PortfoliQA: An Agentic RAG Framework for Knowledge Graph Question Answering via Structured Evidence Portfolios

Weina Zhang
Shanghai University of Electric Power
Shanghai, China
mszhangwn@shiep.edu.cn

Junsheng Huang
Shanghai University of Electric Power
Shanghai, China
jenson@mail.shiep.edu.cn

Zhongqin Bi*
Shanghai University of Electric Power
Shanghai, China
zqbi@shiep.edu.cn

Dan Dai
Aston University
Birmingham, UK
d.dai@aston.ac.uk

ABSTRACT

While Retrieval-Augmented Generation (RAG) has shown promise in grounding Large Language Models (LLMs) in external knowledge, traditional workflows are often static and linear, limiting their effectiveness on complex reasoning tasks over Knowledge Graphs (KGs). Agentic RAG has emerged as a more powerful paradigm, employing autonomous agents for dynamic reasoning. However, for Agentic RAG to be effective on structured KGs, it must overcome critical challenges from traditional approaches: semantic parsing for question that relies on fixed templates, fact collection that is limited by local, one-hop decisions, and a final reasoning step performed over an undifferentiated set of facts. To address these challenges, we propose PortfoliQA : An Agentic RAG Framework for Knowledge Graph Question Answering via Structured Evidence Portfolios. PortfoliQA employs a Planner agent for flexible, constraint-centric semantic parsing to avoid fixed templates; a swarm of Aligner agents that uses a dual-signal, dual-pool search strategy to mitigate the risks of local decision-making during fact collection; and a final LLM Reasoner that evaluates structured Evidence Portfolios, transforming the opaque reasoning process into a transparent assessment. Evaluated on the complex WebQSP and CWQ benchmarks, PortfoliQA achieves highly competitive results. Our analysis shows that this portfolio-centric workflow enhances the accuracy and explainability of complex question-answering tasks on KGs.

KEYWORDS

Agents; Explainable AI; Knowledge Graph; RAG

ACM Reference Format:
Weina Zhang, Junsheng Huang, Zhongqin Bi, and Dan Dai. 2026. PortfoliQA: An Agentic RAG Framework for Knowledge Graph Question Answering via Structured Evidence Portfolios. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 19 pages. https://doi.org/10.65109/VPTX2262

*Corresponding author.

This work is licensed under a Creative Commons Attribution International 4.0 License.

Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus. © 2026 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). https://doi.org/10.65109/VPTX2262

1 INTRODUCTION

Large Language Models (LLMs) have demonstrated remarkable capabilities in complex reasoning tasks [9, 21]. However, their reliability is often undermined by critical issues such as factual hallucination [10] and knowledge obsolescence [12]. Retrieval-Augmented Generation (RAG) has emerged as a potent paradigm to mitigate these limitations by providing LLMs with external knowledge sources [6, 25]. Among these sources, Knowledge Graphs (KGs), with their structured representation of entities and relationships, offer a more efficient and robust foundation for RAG, making KG-based RAG a promising frontier [20, 29].

However, traditional RAG workflows, including most KG-RAG implementations, are often static and linear, which limits their ability to handle multi-step reasoning and complex tasks [22]. To overcome these limitations, a more powerful paradigm, **Agentic RAG**, has emerged. By embedding autonomous AI agents into the RAG pipeline, Agentic RAG leverages their capabilities for planning, tool use, and collaboration to achieve more dynamic, iterative, and intelligent reasoning [22].

Nevertheless, for the Agentic RAG paradigm to unlock its full potential on structured KGs, it must first overcome several fundamental challenges in traditional KG-RAG approaches. A primary challenge lies in **semantic parsing for question**, many existing frameworks require pre-determining a fixed reasoning template for the question [4, 5, 30]. This approach fails when faced with complex questions that require satisfying multiple, non-linear constraints. A second challenge emerges during **fact collection** over the graph; it requires step-by-step graph traversal, but agents often focus only on one-hop neighbors when making local decisions [18, 23, 27]. This may cause them to abandon a potentially correct path simply because the next step appears unpromising. Finally, the third challenge occurs in the **reasoning phase**, where retrieved facts or paths are typically presented to the final reasoning module as a undifferentiated set of facts [8, 13, 16]. This severs the explicit link between a candidate answer and its specific evidence chain for each logical constraint, rendering the final decision an opaque "black box" and undermining the system's explainability.

To systematically address these challenges, we propose **PortfoliQA**: An Agentic RAG Framework for Knowledge Graph Question Answering via Structured Evidence Portfolios. To address the issue of semantic parsing, our **Planner Agent** first decomposes the

question into atomic logical fragments, then uses "query construction tool" to build a "constraint-centric" query plan without the need for fixed reasoning template. To overcome the risks of local decision-making during fact collection, our **Aligner Agents** employ a two-tiered strategy: the search is guided by a scoring function that balances local structural relevance with global semantic intent, and a dual-pool search architecture prevents the premature pruning of paths. Finally, to address the opacity of reasoning, we introduce the "Evidence Portfolio". The Aligner agents associate candidate answers with their evidence chains under all logical constraints. This transforms the final reasoning task from an induction over a "collection of facts" to a transparent, evidence-based evaluation of clear, comparable "Evidence Portfolios" by LLM reasoner.

Our main contributions are summarized as follows:

- We propose PortfoliQA, a novel, fine-tuning-free Agentic RAG framework that reframes KGQA into a more robust and explainable process by introducing the "Evidence Portfolio".
- We design a flexible, agent-based semantic parsing method that, through "atomic decomposition" and "deterministic reconstruction," can unify arbitrarily complex, multi-constraint questions into a single query framework.
- We introduce a graph exploration strategy, combining a dual-signal scoring function and Dual-Pool search architecture, which collectively mitigates the risks of local decision-making during fact collection.
- We demonstrate through comprehensive experiments that our framework significantly enhances the accuracy and explainability of complex question-answering tasks, proving the benefits of agentic AI in the KGQA scenario.

2 RELATED WORK

In this section, we analyze the shortcomings of existing methods, which motivate the need for an agentic framework designed specifically for the structured properties of knowledge graphs.

2.1 KG-enhanced LLM Reasoning

Existing KG-RAG methods can be broadly classified by the granularity of their retrieved knowledge into **triple-based** and **path-based** approaches. Path-based methods, such as RoG [16] and the agent-based ToG [23], focus on retrieving relevant knowledge paths. In contrast, triple-based methods, like SubgraphRAG [13] and G-Retriever [8], aim to retrieve a set of relevant triples. Despite their different retrieval strategies, these methods face a common challenge in the final reasoning stage. They typically present the retrieved set of factual triples or paths to the LLM as a monolithic context for indiscriminate inductive reasoning. In this paradigm, the LLM is confronted with a flat "collection of facts," where the explicit logical association between a candidate answer and its specific, structured evidence chain is often lost. This presents two major difficulties: first, it significantly increases the cognitive load on the LLM; second, it renders the final reasoning step an opaque "black box," greatly diminishing the system's explainability.

2.2 Semantic Parsing for KGQA

One research area in KGQA is translating a natural language question into a formal, machine-executable representation— **semantic parsing**. A dominant approach within this area requires the LLM to commit to a pre-defined reasoning structure. For instance, some methods require a preliminary classification of the question's overall structure (e.g., "chain" or "parallel" in PDRR [30]), while others tend to generate linear, fixed-structure reasoning path templates (e.g., KARPA [5] and DRKG [4]). The effectiveness of these methods is highly contingent on the correctness of this initial structural assumption, making them less suited for complex queries that involve multiple, non-linear constraints. Furthermore, other methods employ a more flexible "atomic decomposition" [15], breaking the question into the smallest possible logical units. However, their core challenge then shifts to how to reliably re-compose the results from these atomized fragments to satisfy the question's holistic logic.

2.3 Agentic Exploration of Knowledge Graphs

Another research area encapsulates the LLM as an autonomous agent that dynamically explores reasoning paths through iterative interaction with the KG. The early work ToG [23], as well as more advanced variants like R2-KG [11] and BYOKG [18], all empower the agent with the ability to use tools and decide on the next action at each step. The core advantage of such methods lies in their exploratory flexibility. However, this step-by-step decision-making process, often focused on one-hop neighbors, is heavily guided by the LLM's inherent understanding of "ideal logic." This presents a challenge: when the actual structure of the KG deviates from this ideal, the agent's internal model may cause it to overlook valid reasoning paths that do not conform to its pre-conceived logical patterns.

3 PRELIMINARIES

In this section, we formally define the core concepts and notations used throughout our PortfoliQA framework.

3.1 Problem Formulation and Definitions

The primary task of Knowledge Graph Question Answering (KGQA) is to find a set of correct answer entities from a large knowledge graph based on a natural language question. Formally, we define the core components and the task as follows:

Knowledge Graph (KG). A Knowledge Graph is a directed multi-graph denoted as $G = (\mathcal{E}, \mathcal{R}, \mathcal{T})$, where $\mathcal{E}$ is a set of entities, $\mathcal{R}$ is a set of relations, and $\mathcal{T} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ is a set of factual triples. Each triple $t = (h, r, t') \in \mathcal{T}$ represents a fact from a head entity h to a tail entity t' connected by a relation r.

Reasoning Path. A reasoning path π over the KG is a sequence of L triples, $\pi = (t_1, t_2, ..., t_L)$, where any two adjacent triples t_i and t_{i+1} share a common entity.

Given these definitions, the task can be formalized: for a given natural language question Q containing one or more topic entities $\mathcal{E}_Q \subset \mathcal{E}$, the goal is to find the set of answer entities $\mathcal{A}_Q \subset \mathcal{E}$ from the knowledge graph G.

3.2 Core Concepts of PortfoliQA

Our framework translates the unstructured question Q into a series of structured representations for planning and execution. Let $\mathcal{V}$ be a set of logical variables, denoted with a '?' prefix (e.g., ?film).Among

these variables, one is designated as the **Target Variable** ($v_{target} \in \mathcal{V}$), which represents the information the question seeks to find.
Logical Fragment. A logical fragment F_i from the set of all fragments $\mathcal{F}$ is an atomic statement of the form $F_i = (s, p, o)$, where the subject s and object o can be either a topic entity $e \in \mathcal{E}_Q$ or a logical variable $v \in \mathcal{V}$. The predicate p is a natural language phrase representing an abstract relation.
Variable Binding. A variable binding is a tuple (v, e), where a logical variable $v \in \mathcal{V}$ is mapped to a concrete entity $e \in \mathcal{E}$ from the KG.
Constrained Logical Query Graph (CLQG). A Constrained Logical Query Graph is an undirected graph $G_L = (\mathcal{V}_L, \mathcal{E}_L)$, where the node set $\mathcal{V}_L = \mathcal{E}_Q \cup \mathcal{V}$ consists of all topic entities and logical variables from the question. The edge set $\mathcal{E}_L$ contains an edge for each logical fragment $F_i \in \mathcal{F}$, connecting the nodes corresponding to its subject and object, thereby capturing the constraint relationships between them.
Query Plan. A Query Plan P is an ordered tuple of constraints, $P = (c_{main}, \langle c_{side,1}, ..., c_{side,N} \rangle)$.
Constraint. A constraint c is defined by an `logical_path`, which describes the logical chain that needs to be executed, (e.g., `Tom Hanks -> stars -> ?film`). We define two types of constraints based on their endpoint:

- A **Generative Constraint** has a logical variable as its endpoint, requiring a search for unknown entities.
- A **Verificative Constraint** has a constant entity or an already-bound variable as its endpoint, requiring a verification of a known connection.

Evidence Portfolio. An Evidence Portfolio $\mathcal{H}$ is the central data structure. It is a collection of Evidence Sets, where each set corresponds to a unique candidate answer. An Evidence Set for a candidate $e_{ans} \in \mathcal{E}$ is a mapping from each constraint $c_i \in P$ to its corresponding fulfilled evidence path π_i or a `NO_EVIDENCE_FOUND` tag. Formally, $\mathcal{H} = \{(e_{ans,1}, L_1), (e_{ans,2}, L_2), ...(e_{ans,n}, L_n), \}$, where each $L_j = \{(c_1, \pi_{j,1}), (c_2, \pi_{j,2}), ...(c_n, \pi_{j,n})\}$.

4 METHODOLOGY

PortfoliQA reframes complex Knowledge Graph Question Answering (KGQA) as an agentic workflow. As illustrated in Figure 1, our framework is composed of two core, specialized agents that collaborate to generate an Evidence Portfolio for natural language questions. The workflow is orchestrated through a series of structured data objects, with each agent performing a distinct role using specific tools. The agent role is summarized in Table 1.

4.1 Initialization

Before the agentic workflow begins, the system first prepares the reasoning environment. Given a question Q, we first delineate a relevant knowledge scope. The extraction of topic entities $\mathcal{E}_Q$ follows prior work [13, 16], using a combination of efficient semantic matching and LLM-assisted grounding. Subsequently, we use the **SubgraphRAG** [13] module to retrieve a local subgraph G_{sub}, centered around these topic entities and containing N highly relevant triples from the large-scale KG G. This subgraph serves as the foundational environment for all subsequent agent actions.

4.2 Planner Agent: Generate Query Plan

The first agentic phase is orchestrated by the **Planner Agent** (α_p), whose goal is to transform the unstructured question Q into a robust, executable plan P. This process is illustrated in Figure 2.

Semantic Parsing. The workflow begins with the Planner Agent (α_p) utilizing a **`semantic_parser`** tool, which leverages an LLM to decompose the question Q into a set of atomic, independent **Logical Fragments ($\mathcal{F}$)**. In this initial step, the semantic_parser is also tasked with identifying the **Target Variable** (v_{target}) from the question's phrasing. Critically, the agent then uses a **`validation_tool`** to assemble the generated fragments into a preliminary CLQG and check for structural properties, such as connectivity. If the graph is found to be invalid, the agent enters a self-correction loop, providing specific feedback (e.g., "Fragment F_2 is disconnected from the main graph") to the `semantic_parser`, which then re-prompts the LLM to generate a revised set of fragments. This iterative refinement continues until a valid CLQG is formed. To guarantee system termination, if the loop fails after a set number of attempts, the agent employs a fallback strategy, ensuring at least one basic main chain from a topic entity to the target variable is constructed.

Query Plan Construction. Once a valid CLQG is available, the Planner Agent utilizes a deterministic **`construction_tool`** to generate the final Query Plan. This process involves three steps, as shown in Algorithm 1 in the Appendix.

(1) **CLQG Construction:** The tool assembles the fragments $F_i \in \mathcal{F}$ into a CLQG (G_L). This graph serves as a structured blueprint of the question's logic, where entities ($\mathcal{E}_Q$) and variables ($\mathcal{V}$) form the vertices, and the logical fragments themselves define the edges connecting them.
(2) **Main Chain Determination:** To establish an execution order, the process begins by identifying all simple paths within the CLQG that connect a topic entity $e \in \mathcal{E}_Q$ to the target variable v_{target}. Each of these paths is considered a **potential main chain**. It then employs a **Main Chain Determination Function** to select one of these candidates as the main generative constraint (c_{main}). The determination is based on the product of the **Expected Fan-Out (EFO)** of the abstract predicates p_i along each potential chain. Fan-out is defined as the number of unique entities connected as objects for a given relation. The EFO for an abstract predicate p is calculated as the weighted average of the fan-outs of the top-N semantically similar concrete relations r_j in the subgraph, where weights are their similarity scores:

$$\text{EFO}(p) = \frac{\sum_{j=1}^{N} \text{sim}(p, r_j) \cdot \text{FanOut}(r_j)}{\sum_{j=1}^{N} \text{sim}(p, r_j)} \tag{1}$$

The score for an entire chain c is the product of the EFOs of its predicates:

$$\text{ChainScore}(c) = \prod_{p_i \in c} \text{EFO}(p_i) \tag{2}$$

The chain with the minimum `ChainScore` is selected as the main generative constraint (c_{main}), as this heuristically corresponds to a more focused initial search.

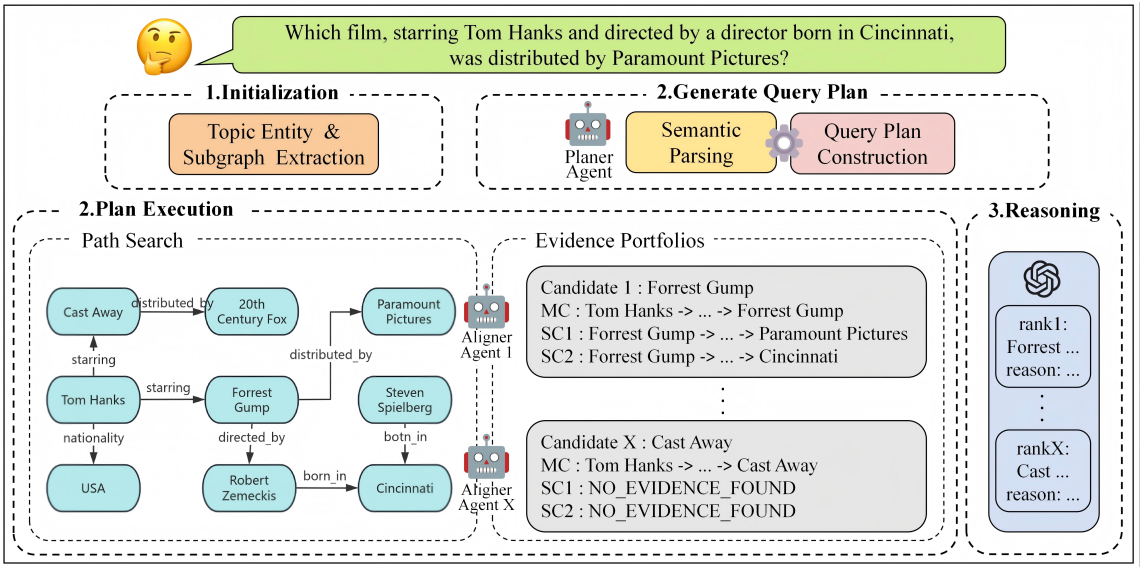

Figure 1: A flowchart of the PortfoliQA framework showing its four main stages: Initialization, Generate Query Plan, Plan Execution, and Reasoning. It depicts two types of agents (Planner and Aligner) and a final LLM Reasoner, illustrating the data flow from a natural language question to a ranked list of answers.

Table 1: The Agent of PortfoliQA.

Agent	Primary Task	Input / Output	Tools Used

Planner Agent (α_p)	Decompose the question and construct a robust, executable Query Plan.	**In:** Question Q
		Out: Query Plan P
			`semantic_parser`
			`validation_tool`
			`construction_tool`

Aligner Agents (A_s)	Execute the Query Plan by finding evidence paths and assembling them into Evidence Portfolios.	**In:** Query Plan P, Subgraph G_{sub} **Out:** Evidence Portfolio $\mathcal{H}$	`path_finding_tool` `path_verification_tool`

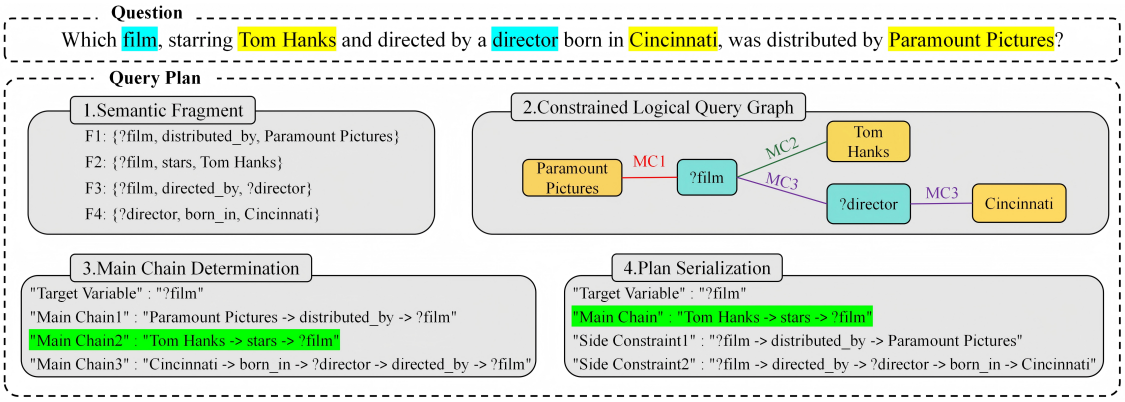

Figure 2: The Query Plan generation process. The Planner Agent first decomposes the question into Semantic Fragments. Then these are assembled into a CLQG. Finally, three potential main chains were identified and generate the final Query Plan.

(3) **Plan Serialization:** Once the main chain (c_{main}) is determined, the tool finalizes the plan by serializing the remaining parts of the CLQG. It performs a graph traversal starting from the nodes of c_{main}. All unvisited edges and paths branching off from the main chain are systematically grouped into **maximal paths**. A maximal path is a simple path that starts

at a node on the main chain and extends to a leaf node in the remaining graph component. Each of these maximal paths is then converted into a **Side Constraint** ($c_{side,i}$), forming an ordered list. The final output is the complete, ordered Query Plan $P = (c_{main}, \langle c_{side,1}, ..., c_{side,N} \rangle)$.

4.3 Aligner Agent : Plan Execution

The second phase is executed by a parallel swarm of **Aligner Agents** ($A_s = \{\sigma_1, ..., \sigma_n\}$). Their collective goal is to execute the Query Plan P, systematically find evidence for each constraint, and populate the final **Evidence Portfolio ($\mathcal{H}$)**.

The process is initiated by executing the main generative constraint (c_{main}) using the `path_finding_tool`. This initial search returns a set of k distinct, high-scoring reasoning paths. These paths are then used to **instantiate k Aligner sub-agents**. Each agent σ_i is assigned one of these initial paths, which simultaneously binds the *target variable* (v_{target}) to a potential candidate answer and forms the agent's initial **memory (b_i)**. Subsequently, all agents work in parallel, with each one's goal (g_i) being to find evidence for all side constraints $\{c_{side,i}\} \in P$ using a suite of specialized tools:

- **Path Finding Tool (τ_{find}):** This tool is invoked for *Generative Constraints*, where the endpoint is an unbound variable. A key challenge in graph search is that the logical length of a reasoning path may not match its actual length in the KG [14]. To address this, our tool's dual-pool architecture incorporates a dynamic stopping criterion. The algorithm maintains two separate pools: an **exploration pool** (the beam itself) for active path expansion, and a **completed pool** for storing any path that has reached or exceeded the expected logical length. The search does not terminate immediately upon reaching the target length. Instead, it enters an "exploratory buffer" phase, allowing the exploration pool to continue expanding. This exploratory phase terminates only when the top-k ranked paths within the completed pool remain stable for a full search iteration, indicating that further exploration is no longer yielding better solutions. The final result consists of the top-k paths from this stable completed pool. This "explore-while-harvesting" architecture ensures that promising solutions are safely collected and not prematurely pruned, while flexibly adapting to the true path lengths within the graph, as shown in the Algorithm 2 in Appendix.
- **Path Verification Tool (τ_{verify}):** This tool is invoked for *Verificative Constraints*, where both the start and end points are known (either as constants or from a previous binding). As this is a point-to-point search, it is possible to directly search for the path with the highest score between the start point and the end point, with its length matching the one specified by the logical_path.

Path Scoring Function. Both tools are guided by a sophisticated **dual-signal heuristic scoring function**, designed to evaluate the quality of a reasoning path π. This function holistically balances two critical aspects of a good path: its local factual soundness and its global semantic relevance.

To ensure a fair trade-off between these two signals, which have different distributions and scales, we first normalize their raw scores into comparable log-probabilities via **Log-Rank Transformation**. For a given scoring dimension, all candidate paths at a step are ranked (from best to worst, with the top candidate assigned rank 0), and a candidate's score is converted as follows:

$$S_{log} = -\log(\text{rank} + 1) \tag{3}$$

The two primary signals are defined as:

- $Score_{\text{struct}}(t)$: The **Structural Score**, representing the relevance of the factual triple t to the question, as provided by SubgraphRAG [13].
- $Score_{\text{sem}}(\pi, c)$: The **Semantic Score**, representing the embedding semantic similarity between the path π and its target constraint c.

Our final heuristic score is the sum of a path quality score and a guidance score. The **path quality score (S_{path})** measures the local factual soundness. It is the cumulative sum of the log-rank transformations of the structural scores of each triple t_j along the path:

$$S_{\text{path}}(\pi) = \sum_{t_j \in \pi} S_{\log}(Score_{\text{struct}}(t_j)) \tag{4}$$

The **guidance score (S_{guidance})** measures the global semantic relevance. It is directly determined by the log-rank transformation of the path's overall semantic score:

$$S_{\text{guidance}}(\pi) = S_{\log}(Score_{\text{sem}}(\pi, c)) \tag{5}$$

The final score for a path π, which balances these two components, is then calculated as:

$$\text{TotalScore}(\pi) = S_{\text{path}}(\pi) + S_{\text{guidance}}(\pi) \tag{6}$$

Iterative Alignment and Portfolio Assembly. The alignment process is iterative. After an Aligner agent executes a constraint (using either τ_{find} or τ_{verify}), the tool returns a top-k list of candidate evidence paths from the completed pool. The agent then selects the single best path from this list. Following this, the agent immediately performs the **alignment process**: it maps the concrete entities from the chosen evidence path to the corresponding logical variables in the constraint, updating its internal **memory (b_i)** with these new variable bindings. To ensure factual grounding, any binding proposed that does not correspond to an actual entity on the evidence path is discarded. This process repeats for all side constraints. Once an agent has successfully executed all constraints in its plan, it performs its final action: it submits its candidate answer, along with the complete collection of evidence paths and final variable bindings, to the central **Evidence Portfolio ($\mathcal{H}$)**. The final confidence score for a complete Evidence Set L_j within the portfolio is the sum of the scores of all its constituent evidence paths:

$$\text{PortfolioScore}(L_j) = \sum_{(c_i, \pi_i) \in L_j} \text{Score}(\pi_i) \tag{7}$$

This score is used to resolve cases where multiple Aligner agents, through different reasoning paths, arrive at the same candidate answer. In such cases, only the Evidence Set with the highest `PortfolioScore` is retained, ensuring that each candidate in the final portfolio is supported by the strongest available evidence chain.

4.4 Evaluating on the Evidence Portfolio

The final phase of the PortfoliQA workflow is the evaluation of the generated Evidence Portfolios. The complete Evidence Portfolio ($\mathcal{H}$), which contains the set of candidate answers and their corresponding structured evidence chains, is presented to a large language model acting as a final **LLM Reasoner**. The reasoner's task is to holistically assess these portfolios and generate a final ranked list of answers A_Q. This assessment is guided by a set of criteria: (1) **Evidence Completeness**, assessing whether evidence was found for all required logical constraints; (2) **Evidence Quality**, considering how well the evidence path complies with this logical constraint; and (3) **Overall Plausibility**, leveraging its own vast world knowledge to help judge the final answer's reasonableness. Based on this comprehensive and explainable evaluation, the LLM reasoner generates the final ranking, completing the question-answering process.

5 EXPERIMENTS

We conducted a comprehensive evaluation of our PortfoliQA framework on two challenging, widely-recognized KGQA benchmarks: **WebQSP** [28] and **CWQ** [24]. Both datasets utilize Freebase [3] as their background knowledge graph. We compare PortfoliQA against a suite of strong and representative baseline methods, with relevant results cited from the original paper and a detailed description of these baselines available in Appendix A. We use **Hit@1** to evaluate whether the generated answer ranks first, and **F1-Score** to assess the coverage of the answer. We present results for PortfoliQA powered by two distinct LLMs, `Llama-3.1-8B` [1] and `GPT-4o-mini` [19], to demonstrate its performance across different model capabilities. Further details on the experimental setup and datasets can be found in Appendix C, and the prompts utilized in this work are available in Appendix E.

5.1 Main Results

Table 2: The performance comparison on the WebQSP and CWQ test sets. The best results in each column are highlighted in bold, and the second-best results are underlined.

Method	WebQSP		CWQ	
	Hit@1	F1	Hit@1	F1
PDRR [30]	79.20	-	62.20	-
KARPA [5]	85.30	64.50	<u>73.70</u>	56.50
DRKG [4]	88.16	**84.05**	66.99	58.04
FRAG [7]	86.70	-	68.00	-
ToG [23]	78.50	51.60	54.00	34.50
DoG [17]	<u>91.00</u>	-	56.00	-
SPLIT-RAG [27]	84.90	72.60	61.10	59.30
RRP [26]	90.00	72.50	64.50	56.50
RoG [16]	85.70	70.80	62.60	56.20
SubgraphRAG [13]	86.60	77.75	57.01	53.23
EoG [14]	85.00	74.10	70.80	<u>65.10</u>
PortfoliQA (Llama-3.1-8B)	87.09	77.08	65.21	60.50
PortfoliQA (GPT-4o-mini)	**93.64**	<u>82.76</u>	**75.29**	**68.34**

The main results are presented in Table 2. The data clearly shows that our PortfoliQA framework, particularly when its agents are powered by GPT-4o-mini, demonstrates a significant performance advantage over the compared baseline methods across both the WebQSP and the more complex CWQ datasets.

We attribute this strong performance to our agentic workflow's systematic solutions to the core challenges outlined in the Section 1.

First, to address the challenge of **semantic parsing** for question, our Planner Agent's flexible "atomic decomposition" and "deterministic reconstruction" approach allows PortfoliQA to robustly handle the complex, multi-constraint questions prevalent in CWQ, outperforming methods that rely on fixed reasoning templates like KARPA and DRKG.

Second, to overcome the risks of local decision-making during **fact collection**, our Aligner Agents employ a robust exploration strategy. The combination of our dual-signal scoring function and the Dual-Pool search architecture prevents the premature pruning of paths that iterative agent methods like ToG are susceptible to.

Finally, by introducing the **Evidence Portfolio**, PortfoliQA resolves the challenge of **reasoning opacity**. Instead of presenting the LLM Reasoner with an undifferentiated set of facts, our structured portfolios establish a clear link between each candidate and its supporting evidence for every logical constraint. This reduces the LLM's cognitive load and enhances the accuracy of its final judgment.

It is also noteworthy that PortfoliQA paired with the smaller Llama-3.1-8B model still yields highly competitive performance. This demonstrates the robustness of the PortfoliQA architecture itself in effectively structuring the reasoning task, enabling even smaller language models to achieve strong results.

5.2 Ablation Study

Analysis on the Value of PortfoliQA's components. To quantitatively verify the contribution of each component within our agentic workflow, we conducted a comprehensive ablation study(results in Table 3). We designed four configurations to deconstruct the system's capabilities: (a) a baseline using only the final LLM Reasoner for zero-shot QA; (b) our core agentic engine (Planner + Aligner agents) without the final LLM Reasoner, relying only on the quantitative `PortfolioScore`; (c) a setting where the LLM Reasoner ranks the candidate answers generated by the agents, but without access to the structured Evidence Portfolios; and (d) the full PortfoliQA framework.

Table 3: Ablation study on the values of PortfoliQA's components.

Configuration	CWQ Hit@1
(a) LLM Reasoner Only (Zero-shot)	41.18
(b) w/o LLM Reasoner (Score-based Ranking)	54.27
(c) w/o Evidence Portfolios (Candidates Only)	65.55
(d) **Full PortfoliQA Framework**	**75.29**

The study first establishes the performance of an ungrounded LLM in configuration (a). The first significant performance leap occurs in configuration (b), which represents our core agentic search

engine. This substantial improvement proves that our structured workflow, executed by the **Planner Agent** and the **Aligner agent swarm**, is far superior to naive LLM reasoning. The most critical insights emerge from the final two stages. In configuration (c), where the LLM Reasoner re-ranks the candidate list using only its internal knowledge, performance again improves significantly, demonstrating that the LLM's latent plausibility judgment is a powerful capability. However, the final and largest performance gain is unlocked in the **Full PortfoliQA** model, where the LLM Reasoner is presented with the complete **Evidence Portfolios**. This provides definitive, quantitative evidence for the value of explainability in reasoning systems. The LLM Reasoner's performance is not just based on its latent knowledge, but is synergistically enhanced when each candidate answer is explicitly linked to its corresponding evidence chain. This validates our framework's core design: a powerful evidence-gathering agentic subsystem whose output is made maximally effective by a final, evidence-aware holistic evaluation.

Analysis on the Value of Evidence Portfolios. A core claim of our work is that the final reasoning process should not operate on an undifferentiated collection of facts, but on a structured representation that preserves the logical association between a candidate answer and its evidence. To validate the contribution of our **Evidence Portfolio**, we conducted an ablation study on the CWQ dataset. We compared the performance of the final **LLM Reasoner** when presented with evidence in three different formats: (a) our full, structured **Evidence Portfolio**; (b) a **Path-based** format, where all evidence paths from the portfolio were extracted and presented as a list; and (c) a **Triple-based** format, where all unique triples from all paths were presented as a simple "bag of facts".

Table 4: Impact of the evidence format presented to the final LLM Reasoner on the CWQ dataset.

Evidence Format	CWQ Hit@1
Triple-based (Bag of Facts)	59.33
Path-based (List of Paths)	64.84
Evidence Portfolio (Ours)	**75.29**

The results in Table 4 show a stark performance difference. The structured **Evidence Portfolio** significantly outperforms both flattened formats. The performance drop from the Portfolio to the Path-based format demonstrates the importance of explicitly linking evidence paths to the logical constraints they satisfy. Without this structure, the cognitive load on the LLM Reasoner increases as it must implicitly infer which path proves which part of the question. The further decline in the Triple-based format is even more telling; by deconstructing the paths into a "bag of facts," the crucial multi-hop reasoning context is lost, forcing the reasoner to attempt the difficult combinatorial task of reassembling the logic from scratch. This experiment confirms that the Evidence Portfolio is not merely a container for facts, but a critical component that structures information in a way that maximizes the reasoning capabilities of the final LLM, directly addressing the *Reasoning Opacity* challenge.

Analysis of the Dual-Signal Scoring Function. To overcome the risks of local decision-making during fact collection, the search is guided by a scoring function that balances local structural relevance with global semantic intent. To validate the contribution of each signal, we conducted an ablation study on the CWQ dataset, with results shown in Table 5.

Table 5: Ablation study of the scoring function signals.

Configuration	CWQ Hit@1
Full Scoring Function	**75.29**
w/o Semantic Score ($Score_{\text{sem}}$)	66.56
w/o Structural Score ($Score_{\text{struct}}$)	63.34

The data reveals that both signals are indispensable and play complementary roles in guiding the Aligner agents' search tools. The **Structural Score** ($Score_{\text{struct}}$) acts as a **local guide**, ensuring that each individual step (each triple) in a reasoning path is highly relevant to the question. Removing it forces the search to proceed without this fine-grained validation, causing a catastrophic performance collapse of over 11 percentage points. Conversely, the **Semantic Score** ($Score_{\text{sem}}$) provides the **global, strategic direction**, ensuring that the entire path remains semantically coherent with the query's overall intent. Without this "compass," the search, even if composed of individually relevant steps, drifts into logically irrelevant territory, resulting in a significant performance degradation of nearly 9 points. Therefore, it is the synergistic combination of this local, step-by-step guidance and overarching global direction that allows our fact collection process to effectively mitigate the risks of local decision-making.

5.3 Hyperparameter Analysis

To determine the optimal configuration for PortfoliQA, we tuned three key hyperparameters on the CWQ validation set.

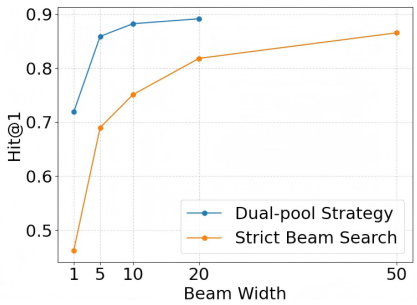

Figure 3: Impact of Beam Search Width on the CWQ validation set.

Beam Search Width. This parameter controls the size of the *exploration pool*. As shown in Figure 3, performance on our Dual-Pool strategy saturates after a width of 10, which we selected as the optimal trade-off point for our main experiments. This high efficiency is attributed to our Dual-Pool architecture, which is specifically designed to mitigate the risks of local decision-making during fact

collection. Unlike a strict search that makes irreversible pruning decisions at each step based only on the promise of the next hop, our method decouples exploration from solution collection. At each depth, any path that satisfies the target logical length is immediately moved to a separate *completed pool*, protecting it from being discarded even if its score is not high enough to remain in the exploration beam. This "explore-while-harvesting" strategy prevents the premature loss of correct paths that might be momentarily outscored by locally promising but ultimately incorrect alternatives. This architectural choice explains why our strategy with a narrow beam of 10 significantly outperforms a Strict Beam Search with a much wider beam of 50.

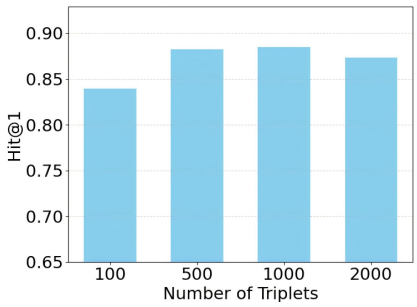

Figure 4: Impact of Subgraph Size (number of triples) on the CWQ validation set.

Subgraph Size. This parameter determines the size of the initial knowledge context retrieved by SubgraphRAG [13]. As shown in Figure 4. Performance peaks at 500 triples, which we select for our experiments. Subgraphs smaller than this risk missing critical evidence paths, while larger subgraphs introduce an excess of irrelevant triples that can misleading the path search process.

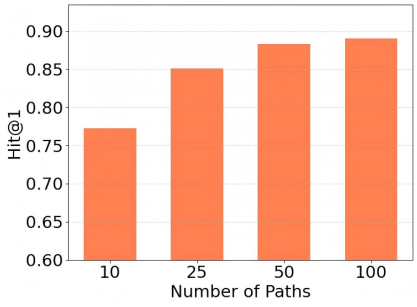

Figure 5: Impact of the number of paths retrieved from the completed pool on the CWQ validation set.

Number of Paths (Top-k). This hyperparameter governs two key aspects of the Plan Execution phase: the number of Aligner agents instantiated from the main constraint search, and the number of candidate evidence paths retrieved by the search tools (τ_{find} and τ_{verify}) for each subsequent constraint. As shown in Figure 5. We selected k=50 for our main experiments. This value represents a strong balance point, ensuring a sufficient diversity of initial agent perspectives and providing a rich set of evidence choices for each constraint, thereby capturing the vast majority of performance gains. While further increasing k to 100 yields a marginal improvement, we selected k=50 as a robust setting that effectively demonstrates the framework's capabilities.

5.4 Robustness to Knowledge Incompleteness

To empirically validate the robustness of our flexible, "constraint-centric" semantic parsing model, we conducted a targeted stress test on multi-entity questions from the CWQ dataset. We simulated a common real-world scenario of an incomplete KG by programmatically degrading the subgraphs: for each multi-entity question, we randomly retained only one of its topic entities and removed all others along with their associated triples. This guarantees that evidence for one or more logical constraints is missing from the search space. The results, presented in Table 6, demonstrate the remarkable resilience of our framework. Even under these degraded conditions, PortfoliQA maintains a high level of accuracy. This resilience is a direct result of our semantic parsing and execution model. In contrast, approaches like PDRR, which rely on a strict intersection of results from independently solved sub-problems, would catastrophically fail in such scenarios, as a single missing constraint would lead to an empty final candidate set. This highlights a core architectural advantage of PortfoliQA: by treating each logical constraint independently within the Evidence Portfolio and transforming 'missing evidence' from a failure state into an evaluable signal (i.e., the `NO_EVIDENCE_FOUND` tag), our framework can gracefully degrade and make reasoned judgments on the available, albeit incomplete, information.

Table 6: Robustness analysis on the CWQ dataset.

Question Type	# Samples	Original Hit@1	Degraded Hit@1
2 Entities	1055	86.92	77.63
>2 Entities	230	89.57	85.65
Overall	**1285**	**87.39**	**79.07**

6 CONCLUSION

In this paper, we introduced **PortfoliQA**, an agentic RAG framework for Knowledge Graph Question Answering (KGQA) that reframes the complex task into a structured evidence evaluation process. PortfoliQA addresses key challenges in existing approaches through a specialized agentic workflow: a **Planner Agent** enhances the flexibility of semantic parsing; a swarm of **Aligner Agents** improves the robustness of fact collection; and the introduction of the Evidence Portfolio resolves the opacity of the final reasoning stage. Our experiments demonstrate that this portfolio-centric methodology significantly enhances the accuracy and explainability of complex KGQA tasks. While our framework shows strong performance, future work will focus on optimizing its computational cost. Furthermore, extending the Evidence Portfolio-centric paradigm to other structured reasoning domains represent exciting directions for future research.

REFERENCES

[1] Meta AI. 2024. Llama-3.1-8B. https://huggingface.co/meta-llama/Llama-3.1-8B Accessed: 2025-10-5.

[2] Alibaba-NLP. 2024. gte-large-en-v1.5. https://https://huggingface.co/Alibaba-NLP/gte-large-en-v1.5 Accessed: 2025-10-5.

[3] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data* (Vancouver, Canada) *(SIGMOD '08)*. Association for Computing Machinery, New York, NY, USA, 1247–1250. https://doi.org/10.1145/1376616.1376746

[4] Yan Chen, Shuai Sun, and Xiaochun Hu. 2025. DRKG: Faithful and Interpretable Multi-Hop Knowledge Graph Question Answering via LLM-Guided Reasoning Plans. *Applied Sciences* 15, 12 (2025). https://doi.org/10.3390/app15126722

[5] Siyuan Fang, Kaijing Ma, Tianyu Zheng, Xeron Du, Ningxuan Lu, Ge Zhang, and Qingkun Tang. 2025. KARPA: A Training-free Method of Adapting Knowledge Graph as References for Large Language Model's Reasoning Path Aggregation. In *Findings of the Association for Computational Linguistics: ACL 2025*, Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (Eds.). Association for Computational Linguistics, Vienna, Austria, 24724–24746. https://doi.org/10.18653/v1/2025.findings-acl.1269

[6] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-Augmented Generation for Large Language Models: A Survey. arXiv:2312.10997 [cs.CL] https://arxiv.org/abs/2312.10997

[7] Zengyi Gao, Yukun Cao, Hairu Wang, Ao Ke, Yuan Feng, Xike Xie, and S Kevin Zhou. 2025. FRAG: A Flexible Modular Framework for Retrieval-Augmented Generation based on Knowledge Graphs. arXiv:2501.09957 [cs.CL] https://arxiv.org/abs/2501.09957

[8] 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:2402.07630 [cs.LG] https://arxiv.org/abs/2402.07630

[9] Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards Reasoning in Large Language Models: A Survey. In *Findings of the Association for Computational Linguistics: ACL 2023*, Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, Toronto, Canada, 1049–1065. https://doi.org/10.18653/v1/2023.findings-acl.67

[10] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of Hallucination in Natural Language Generation. *ACM Comput. Surv.* 55, 12, Article 248 (March 2023), 38 pages. https://doi.org/10.1145/3571730

[11] Sumin Jo, Junseong Choi, Jiho Kim, and Edward Choi. 2025. R2-KG: General-Purpose Dual-Agent Framework for Reliable Reasoning on Knowledge Graphs. arXiv:2502.12767 [cs.CL] https://arxiv.org/abs/2502.12767

[12] Jungo Kasai, Keisuke Sakaguchi, Yoichi Takahashi, Ronan Le Bras, Akari Asai, Xinyan Velocity Yu, Dragomir Radev, Noah A. Smith, Yejin Choi, and Kentaro Inui. 2023. REALTIME QA: what's the answer right now?. In *Proceedings of the 37th International Conference on Neural Information Processing Systems* (New Orleans, LA, USA) *(NIPS '23)*. Curran Associates Inc., Red Hook, NY, USA, Article 2130, 19 pages.

[13] Mufei Li, Siqi Miao, and Pan Li. 2025. Simple Is Effective: The Roles of Graphs and Large Language Models in Knowledge-Graph-Based Retrieval-Augmented Generation. arXiv:2410.20724 [cs.CL] https://arxiv.org/abs/2410.20724

[14] Songze Li, Zhiqiang Liu, Zhengke Gui, Huajun Chen, and Wen Zhang. 2025. Enrich-on-Graph: Query-Graph Alignment for Complex Reasoning with LLM Enriching. arXiv:2509.20810 [cs.CL] https://arxiv.org/abs/2509.20810

[15] Yading Li, Dandan Song, Changzhi Zhou, Yuhang Tian, Hao Wang, Ziyi Yang, and Shuhao Zhang. 2024. A Framework of Knowledge Graph-Enhanced Large Language Model Based on Question Decomposition and Atomic Retrieval. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (Eds.). Association for Computational Linguistics, Miami, Florida, USA, 11472–11485. https://doi.org/10.18653/v1/2024.findings-emnlp.670

[16] Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. 2024. Reasoning on Graphs: Faithful and Interpretable Large Language Model Reasoning. arXiv:2310.01061 [cs.CL] https://arxiv.org/abs/2310.01061

[17] Jie Ma, Zhitao Gao, Qi Chai, Wangchun Sun, Pinghui Wang, Hongbin Pei, Jing Tao, Lingyun Song, Jun Liu, Chen Zhang, and Lizhen Cui. 2024. Debate on Graph: a Flexible and Reliable Reasoning Framework for Large Language Models. arXiv:2409.03155 [cs.CL] https://arxiv.org/abs/2409.03155

[18] Costas Mavromatis, Soji Adeshina, Vassilis N. Ioannidis, Zhen Han, Qi Zhu, Ian Robinson, Bryan Thompson, Huzefa Rangwala, and George Karypis. 2025. BYOKG-RAG: Multi-Strategy Graph Retrieval for Knowledge Graph Question Answering. arXiv:2507.04127 [cs.CL] https://arxiv.org/abs/2507.04127

[19] OpenAI. 2024. Hello gpt-4o. https://openai.com/index/hello-gpt-4o Accessed: 2025-10-5.

[20] 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:2408.08921 [cs.AI] https://arxiv.org/abs/2408.08921

[21] Shuofei Qiao, Yixin Ou, Ningyu Zhang, Xiang Chen, Yunzhi Yao, Shumin Deng, Chuanqi Tan, Fei Huang, and Huajun Chen. 2023. Reasoning with Language Model Prompting: A Survey. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, Toronto, Canada, 5368–5393. https://doi.org/10.18653/v1/2023.acl-long.294

[22] Aditi Singh, Abul Ehtesham, Saket Kumar, and Tala Talaei Khoei. 2025. Agentic Retrieval-Augmented Generation: A Survey on Agentic RAG. arXiv:2501.09136 [cs.AI] https://arxiv.org/abs/2501.09136

[23] Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel M. Ni, Heung-Yeung Shum, and Jian Guo. 2024. Think-on-Graph: Deep and Responsible Reasoning of Large Language Model on Knowledge Graph. arXiv:2307.07697 [cs.CL] https://arxiv.org/abs/2307.07697

[24] Alon Talmor and Jonathan Berant. 2018. The Web as a Knowledge-Base for Answering Complex Questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, Marilyn Walker, Heng Ji, and Amanda Stent (Eds.). Association for Computational Linguistics, New Orleans, Louisiana, 641–651. https://doi.org/10.18653/v1/N18-1059

[25] Tu Vu, Mohit Iyyer, Xuezhi Wang, Noah Constant, Jerry Wei, Jason Wei, Chris Tar, Yun-Hsuan Sung, Denny Zhou, Quoc Le, and Thang Luong. 2024. FreshLLMs: Refreshing Large Language Models with Search Engine Augmentation. In *Findings of the Association for Computational Linguistics: ACL 2024*, Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.). Association for Computational Linguistics, Bangkok, Thailand, 13697–13720. https://doi.org/10.18653/v1/2024.findings-acl.813

[26] Yilin Xiao, Chuang Zhou, Qinggang Zhang, Bo Li, Qing Li, and Xiao Huang. 2025. Reliable Reasoning Path: Distilling Effective Guidance for LLM Reasoning with Knowledge Graphs. arXiv:2506.10508 [cs.CL] https://arxiv.org/abs/2506.10508

[27] Ruiyi Yang, Hao Xue, Imran Razzak, Hakim Hacid, and Flora D. Salim. 2025. Divide by Question, Conquer by Agent: SPLIT-RAG with Question-Driven Graph Partitioning. arXiv:2505.13994 [cs.AI] https://arxiv.org/abs/2505.13994

[28] Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. 2016. The Value of Semantic Parse Labeling for Knowledge Base Question Answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Katrin Erk and Noah A. Smith (Eds.). Association for Computational Linguistics, Berlin, Germany, 201–206. https://doi.org/10.18653/v1/P16-2033

[29] Qinggang Zhang, Shengyuan Chen, Yuanchen Bei, Zheng Yuan, Huachi Zhou, Zijin Hong, Hao Chen, Yilin Xiao, Chuang Zhou, Junnan Dong, Yi Chang, and Xiao Huang. 2025. A Survey of Graph Retrieval-Augmented Generation for Customized Large Language Models. arXiv:2501.13958 [cs.CL] https://arxiv.org/abs/2501.13958

[30] Yihua Zhu, Qianying Liu, Akiko Aizawa, and Hidetoshi Shimodaira. 2025. Beyond Chains: Bridging Large Language Models and Knowledge Bases in Complex Question Answering. arXiv:2505.14099 [cs.CL] https://arxiv.org/abs/2505.14099