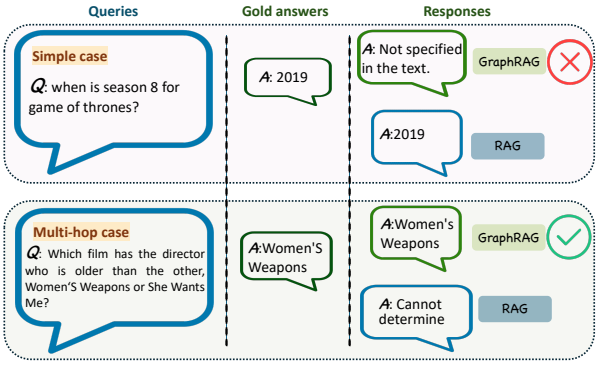

Figure 1: An illustrative example showing that GraphRAG often underperforms vanilla RAG on simple single-hop queries, but excels on multi-hop queries requiring relational reasoning; conversely, vanilla RAG tends to perform better on single-hop queries while struggling with multi-hop reasoning.

Despite its effectiveness, standard RAG often struggles with complex queries that require reasoning over multiple pieces of interconnected evidence. Since documents are typically indexed as isolated chunks, important entity relations and cross-document dependencies may be lost during retrieval. Graph retrieval-augmented generation (GraphRAG) (Zhang et al., 2025a; Procko and Ochoa, 2024; Peng et al., 2024) addresses this limitation by organizing external knowledge into graph structures and retrieving evidence through entities, relations, and graph traversal. This design is particularly useful for multi-hop questions, where the answer depends on relational paths rather than a single local passage.

However, GraphRAG is not universally beneficial. Graph construction and graph-based traversal introduce additional cost, and the retrieved graph context may contain irrelevant or noisy relational information for simple factual queries. Recent studies show that GraphRAG can underperform vanilla RAG on single-hop or time-sensitive queries (Han et al., 2025; Zhou et al., 2025; Xiang et al., 2025). These observations suggest a fundamental trade-off: RAG is often more efficient and reliable for simple queries, whereas GraphRAG is more suitable for queries requiring relational or multi-hop reasoning. Therefore, applying graph-based retrieval to every query is suboptimal.

An ideal system should adaptively select the appropriate retrieval paradigm according to query complexity. However, this raises two challenges: how to estimate query complexity accurately and efficiently, and how to route queries between dense retrieval and graph-based retrieval without introducing excessive overhead. Existing adaptive RAG methods often rely on LLMs or language-model classifiers as semantic routers, which can be slow and high-cost when invoked for every input query. Moreover, such routers usually model complexity implicitly through semantic representations, prompting signals, or model uncertainty, making the routing decision less interpretable and difficult to connect with explicit structural properties of the query.

To address these challenges, we propose EA-GraphRAG, an efficient and adaptive framework that integrates RAG and GraphRAG through syntax-aware query complexity analysis. EA-GraphRAG first extracts explicit syntactic and structural features from each query, and then uses a lightweight MLP adapter to map these features to a continuous complexity score. Based on this score, the framework performs tri-routing: low-complexity queries are handled by dense retrieval, high-complexity queries are handled by graph-based retrieval, and intermediate cases are addressed through complexity-aware reciprocal rank fusion. This design provides a low-cost and interpretable alternative to LLM-based routing, while preserving the complementary strengths of RAG and GraphRAG. Experiments on two single-hop and two multi-hop QA benchmarks demonstrate that EA-GraphRAG improves both answer accuracy and retrieval efficiency under mixed query distributions.

Our contributions are summarized as follows:

- We reveal the complementary behavior of RAG and GraphRAG under mixed query complexity, showing that graph-based retrieval should be selectively applied rather than uniformly used.
- We propose EA-GraphRAG, a syntax-aware adaptive framework that uses a lightweight MLP adapter to estimate query complexity and route queries among dense retrieval, graph-based retrieval, and fusion.
- Extensive experiments on comprehensive benchmarks consisting of both single-hop QA datasets and multi-hop QA datasets demonstrate that EA-GraphRAG outperforms the state-of-the-art baselines.

2 Related Work

Retrieval-Augmented Generation (RAG) enhances LLMs by retrieving external knowledge and using it as evidence for generation (Gao et al., 2023; Lewis et al., 2020). Early retrieval components include sparse lexical retrievers such as BM25 (Robertson and Walker, 1994) and dense retrievers such as Contriever (Izacard et al., 2021) and ColBERTv2 (Santhanam et al., 2022). Later studies further integrate retrieval with reasoning or few-shot learning, such as IRCoT (Trivedi et al., 2023) and Atlas (Izacard et al., 2023). However, standard RAG retrieves isolated text chunks and may fail to capture relational structures across documents, limiting its effectiveness on complex multi-hop queries. To address this issue, GraphRAG methods introduce graph structures into retrieval. Microsoft GraphRAG (Edge et al., 2024) performs hierarchical community-based search over a constructed knowledge graph, RAPTOR (Sarthi et al., 2024) builds recursive tree-structured summaries for multi-scale retrieval, HippoRAG (Gutiérrez et al., 2024) and HippoRAG2 (Gutiérrez et al., 2025) use graph-based memory and Personalized PageRank for associative retrieval, and LightRAG (Guo et al., 2025) improves scalability through lightweight graph-enhanced indexing. These methods improve reasoning over connected evidence, but their static use of graph retrieval can be inefficient or even harmful for simple queries.

Recent studies have explored adaptive RAG, where the system dynamically decides whether, when, or how to retrieve external evidence for a given query. Self-RAG (Asai et al., 2024) trains a

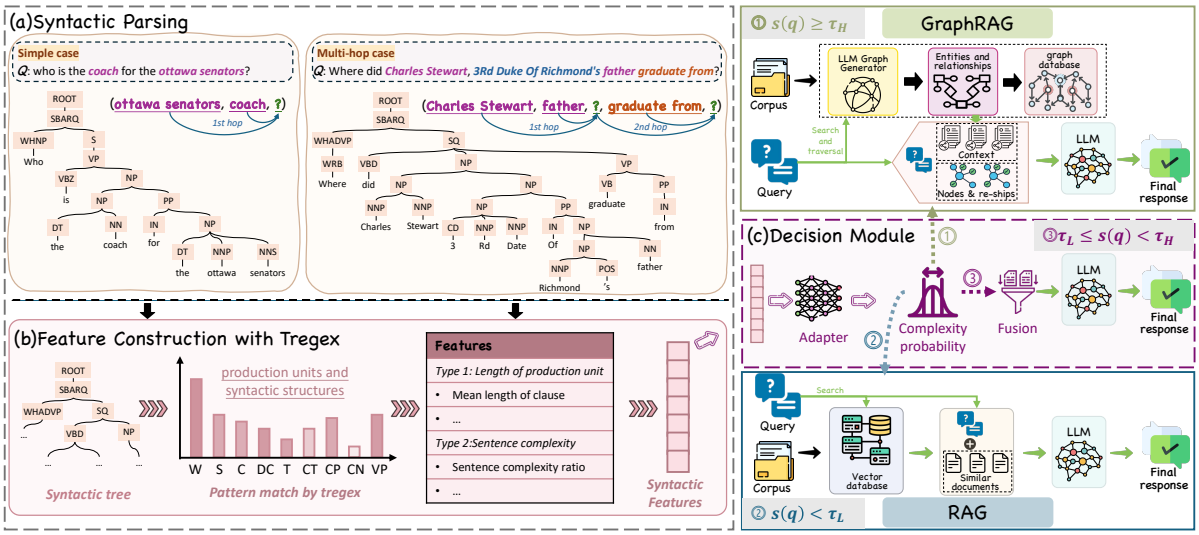

Figure 2: **The overview of our EA-GraphRAG framework.** In the (c) Decision Module stage, an MLP adapter produces a *scalar* complexity score $s(q) \in (0, 1)$. Two thresholds route the query: $s(q) < \tau_L \Rightarrow$ Dense retrieval; $s(q) \geq \tau_H \Rightarrow$ Graph-based retrieval; $\tau_L \leq s(q) < \tau_H \Rightarrow$ Fusion, where the documents retrieved by dense retrieval and graph-based retrieval are merged by weighted Reciprocal Rank Fusion (wRRF).

language model to use reflection tokens to decide when to retrieve, critique retrieved evidence, and revise its generations. SeaKR (Yao et al., 2025) estimates the uncertainty of an LLM from its internal states and activates retrieval when the model is not sufficiently confident. Adaptive-RAG (Jeong et al., 2024) uses a query-complexity classifier to select among no retrieval, single-step retrieval, and multi-step retrieval. Han et al. (2025) systematically compare RAG and GraphRAG, showing their complementary strengths, and further propose an LLM-based classifier to select between RAG and GraphRAG. Dynamic strategy planning methods (Parekh et al., 2025) use an LLM-based planner to choose suitable question-answering strategies for different queries. AdaGCRAG (Zhang et al., 2025b) adaptively combines graph and chunk retrieval to reduce irrelevant graph-induced context in lightweight RAG.

Although these methods demonstrate the value of adaptive retrieval, many of them rely on LLMs as semantic routers or classifiers. Such LLM-based routing is flexible but often slow and high-cost, especially when the router must be invoked for every input query. Moreover, existing routers usually model query complexity implicitly from semantic representations, prompting signals, or model uncertainty, making the decision process less interpretable and difficult to relate to explicit structural properties of the query. Linguistic studies provide a complementary view: syntactic complexity has long been studied as an explicit and measurable property of language, characterized by structural indicators such as clauses, T-units, dependent clauses, complex nominals, coordination, and subordination (Lu, 2010; Zhang and Lu, 2022; Ouyang et al., 2022). Inspired by this perspective, EA-GraphRAG models query complexity with syntactic and structural features rather than an LLM router. This provides an efficient, low-cost, and interpretable mechanism for adaptive retrieval, while directly modeling when dense retrieval is sufficient, when graph-based retrieval is needed, and when both should be fused.

3 EA-GraphRAG

We consider retrieval-augmented question answering under a complexity-aware tri-routing setting. Given a query q and a corpus $\mathcal{C} = \{c_1, c_2, \ldots, c_N\}$, the system retrieves evidence passages and feeds them to a fixed LLM generator $\mathcal{G}$ to produce the answer. EA-GraphRAG provides three retrieval paths: a dense retrieval path, a graph-based retrieval path, and a complexity-aware fusion path.

Formally, the dense retriever $\mathcal{R}_{\text{rag}}$ returns a ranked passage list $\pi_{\text{rag}}(q) \subset \mathcal{C}$, while the graph-based retriever $\mathcal{R}_{\text{gr}}$ operates on a pre-built heterogeneous graph $G = (V, E)$ and returns $\pi_{\text{gr}}(q) \subset \mathcal{C}$. For uncertain cases, a fusion operator $\mathcal{F}_{\text{fus}}$ merges both lists with weighted Reciprocal Rank Fusion

Algorithm 1 Inference pipeline of EA-GraphRAG

Require: query q; thresholds (τ_L, τ_H); graph $G = (V, E)$, $V = N \cup \mathcal{C}$; facts $\mathcal{F}$; encoder M; damping α; retrieved chunks K; RRF constant k; generator $\mathcal{G}$
Ensure: answer $\hat{y}$

Extract query features: $\mathbf{x} \leftarrow \phi(q)$
Compute complexity score: $s(q) \leftarrow \sigma(\ell_\theta(\phi(q))) \in (0, 1)$
if $s(q) \geq \tau_H$ **then** ▷ Graph-based retrieval
  Score facts: $s_i \leftarrow \langle M(q), M(f_i) \rangle$, $f_i \in \mathcal{F}$
  Select top facts: $\mathcal{F}_{top} \leftarrow \mathrm{Rerank}(\mathrm{TopK}(\tilde{\mathrm{s}}, k), q)$
  Assign entity weights: $w(n) \leftarrow \tilde{s}_f / |\{c : (n, c) \in E\}|$
  Select seed entities: $Q \leftarrow \mathrm{TopK}(\{w(n) : n \in N\}, k')$
  Initialize PPR scores: $\mathbf{r}_0[v] \leftarrow w(v)$ if $v \in Q$, else 0
  Run PPR: $\mathbf{r} \leftarrow \mathrm{PPR}(G, \mathbf{r}_0, \alpha)$
  Rank chunks: $L \leftarrow \mathrm{TopK}(\mathbf{r}[\mathcal{C}], K)$
else if $s(q) \leq \tau_L$ **then** ▷ Dense retrieval
  Encode query: $\mathbf{q} \leftarrow M(q)$
  Rank chunks: $L \leftarrow \mathrm{TopK}(\{\langle \mathbf{q}, M(c) \rangle : c \in \mathcal{C}\}, K)$
else ▷ Complexity-aware fusion
  Run dense retrieval: $\mathbf{q} \leftarrow M(q)$
  Get dense list: $L_r \leftarrow \mathrm{TopK}(\{\langle \mathbf{q}, M(c) \rangle : c \in \mathcal{C}\}, K)$
  Run graph retrieval: compute $\mathcal{F}_{top}$, Q, and $\mathbf{r}$
  Get graph list: $L_g \leftarrow \mathrm{TopK}(\mathbf{r}[\mathcal{C}], K)$
  Fusion weights: $w_{gr} \leftarrow s(q)$; $w_{rag} \leftarrow 1 - s(q)$
  Apply weighted RRF: $\mathrm{RRF}(c) \leftarrow \frac{w_{rag}}{k + r_r(c)} + \frac{w_{gr}}{k + r_g(c)}$
  Fused ranking: $L \leftarrow \mathrm{TopK}(\{\mathrm{RRF}(c) : c \in L_r \cup L_g\}, K)$
end if
Pack context and generate: $\hat{y} \leftarrow \mathcal{G}(q, \mathrm{Pack}(L))$
return $\hat{y}$

(wRRF):

$$\pi_{\mathrm{fus}}(q) = \mathcal{F}_{\mathrm{fus}}\big(\pi_{\mathrm{rag}}(q), \pi_{\mathrm{gr}}(q), s(q)\big),$$

where $s(q) \in (0, 1)$ is a query complexity score. The score is predicted by a lightweight adapter g_θ from query features $\phi(q) \in \mathbb{R}^d$. If $s(q) \geq \tau_H$, EA-GraphRAG uses graph-based retrieval; if $s(q) \leq \tau_L$, it uses dense retrieval; otherwise, it fuses both retrieval results. Algorithm 1 summarizes the inference pipeline.

3.1 Offline Indexing

Following (Gutiérrez et al., 2025), we index the corpus $\mathcal{C}$ as a heterogeneous graph $G = (V, E)$, where $V = N \cup \mathcal{C}$. Here, N contains noun phrases and entities extracted from the corpus, and $\mathcal{C}$ contains passage nodes. The edge set E includes three types of edges: relation edges from OpenIE triples (h, r, t), occurrence edges linking entities to their source passages, and synonymy edges between semantically similar entity nodes. This graph preserves both entity-level relational structure and passage-level textual evidence for graph-based retrieval. Hyperparameters follow the default settings in (Gutiérrez et al., 2025).

3.2 Query Complexity Measurement

The key design of EA-GraphRAG is to estimate whether a query requires graph-based reasoning. Instead of using an LLM as a semantic router, we construct explicit query features and train a lightweight adapter to predict a complexity score. We use syntactic and structural features as the primary complexity signals, because query complexity is often reflected by clause structure, dependency patterns, coordination, subordination, and nested nominal structures. In addition, we include lightweight lexical and entity-level features as auxiliary indicators, such as query length, entity density, WH-question type, and logical connectors. These auxiliary features do not require LLM inference and are used only to complement the syntactic complexity representation.

Specifically, the featurizer $\phi(q)$ extracts four groups of features: (i) constituency-based syntactic complexity measures, including clauses, T-units, dependent clauses, complex T-units, coordinate phrases, complex nominals, and verb phrases; (ii) dependency-based structural measures, including dependency distance, long-range dependencies, dependency relation counts, and tree imbalance; (iii) lightweight lexical and entity-level signals, including token statistics, POS ratios, named entity counts, WH-word indicators, and logical connectors; and (iv) tree-level and interaction features, such as tree depth, tree width, branching factor, tokens per clause, entities per token, and connectors per clause. Details of the feature groups are provided in Appendix B.

The featurizer $\phi(q)$ assembles all feature categories into a d'-dimensional vector, where d' is the raw feature dimensionality before selection. We apply mutual information-based feature selection to identify the top-k most discriminative features for predicting whether GraphRAG outperforms RAG, where k is the number of selected features. The final feature vector $\mathbf{x} \in \mathbb{R}^d$ has dimensionality $d = k$ and is standardized using z-score normalization:

$$\mathbf{x} = (\mathbf{x} - \boldsymbol{\mu}) \oslash \boldsymbol{\sigma},$$

where $\boldsymbol{\mu} \in \mathbb{R}^d$ and $\boldsymbol{\sigma} \in \mathbb{R}^d$ are the mean and standard deviation vectors estimated on the training

data, and $\oslash$ denotes element-wise division.

3.3 MLP Adapter

Given the feature vector $\phi(q)$, the adapter predicts a continuous complexity score:

$$s(q) = \sigma(\ell_\theta(\phi(q))) \in (0, 1), \tag{1}$$

where ℓ_θ is the MLP function parameterized by θ, and $\sigma(t) = 1/(1 + e^{-t})$ is the sigmoid function.

The adapter is trained on disagreement cases between RAG and GraphRAG. Let $z_i^{\text{rag}}, z_i^{\text{gr}} \in \{0, 1\}$ indicate whether RAG and GraphRAG correctly answer sample i, respectively. We construct the training set

$$\mathcal{T} = \{i : z_i^{\text{rag}} \oplus z_i^{\text{gr}} = 1\},$$

where $\oplus$ denotes exclusive OR. Each sample is labeled as $y_i = 1$ if GraphRAG is correct and RAG is incorrect, and $y_i = 0$ if RAG is correct and GraphRAG is incorrect. Samples where both methods agree are excluded, since they do not provide direct supervision for selecting between the two retrieval paths. The adapter is optimized as a binary classifier, and implementation details are provided in Appendix C.

At inference time, the complexity score controls the routing decision:

$$\text{route}(q) = \begin{cases} \text{GraphRAG}, & s(q) \geq \tau_H, \\ \text{RAG}, & s(q) \leq \tau_L, \\ \text{Fusion}, & \tau_L < s(q) < \tau_H, \end{cases} \tag{2}$$

where thresholds $\tau_L < \tau_H$ are tuned on the validation set to balance answer accuracy and retrieval latency.

3.4 Online Retrieval

Dense retrieval. For the dense retrieval path, we encode the query and passages with encoder M, and rank passages by inner-product similarity:

$$\pi_{\text{rag}}(q) = \text{TopK}_{c \in \mathcal{C}} \langle M(q), M(c) \rangle. \tag{3}$$

Graph-based retrieval. For the graph-based path, EA-GraphRAG first matches the query against extracted facts:

$$s_i = \langle M(q), M(f_i) \rangle, \quad \forall f_i \in \mathcal{F}, \tag{4}$$

where $\mathcal{F}$ denotes the set of all facts, M is the encoder for fact-query matching, f_i represents the i-th fact. The similarity scores are normalized to obtain $\tilde{\mathbf{s}} = [\tilde{s}_1, \tilde{s}_2, \ldots, \tilde{s}_{|\mathcal{F}|}]^\top$, where $\tilde{s}_i$ is the normalized similarity score for fact f_i. Then the top facts are selected and reranked:

$$\mathcal{F}_{\text{top}} = \text{Rerank}\big(\text{TopK}(\tilde{\mathbf{s}}, k), q\big). \tag{5}$$

where k is a hyperparameter controlling the number of candidate facts. Entities appearing in $\mathcal{F}_{\text{top}}$ are used as seed nodes, with weights normalized by the number of passages containing each entity to reduce hub dominance. We initialize the reset vector $\mathbf{r}_0$ on these seed entities and run Personalized PageRank:

$$\mathbf{r} = \text{PPR}(G, \mathbf{r}_0, \alpha), \tag{6}$$

where α is the damping factor. Passage nodes are then ranked by their PPR scores:

$$\pi_{\text{gr}}(q) = \text{TopK}\big(\mathbf{r}[\mathcal{C}], K\big). \tag{7}$$

Complexity-aware fusion. When $\text{route}(q) = \text{Fusion}$, we combine $\pi_{\text{rag}}(q)$ and $\pi_{\text{gr}}(q)$ by weighted RRF:

$$\text{RRF}(c) = (1 - s(q)) \cdot \frac{1}{k + r_{\text{rag}}(c)} + s(q) \cdot \frac{1}{k + r_{\text{gr}}(c)}, \tag{8}$$

where $r_{\text{rag}}(c)$ and $r_{\text{gr}}(c)$ are the ranks of passage c in the dense and graph-based retrieval lists. The fused list is:

$$\pi_{\text{fus}}(q) = \text{TopK}_{c \in \pi_{\text{rag}}(q) \cup \pi_{\text{gr}}(q)} \text{RRF}(c). \tag{9}$$

Answer generation. The final evidence set is determined by the selected route:

$$D(q) = \begin{cases} \pi_{\text{rag}}(q), & \text{if route}(q) = \text{RAG}, \\ \pi_{\text{gr}}(q), & \text{if route}(q) = \text{GraphRAG}, \\ \pi_{\text{fus}}(q), & \text{if route}(q) = \text{Fusion}. \end{cases} \tag{10}$$

The generator produces the final answer as:

$$\hat{y} = \mathcal{G}(q, \text{Pack}(D(q))), \tag{11}$$

where $\text{Pack}(\cdot)$ formats the retrieved passages into the input context of the generator.

4 Experiments

We evaluate EA-GraphRAG on both single-hop and multi-hop question answering benchmarks to examine whether it can adaptively balance dense retrieval and graph-based retrieval under mixed query distributions.

Method	Simple QA				Multi-Hop QA				Mix	
	NQ		PopQA		HotpotQA		2Wiki		Mix	
	Acc.	GPT-Acc.	Acc.	GPT-Acc.	Acc.	GPT-Acc.	Acc.	GPT-Acc.	Acc.	GPT-Acc.
Large Language Models										
Llama-3-8B (Dubey et al., 2024)	28.2	30.8	9.3	8.1	16.4	19.9	24.5	11.6	19.6	17.6
Qwen3-8B (Yang et al., 2025)	37.6	38.7	24.4	22.8	24.3	28.8	35.4	25.3	30.5	28.9
GPT-3.5-turbo	56.5	62.5	36.1	35.3	37.2	45.9	37.2	33.2	41.8	44.3
GPT-4o-mini (Hurst et al., 2024)	54.2	61.1	34.3	33.8	34.8	44.1	34.3	34.8	39.5	43.4
RAG Baselines										
BM25 (Robertson and Walker, 1994)	62.3	67.5	50.4	51.3	56.1	68.2	48.0	52.9	54.2	59.9
Contriever (Izacard et al., 2021)	64.9	68.1	68.8	68.2	53.6	65.5	37.3	46.3	56.1	62.2
ColBERTv2 (Santhanam et al., 2022)	68.7	71.7	72.8	71.1	63.4	75.9	56.4	62.7	65.3	70.2
GraphRAG Baselines										
RAPTOR (Sarthi et al., 2024)	62.4	66.7	55.7	53.5	60.7	72.0	34.7	44.3	53.4	59.1
G-retriever (He et al., 2024)	66.4	67.1	28.8	28.3	45.3	51.6	50.6	32.1	47.7	44.7
LightRAG (Guo et al., 2025)	64.1	69.4	68.5	68.1	59.3	72.2	46.9	57.4	59.6	66.7
KGP (Wang et al., 2024)	61.3	67.6	52.6	51.6	54.8	63.7	43.6	53.8	53.1	59.2
HippoRAG (Gutiérrez et al., 2024)	57.2	61.7	70.6	70.4	57.6	70.6	68.6	73.8	63.5	69.1
HippoRAG2 (Gutiérrez et al., 2025)	67.7	70.9	73.2	71.7	65.5	79.5	67.7	72.5	68.5	73.6
Ours	**69.1**	**72.6**	**75.1**	**73.5**	**65.9**	**80.2**	**76.3**	**81.5**	**71.6**	**76.9**

Table 1: QA performance across single-hop, multi-hop, and mixed benchmarks. Acc. denotes Contain-Match Accuracy, and GPT-Acc. denotes GPT-Evaluation Accuracy. Best results are in **bold**, and second-best results are underlined.

4.1 Baselines

We compare EA-GraphRAG with three groups of baselines: (1) LLM-only baselines, including Llama-3-8B (Dubey et al., 2024), Qwen3-8B (Yang et al., 2025), GPT-3.5-turbo, and GPT-4o-mini (Hurst et al., 2024); (2) dense and sparse RAG baselines, including BM25 (Robertson and Walker, 1994), Contriever (Izacard et al., 2021), and ColBERTv2 (Santhanam et al., 2022); and (3) GraphRAG baselines, including RAPTOR (Sarthi et al., 2024), G-retriever (He et al., 2024), LightRAG (Guo et al., 2025), KGP (Wang et al., 2024), HippoRAG (Gutiérrez et al., 2024), and HippoRAG2 (Gutiérrez et al., 2025).

4.2 Datasets

We use two single-hop QA datasets, Natural Questions (NQ) (Kwiatkowski et al., 2019) and PopQA (Mallen et al., 2022), and two multi-hop QA datasets, 2WikiMultihopQA (Ho et al., 2020) and HotpotQA (Yang et al., 2018). Following the data processing protocol of HippoRAG2 (Gutiérrez et al., 2025), we sample 1,000 queries from each dataset. We further construct a mixed benchmark by combining all four datasets and merging their corpora, resulting in 4,000 queries and 34,239 documents. Table 2 reports the dataset statistics.

	NQ	PopQA	2Wiki	HotpotQA	Mix
Num. queries	1,000	1,000	1,000	1,000	4,000
Num. docs	9,633	8,676	6,119	9,811	34,239

Table 2: Dataset statistics

4.3 Metrics

Evaluation is conducted from both answer accuracy and retrieval performance perspectives. For answer evaluation, following existing work (Wang et al.), we employ two complementary metrics that jointly assess surface fidelity and semantic correctness:

Contain-Match Accuracy, which measures whether the gold answer appears as a substring within the model's prediction, allowing minor surface-form variations while maintaining semantic precision.

GPT-Evaluation Accuracy, an LLM-based criterion in which the evaluator receives the question, gold answer, and prediction, and determines whether the predicted answer is semantically equivalent to the reference.

Together, these metrics offer a balanced and interpretable assessment by combining the reproducibility of string matching with the nuanced semantic understanding of LLM-based judgment.

For retrieval evaluation, we report two complementary metrics. Retrieval Time per Query (s) quantifies efficiency by measuring the average time the retriever needs to return candidate documents, reflecting end-to-end retrieval latency. Recall@k assesses quality as the proportion of queries

Method	Simple QA				Multi-Hop QA				Mix	
	NQ		PopQA		HotpotQA		2Wiki		Mix	
	R@3	R@5	R@3	R@5	R@3	R@5	R@3	R@5	R@3	R@5
RAG Baselines										
BM25 (Robertson and Walker, 1994)	37.2	51.9	24.8	29.8	62.6	71.8	57.3	61.4	45.5	53.7
Contriever (Izacard et al., 2021)	36.2	50.9	34.3	42.4	60.6	68.5	45.1	50.9	44.0	50.9
ColBERTv2 (Santhanam et al., 2022)	47.4	65.4	47.0	48.9	71.5	77.8	63.9	67.6	57.4	64.9
GraphRAG Baselines										
RAPTOR (Sarthi et al., 2024)	44.3	59.7	28.9	29.3	45.9	50.1	33.4	34.9	38.1	43.5
HippoRAG (Gutiérrez et al., 2024)	29.6	43.4	44.6	52.8	68.9	76.2	77.6	86.0	55.2	64.7
HippoRAG2 (Gutiérrez et al., 2025)	52.3	70.8	48.0	50.9	**82.7**	**89.6**	79.4	85.2	65.6	74.1
Ours	**53.9**	**71.3**	**49.2**	**54.1**	80.5	87.2	**80.9**	**88.1**	**66.0**	**74.9**

Table 3: Retrieval recall (R@3 / R@5) shown as percentages with one decimal. Best per column in **bold**; second-best underlined.

for which at least one relevant document appears among the top-k results. Together, these metrics characterize the accuracy–efficiency trade-off and offer a more holistic view of system performance.

4.4 Implementation Details

Our experiments were conducted on a server equipped with six RTX-3090 GPUs. For fair comparison, all RAG and GraphRAG baselines are evaluated under the same setup, using GPT-4o-mini for answer generation and graph construction, GPT-4o for evaluation, with the number of retrieved chunks set to $K = 5$. We split the 4,000-query dataset into 1,000 training queries and 3,000 test queries. From the 1,000 training queries, we select 200 disagreement cases where RAG and GraphRAG produce different correctness outcomes, and use them to train the query adapter. Although the adapter is trained on a small disagreement set, it operates on low-dimensional syntactic and structural features rather than high-dimensional semantic embeddings, which helps reduce overfitting and provides a lightweight signal for query routing.

4.5 Main Results

4.5.1 QA Performance

Table 1 reports the answer accuracy results. EA-GraphRAG achieves the best overall performance on the mixed benchmark, with 71.6 Acc. and 76.9 GPT-Acc., outperforming both RAG and GraphRAG baselines. The improvement is particularly clear on 2Wiki, where relational reasoning across multiple documents is required: EA-GraphRAG improves Acc. from 56.4 to 76.3 over the strongest dense baseline ColBERTv2, and also outperforms HippoRAG2 in both Acc. and GPT-Acc. On single-hop datasets, EA-GraphRAG remains competitive with strong dense retrievers, showing that the routing mechanism avoids unnecessary graph-based retrieval for simpler queries. These results indicate that dynamically selecting between dense retrieval, graph-based retrieval, and fusion can better handle heterogeneous query complexity than using a fixed retrieval paradigm.

4.5.2 Retrieval Performance

Table 3 shows the retrieval recall results. On the mixed benchmark, EA-GraphRAG achieves the best overall recall, with R@3 of 66.0 and R@5 of 74.9. Compared with ColBERTv2, it improves R@3 and R@5 by 8.6 and 10.0 points, respectively. Compared with HippoRAG2, it also achieves modest improvements on the mixed benchmark while maintaining strong performance across both simple and multi-hop settings. These results suggest that complexity-aware routing and fusion can retrieve more reliable evidence by combining the complementary strengths of dense and graph-based retrieval.

We further report retrieval latency in Table 4. Dense retrieval is much faster than graph-based retrieval, while graph-based retrieval provides additional reasoning capacity for complex questions. EA-GraphRAG reduces unnecessary graph-retrieval cost compared with always using graph-based retrieval: the full system runs at 2.19 s/query, compared with 3.23 s/query for graph-based retrieval. The variant without fusion further reduces latency to 1.14 s/query. This shows that the two thresholds τ_L and τ_H provide a tunable accuracy–

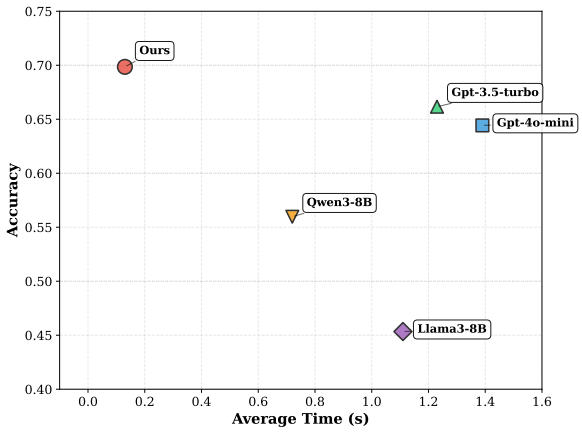

Figure 3: Accuracy and efficiency comparison of adapters.

latency trade-off by controlling how often the system invokes graph retrieval or fusion.

Retrieval strategy	Time/query (s)
dense retrieval	0.08
graph-based retrieval	3.23
Ours w/o fusion	1.14
Ours	**2.19**

Table 4: Average retrieval time per query (seconds).

4.6 Analysis of the Query Adapter

We analyze whether syntactic query complexity provides an effective and efficient routing signal. Figure 3 compares our syntax-aware MLP adapter with representative LLM-based semantic routers. For the LLM-based routers, we follow the query classification prompt template from (Han et al., 2025); the full prompt is provided in Appendix D.

The proposed adapter achieves a better accuracy–latency trade-off, because it estimates query complexity from explicit syntactic and structural features without invoking a heavyweight LLM for every query. This is important for adaptive RAG, where the routing module itself should not introduce substantial online cost.

The results also support the effectiveness of syntactic complexity for routing between RAG and GraphRAG. Compared with LLM-based semantic routers, our adapter provides a lightweight and interpretable approximation of whether a query is likely to require graph-based reasoning. Queries with richer clause structures, longer dependencies, more entity constraints, or coordination patterns tend to receive higher complexity scores and are more likely to be routed to graph retrieval or fusion. In contrast, simpler factual queries are more often assigned to dense retrieval. This behavior explains why EA-GraphRAG can reduce unnecessary graph traversal while preserving the benefits of graph retrieval for complex queries.

4.7 Ablation Study

Setting	Acc.	GPT-Acc.
GPT-4o-mini w/o retrieval	39.5	43.4
+ Dense retrieval	65.1	70.1
+ Graph-based retrieval	70.5	75.9
+ Fusion module	**71.6**	**76.9**

Table 5: Ablation on the mixed benchmark.

Table 5 reports the ablation results on the mixed benchmark. Dense retrieval provides a substantial gain over the generator-only baseline, confirming the importance of grounding generation with external evidence. Adding graph-based retrieval further improves Acc. from 65.1 to 70.5 and GPT-Acc. from 70.1 to 75.9, indicating that graph structure provides complementary relational evidence beyond local dense matching. The full model with the fusion module achieves the best result, reaching 71.6 Acc. and 76.9 GPT-Acc. This suggests that hard routing is not always sufficient for boundary cases, where both dense semantic matching and graph-derived evidence may be useful. Thus, the fusion path improves robustness under mixed query distributions by providing a soft fallback when the adapter is less certain.

5 Conclusion

We proposed EA-GraphRAG, a syntax-aware adaptive framework for integrating RAG and GraphRAG. Instead of applying graph-based retrieval to all queries or using an LLM-based router, EA-GraphRAG estimates query complexity with lightweight syntactic and structural features, and routes queries to dense retrieval, graph-based retrieval, or complexity-aware fusion. Experiments on single-hop, multi-hop, and mixed QA benchmarks show that EA-GraphRAG improves answer accuracy and retrieval recall while reducing unnecessary graph-retrieval cost compared with always using graph-based retrieval. These results suggest that syntactic query complexity provides an efficient and interpretable signal for balancing dense retrieval and graph-based retrieval under heterogeneous query distributions.

Limitations

Although the fusion path can handle some uncertain cases where syntactic features are insufficient for routing, syntactic complexity still only partially reflects query complexity. Some syntactically simple queries may require evidence from rare entities or multiple passages, while some syntactically complex queries may be answered by a single relevant passage. Thus, syntax-aware routing may not capture all factors that determine whether graph-based retrieval is needed.

Future work will explore additional complexity measurements, especially retrieval complexity, such as retrieval-score uncertainty, evidence dispersion, entity coverage, and graph connectivity among candidate passages. Combining syntactic and retrieval-side signals may enable more precise routing among dense retrieval, graph-based retrieval, and fusion.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avi Sil, and Hannaneh Hajishirzi. 2024. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *International conference on learning representations*, volume 2024, pages 9112–9141.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.

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*.

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

Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. 2025. Lightrag: Simple and fast retrieval-augmented generation. *Preprint*, arXiv:2410.05779.

Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2024. Hipporag: Neurobiologically inspired long-term memory for large language models.

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. *arXiv preprint arXiv:2502.14802*.

Haoyu Han, Harry Shomer, Yu Wang, Yongjia Lei, Kai Guo, Zhigang Hua, Bo Long, Hui Liu, and Jiliang Tang. 2025. Rag vs. graphrag: A systematic evaluation and key insights. *arXiv preprint arXiv:2502.11371*.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. 2024. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. *arXiv preprint arXiv:2402.07630*.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. *arXiv preprint arXiv:2011.01060*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2023. Atlas: Few-shot learning with retrieval augmented language models.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C Park. 2024. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. *arXiv preprint arXiv:2403.14403*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, and 1 others. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks.

Xiaofei Lu. 2010. Automatic analysis of syntactic complexity in second language writing. *International journal of corpus linguistics*, 15(4):474–496.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. *arXiv preprint arXiv:2212.10511*.

Jinghui Ouyang, Jingyang Jiang, and Haitao Liu. 2022. Dependency distance measures in assessing l2 writing proficiency. *Assessing Writing*, 51:100603.

Tanmay Parekh, Pradyot Prakash, Alexander Radovic, Akshay Shekher, and Denis Savenkov. 2025. Dynamic strategy planning for efficient question answering with large language models. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 6038–6059.

Boci Peng, Yun Zhu, Yongchao Liu, Xiaohe Bo, Haizhou Shi, Chuntao Hong, Yan Zhang, and Siliang Tang. 2024. Graph retrieval-augmented generation: A survey. *arXiv preprint arXiv:2408.08921*.

Tyler Thomas Procko and Omar Ochoa. 2024. Graph retrieval-augmented generation for large language models: A survey. In *Conference on AI, Science, Engineering, and Technology (AIxSET)*.

Stephen E Robertson and Steve Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR'94: Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, organised by Dublin City University*, pages 232–241. Springer.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2022. Colbertv2: Effective and efficient retrieval via lightweight late interaction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3715–3734.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. 2024. Raptor: Recursive abstractive processing for tree-organized retrieval. In *International Conference on Learning Representations (ICLR)*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10014–10037.

Shu Wang, Yixiang Fang, Yingli Zhou, Xilin Liu, and Yuchi Ma. Archrag: attributed community-based hierarchical retrieval-augmented generation (2025). *URL https://arxiv. org/abs/2502.09891*.

Yu Wang, Nedim Lipka, Ryan A Rossi, Alexa Siu, Ruiyi Zhang, and Tyler Derr. 2024. Knowledge graph prompting for multi-document question answering.

Zhishang Xiang, Chuanjie Wu, Qinggang Zhang, Shengyuan Chen, Zijin Hong, Xiao Huang, and Jinsong Su. 2025. When to use graphs in rag: A comprehensive analysis for graph retrieval-augmented generation. *arXiv preprint arXiv:2506.05690*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering.

Zijun Yao, Weijian Qi, Liangming Pan, Shulin Cao, Linmei Hu, Liu Weichuan, Lei Hou, and Juanzi Li. 2025. Seakr: Self-aware knowledge retrieval for adaptive retrieval augmented generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 27022–27043.

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

Xiaopeng Zhang and Xiaofei Lu. 2022. Revisiting the predictive power of traditional vs. fine-grained syntactic complexity indices for l2 writing quality: The case of two genres. *Assessing Writing*, 51:100597.

Yanqiu Zhang, Zhen Xu, DongDong Huo, Xiaokun Guo, Qihui Zhou, and Yan Zhang. 2025b. Adagcrag: Adaptive graph-chunk retrieval for lightweight rag. In *International Semantic Web Conference*, pages 594–610. Springer.

Yingli Zhou, Yaodong Su, Youran Sun, Shu Wang, Taotao Wang, Runyuan He, Yongwei Zhang, Sicong Liang, Xilin Liu, Yuchi Ma, and 1 others. 2025. In-depth analysis of graph-based rag in a unified framework. *arXiv preprint arXiv:2503.04338*.

A Additional Experiments

A.1 Case Study

Table 6 presents two representative examples that illustrate why EA-GraphRAG outperforms using either retrieval paradigm alone. For the single-hop factoid "*When is season 8 for game of thrones?*," the graph-based retriever HippoRAG places excessive emphasis on entity matches and link structure. As a result, it retrieves passages that contain relevant entities but do not include the local sentence-level evidence specifying the release date, which introduces off-target context and leads to an incorrect answer. Dense retrieval, by prioritizing sentence- and passage-level semantic similarity, directly surfaces passages stating that the season "premiere[s] in April 2019," which is sufficient to answer the question.

For the multi-hop query "*What is the date of death of the director of The Organization (Film)?*," dense retrieval tends to return generic director biographies and fails to recover the required evidence chain from the film to its director Don Medford and then to his death date. In contrast, the graph-based path uses entity linking and diffusion over the graph to traverse these relations and retrieve evidence containing "December 12, 2012." EA-GraphRAG addresses these complementary failure modes by routing simple factoids to dense retrieval and compositional multi-hop questions to graph-based retrieval, while reserving the fusion path for ambiguous cases. Together, these examples show that dynamic routing enables a single framework to consistently retrieve appropriate evidence across both simple and complex queries.

A.2 Additional Analysis of the Query Adapter

We further analyze the behavior of the syntax-aware adapter by visualizing its complexity scores on the test set. The score $s(q)$ reflects the adapter's preference for graph-based retrieval, where higher values indicate that the query is more likely to benefit from graph-derived evidence.

As shown in Figure 4, single-hop queries receive lower scores on average, while multi-hop queries are generally assigned higher scores. This suggests that the adapter captures complexity signals that are relevant to retrieval routing. The single-hop distribution is more concentrated because such queries often follow stable factual patterns and can be answered from localized evidence. In contrast, multi-hop queries are more diverse: some explicitly contain relational chains or comparison structures, whereas others are syntactically simple but require implicit entity linking or evidence aggregation across passages. Therefore, their scores cover a wider range.

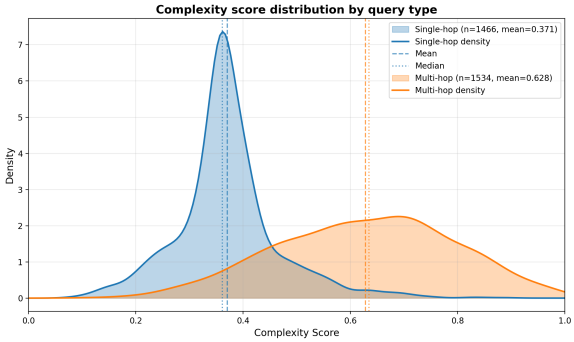

Figure 4: Distribution of MLP adapter complexity scores on the test set for single-hop and multi-hop queries.

The partial overlap between the two distributions shows that query complexity is not strictly binary. This supports our tri-routing design, where the fusion path serves as a soft fallback for cases that cannot be reliably assigned to either dense retrieval or graph-based retrieval alone..

B Details of Query Complexity Features

This appendix provides the detailed feature groups used by the query featurizer $\phi(q)$. We use syntactic and structural features as the primary signals for query complexity, and include lightweight lexical and entity-level features as auxiliary indicators.

B.1 Constituency-based Syntactic Complexity Features

We extract constituency-based syntactic complexity features following linguistic measures based on clauses, T-units, dependent clauses, coordination, and complex nominals. Specifically, we parse each query with Stanza and identify syntactic units using Tregex patterns. The extracted units include words W, sentences S, clauses C, dependent clauses DC, T-units T, complex T-units CT, coordinate phrases CP, complex nominals CN, and verb phrases VP.

Based on these counts, we compute the following ratio-based measures:

- Length of production units: $MLC = W/C$, $MLS = W/S$, and $MLT = W/T$.
- Sentence complexity: C/S.

Single-hop Question	When is season 8 for game of thrones?
Ground Truth	2019
GraphRAG	*Retrieved context:* 1. ✗ "Game of Thrones": Game of Thrones Game of Thrones is an American fanta... 2. ✗ "The Walking Dead (season 8)": marked the first crossover between the two series... ... *Prediction:* ✗ The text does not provide a specific date for Season 8 of Game of Thrones.
EA-GraphRAG	*Retrieved context:* 1. ✓ "Game of Thrones (season 8)": ...to premiere in **April 2019**. Ramin Djawadi... 2. ✓ "Game of Thrones (season 8)": ...The season is scheduled to premiere in **April 2019**... ... *Prediction:* ✓ April 2019.
Multi-hop Question	What is the date of death of the director of film The Organization (Film)?
Ground Truth	December 12, 2012
Vanilla RAG	*Retrieved context:* 1. ✗ "Lino Brocka": Catalino Ortiz Brocka (April 3, 1939 – May 22, 1991) was a Filipino film director... 2. ✗ "Wallace Fox": Wallace Fox (March 9, 1895 – June 30, 1958) was an American film director... ... *Prediction:* ✗ Not provided in the text.
EA-GraphRAG	*Retrieved context:* 1. ✓ "The Organization (film)": ...**The Organization** is a 1971 DeLuxe Color American crime thriller film starring Sidney Poitier as Virgil Tibbs and directed by **Don Medford**... 2. ✓ "Don Medford": ...**Donald Muller** (November 26, 1917 – **December 12, 2012**)... ... *Prediction:* ✓ December 12, 2012.

Table 6: Cases comparing Vanilla RAG, GraphRAG and EA-GraphRAG.

- Subordination: C/T, CT/T, DC/C, and DC/T.
- Coordination: CP/C, CP/T, and T/S.
- Particular structures: CN/C, CN/T, and VP/T.

These measures characterize how much syntactic structure is required to express the query. Queries with more clauses, dependent clauses, coordination, or complex nominals are more likely to involve compositional reasoning and thus may benefit from graph-based retrieval.

B.2 Dependency-based Structural Features

We also extract dependency-based features using SpaCy dependency parses. These features capture structural relations that are not fully described by constituency ratios. They include:

- Maximum and average dependency distance.
- Number of long-range dependencies.
- Counts of subject–verb, object–verb, modifier, coordination, and subordination relations.
- Dependency tree imbalance, computed from the relative distribution of left and right children of the root.

Long dependency distances and richer dependency relations may indicate that the query contains multiple linked entities or relational constraints, which are useful signals for routing.

B.3 Lightweight Lexical and Entity-level Features

In addition to syntactic features, we use lightweight lexical and entity-level features as auxiliary signals. These features include:

- Basic length statistics: character length, number of tokens, average token length, and number of sentences.
- POS-based counts and ratios: nouns, verbs, adjectives, adverbs, pronouns, determiners, prepositions, conjunctions, content-word ratio, and function-word ratio.
- WH-question features: number of WH words, WH-word position, encoded first WH word,

Prompt for Query Classification

System Prompt. Classifying queries into fact-based and reasoning-based categories.
You are an AI model tasked with classifying queries into one of two categories based on their complexity and reasoning requirements.
Category Definitions
1. Fact-Based Queries
- The answer can be directly retrieved from a knowledge source or requires only factual details.
- The query does not require multi-step reasoning, inference, or cross-referencing multiple sources.
2. Reasoning-Based Queries
- The answer cannot be found through a single lookup.
- The query requires cross-referencing multiple sources, logical inference, or multi-step reasoning.
Examples
Fact-Based Queries
{{ Fact-Based Queries Examples }}
Reasoning-Based Queries
{{ Reasoning-Based Queries Examples }}

Figure 5: Prompt used for query classification.

and binary indicators for *what*, *who*, *when*, *where*, *why*, *how*, and *which* questions.

- Named entity features: number of entities, entity density, number of entity types, and counts of person, organization, location, and date entities.
- Logical and complexity markers: logical connector counts, coordination, subordination, relative clause, negation, passive voice, and question mark indicators.
- Lexical diversity and information density: unique token count, lexical diversity, content-word count, and information density.

These features provide low-cost auxiliary evidence about the surface form, entity load, and question type of the query.

B.4 Tree-level and Interaction Features

We further compute global tree-structure and interaction features:

- Tree depth, tree width, number of leaf and non-leaf nodes.
- Leaf ratio, branching factor, depth-to-width ratio, and depth per token.
- Tokens per clause, entities per token, and connectors per clause.

These interaction features combine syntactic structure with query length and entity density, providing a compact representation of how much structural information is packed into the query.

B.5 Feature Selection and Normalization

Let $\mathbf{x} \in \mathbb{R}^{d'}$ denote the raw feature vector after concatenating all feature groups. We compute mutual information between each feature and the routing label on the training set, and keep the top-k features. In our implementation, k is set to $\min(85, d')$. The selected features are then standardized using the training-set mean and standard deviation:

$$\mathbf{x} = (\mathbf{x} - \boldsymbol{\mu}) \oslash \boldsymbol{\sigma}.$$

The same selected feature indices and normalization statistics are applied during inference.

C Details of the MLP Adapter

The adapter g_θ maps the selected query feature vector $\phi(q) \in \mathbb{R}^d$ to a scalar complexity score $s(q) \in (0, 1)$. This appendix describes the architecture and training details.

C.1 Architecture

Our MLP incorporates feature attention and residual connections. The model first applies feature-level attention to the input, then passes through hidden layers with dimensions 256, 128, and 64, with residual connections where dimensions match. The final hidden representation is processed through an output attention layer before computing the logit $\ell_\theta(\mathbf{x})$, where ℓ_θ denotes the MLP function parameterized by θ. The complexity score is:

$$s(q) = \sigma(\ell_\theta(\phi(q))).$$

C.2 Training

We form a disagreement training set $\mathcal{T} = \{i : z_i^{\text{rag}} \oplus z_i^{\text{gr}} = 1\}$, where $z_i^{\text{rag}}, z_i^{\text{gr}} \in \{0, 1\}$ are bi-

nary indicators of whether RAG and GraphRAG answer sample i correctly as measured by automatic QA metrics, and $\oplus$ denotes exclusive OR. Each sample $i \in \mathcal{T}$ is labeled $y_i = 1$ if GraphRAG is correct and RAG is incorrect, and $y_i = 0$ if RAG is correct and GraphRAG is incorrect. Samples where both methods agree are excluded from training. We minimize BCE-with-logits loss with label smoothing using AdamW optimizer with cosine annealing learning rate schedule.

C.3 Threshold Selection

The thresholds τ_L and τ_H are tuned on the validation set. Lower values of τ_H activate graph-based retrieval more frequently and may improve recall for complex questions but increase latency. Higher values of τ_L route more queries to dense retrieval and improve efficiency, but may miss relational evidence. The fusion interval (τ_L, τ_H) provides a buffer for uncertain cases, reducing the risk of hard routing errors.

D Prompt for Query Classification

For the LLM-based routers analyzed in Section 4.6, we follow the query classification prompt template from (Han et al., 2025). The full prompt is shown in Figure 5.