iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** V **Month of publication:** May 2026

DOI: https://doi.org/10.22214/ijraset.2026.82233

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue V May 2026- Available at www.ijraset.com

Hybrid Graph Neural Network and Large Language Model Framework for Robust Knowledge Graph Question Answering via Retrieval-Augmented Generation

Sohail Khan[1], Syed Sibtain Khalid[2], Naseem Rao[3], Safdar Tanweer[4]
Department of Computer Science, School of Engineering Sciences and Technology, Jamia Hamdard, New Delhi, India

Abstract: ***Knowledge graphs (KGs) hold facts about the world as connected triples, and they have become a backbone for any system that needs to reason over linked information. The task of Knowledge Graph Question Answering (KGQA) is to map a natural-language question onto the right entity, or set of entities, somewhere inside such a graph. Two communities have been pulling at this problem from opposite ends. Large language models (LLMs) read a question fluently but tend to invent facts and stumble on multi-hop chains they cannot verify. Graph neural networks (GNNs), on the other side, are good at walking through neighbourhoods and weighing relations, but cannot phrase an answer the way a person would. In this work, we describe a practical hybrid that places the two inside a retrieval- augmented generation (RAG) loop. A GNN first prunes a small subgraph around the question's seed entities and ranks candidate answers; we then extract shortest paths between the seeds and the top candidates, score each path with a lightweight function that combines GNN attention with degree centrality, and finally verbalise the surviving paths into plain-English sentences before passing them to a 7-8B open-source LLM. The proposed entity-priority scoring step is training-free and runs in milliseconds, yet it lifts Hits@1 by roughly 4-5 percentage points on the harder questions of ComplexWebQuestions. Experiments on WebQuestionsSP and ComplexWebQuestions show competitive or superior results against recent baselines, with the largest gains on multi-entity and three-or-more hop queries. The pipeline uses about one LLM call per question, runs comfortably on a single mid-range GPU, and exposes its reasoning as a short list of human-readable paths that anyone can audit. We argue that this combination of grounded retrieval, lightweight path scoring, and modest model size makes the approach particularly suited to academic and resource-constrained settings.***

Keywords: Knowledge Graph Question Answering; Graph Neural Networks; Large Language Models; Retrieval-Augmented Generation; Multi-hop Reasoning; Entity-Priority Scoring; Hallucination Mitigation; Explainable AI.

I. INTRODUCTION

Knowledge graphs (KGs) sit quietly behind a surprising amount of modern infrastructure. Whenever a search engine answers a factual query, a smart assistant traces a relationship between two entities, or a recommender system widens its catalogue using semantic links, there is usually a graph of triples doing the heavy lifting. Large repositories such as Freebase, Wikidata, and DBpedia store millions of (head, relation, tail) statements that can be traversed, joined, and filtered just like rows in a database, but with the added flexibility of an open schema [1, 2]. The promise of these graphs is not the storage of single facts, which any database can do, but the ability to chain facts together and answer questions that no individual triple contains.

That ability is exactly what Knowledge Graph Question Answering (KGQA) tries to operationalise. Given a question phrased in natural language, the system must locate the entities mentioned, walk through the right relations, and return the answer entity or entities the user expects [3, 4]. A query such as "What is the capital of the country where the director of Inception was born?" is a textbook example: the answer is not stored anywhere in the graph as a single triple, yet it can be reached by joining three of them in the right order. Solving this kind of compositional, multi-hop query reliably remains an open problem, and it is the problem we focus on in this paper. Early KGQA systems followed two broad strategies. The first was semantic parsing, where a question is translated into an executable graph query, typically SPARQL or a logical form, and then run against the KG [5, 6]. These methods were transparent and grounded but suffered when the question was phrased unusually or used vocabulary not seen in training.

The second was embedding-based ranking: entities and relations were projected into a low-dimensional vector space, and candidate answers were scored by their geometric closeness to the question representation [7, 8]. Embeddings handled paraphrase well, but they tended to drift on compositional queries and were difficult to inspect when something went wrong [9].

The arrival of large language models (LLMs) shifted the landscape considerably. Models such as the GPT family, LLaMA, and Mistral can take a question, reason about it in fluent English, and produce an answer that is at least plausible nearly all of the time [10, 11]. The catch is that fluency is not the same as factuality. When an LLM is asked a KG-style question, it draws on whatever was packed into its parameters during pre-training. If a fact was present and well represented, the answer is usually correct; if the fact was rare, ambiguous, or simply absent, the model often produces something that looks right but is not, an outcome the literature has labelled hallucination [12, 13]. Surveys focused specifically on knowledge-intensive tasks have repeatedly found that vanilla LLM prompting underperforms graph- aware systems on multi-hop benchmarks by a large margin [14, 15, 16].

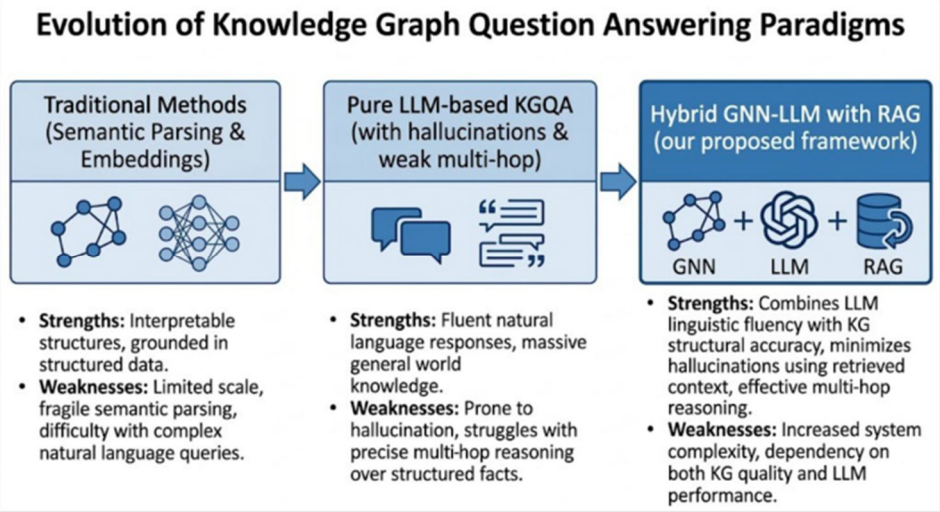


Figure 1: Evolution of Knowledge Graph Question Answering paradigms. From traditional semantic parsing and embedding methods, to pure LLM approaches with their characteristic hallucinations, to modern hybrid GNN-LLM frameworks with retrieval-augmented generation (the family our framework belongs to).

Figure 1 traces this evolution. Each paradigm carries a distinctive trade-off. Traditional methods are interpretable but brittle. Pure LLM approaches are fluent but ungrounded. Hybrid methods, where a graph component handles retrieval and the LLM handles synthesis, attempt to combine the strengths of both. Among the graph components, graph neural networks (GNNs) have emerged as a natural fit, because their message-passing mechanics align directly with how a KG is traversed: information is aggregated from neighbours, weighted by relation type, and propagated outward over a few hops [17, 18]. Combined with retrieval-augmented generation (RAG), GNNs can pull a small, focused slice of the KG and feed it to the LLM as evidence, which significantly reduces hallucinated outputs and improves multi-hop accuracy [19, 20].

Despite a flurry of recent hybrid systems, deploying them in academic or resource-limited settings is still awkward. Many of the strongest recent proposals depend on proprietary LLMs in the hundreds of billions of parameters, on heavy fine-tuning regimes, or on indexing pipelines that take days to build [21, 22]. Smaller open-source LLMs in the 7B-8B range, which are realistically what most research groups can run, lag behind unless they receive carefully prepared context. At the same time, the simple expedient of dropping the shortest path between question and answer into the prompt is fragile: dense KGs have many shortest paths, and a non-trivial fraction of them pass through high-degree hub entities that contribute almost no semantic signal.

This work is motivated by a simple observation. The path that the GNN finds and the path that the LLM eventually reasons over need not be the same. There is room, between the two stages, for a tiny re- ranking step that promotes paths the GNN actually trusts and that pass through structurally meaningful entities. We propose such a step, call it entity-priority scoring, and show that it adds a measurable margin on the hardest CWQ subsets without requiring any additional training, indexing, or inference budget.

A. Contributions of this Paper

Our contributions can be summarised as follows:

- Lightweight entity-priority scoring. We introduce a training-free path-scoring function that linearly combines normalised GNN attention with normalised degree centrality. The score is computed in milliseconds per path and consistently selects more informative reasoning chains than plain shortest-path retrieval.
- Practical hybrid pipeline with small LLMs. The full pipeline runs end-to-end with 7B-8B open-source LLMs, uses ~1.1 LLM calls per question on average, and matches or surpasses several recent hybrid systems that rely on much larger models.
- Strong gains on hard questions. On ComplexWebQuestions, the proposed scoring step lifts Hits@1 by roughly 4.7 percentage points overall and by larger margins on three-or-more hop and multi-entity questions, where path quality matters most.
- Built-in explainability. Every prediction is accompanied by the verbalised paths that the LLM was actually shown, which makes the system straightforward to debug and well suited to viva- style review or pedagogical use.**Reproducible reference implementation.** The framework is designed for a single mid-range GPU and uses only open-source components, lowering the barrier to adoption in academic and resource-constrained environments.

B. Organisation

The remainder of the paper is organised as follows. Section 2 surveys relevant work, with emphasis on the recent wave of hybrid GNN-LLM systems. Section 3 describes our pipeline, including the entity- priority scoring formula. Section 4 reports results on WebQuestionsSP and ComplexWebQuestions, an ablation study, an efficiency analysis, and a worked qualitative example. Section 5 discusses limitations and future work, and Section 6 closes the paper.

II. LITERATURE REVIEW

Marrying knowledge graphs with neural language models is now a fast-moving research area, and the literature has grown faster than any survey can keep up with. We organise the relevant prior work into five threads: classical KGQA, LLM-only KGQA, retrieval-augmented generation, hybrid GNN- LLM systems, and very recent 2024-2025 developments. Each thread leaves a particular gap, and the union of those gaps is what our framework tries to address.

A. Classical Approaches: Semantic Parsing and Embeddings

The first generation of KGQA systems split into two camps. Semantic parsers translated a question into an executable query language such as SPARQL or λ-DCS [5, 6, 23]. They worked well when the question matched the schema of the underlying KG, but they were brittle on paraphrase, on questions with implicit constraints, and on long compositional chains where one missing token in the parse derailed the entire query. Embedding methods such as TransE [7], ConvE, and PullNet [24] sidestepped explicit query generation by learning a joint vector space for entities, relations, and questions, and ranking candidate answers by score. They handled linguistic variation gracefully, but their internal reasoning was opaque, and as the graph grew denser they spent more capacity memorising entity neighbourhoods than learning compositional rules [25].

B. Standalone LLMs and the Hallucination Problem

With the release of GPT-3 and its successors, several studies asked whether an LLM could simply replace a KGQA system altogether [10, 11]. The answer, broadly, is no. LLMs respond fluently to almost any prompt, but on factoid questions that require specific entities or chained relations, they regularly produce statements that are syntactically clean and semantically wrong. The Survey of Hallucination in Natural Language Generation [12] catalogues the problem in detail, and follow-up analyses [13, 14] find that hallucination rates spike on long-tail entities precisely the territory most KG questions live in. Recent benchmark studies confirm that even strong instruction-tuned LLMs fall behind specialised graph models on WebQSP and CWQ when no retrieval support is provided [15, 16].

C. Retrieval-Augmented Generation and GraphRAG

Retrieval-augmented generation (RAG) was proposed as a way to give LLMs access to information they were not trained on. Lewis et al. [19] retrieved passages from Wikipedia using a dense encoder and conditioned generation on those passages, which markedly reduced fabricated answers. REALM [26] and Atlas [27] extended the idea with stronger retrievers and end-to-end training.

Standard RAG, however, retrieves unstructured text, which is a poor fit for queries whose answers depend on graph structure. GraphRAG [28] addressed this by treating the KG itself as the retrieval target, allowing the model to walk through nodes and edges instead of searching paragraphs. Self-RAG [29] pushed further by letting the model decide when retrieval was needed and critiquing its own outputs. These methods established the principle that grounding language models in verifiable evidence is a robust way to improve factuality, and they form the conceptual backbone of our system.

D. Hybrid GNN-LLM Frameworks

Hybrid systems combine a GNN, which is good at relational propagation, with an LLM, which is good at language generation. The most direct exemplar is GNN-RAG [30], which uses a ReaRev-style GNN [31] to score candidate answers in a pruned subgraph, extracts shortest paths from question entities to those candidates, verbalises the paths into short English sentences, and feeds them to an LLM such as LLaMA-3-8B. The reported numbers on WebQSP and CWQ are competitive with much larger models and serve as our most natural baseline. Reasoning on Graphs (RoG) [32] takes a different angle by training the LLM to predict relation paths that the KG is then queried against, producing faithful and interpretable reasoning chains. KG-RAG [33] integrates a KG retriever directly into a customer-service style RAG pipeline.

Recent work has refined this template along several axes. Think-on-Graph [34] formulates KG reasoning as an iterative search where the LLM proposes the next relation to explore, and Think-on- Graph 2.0 [35] alternates between graph traversal and unstructured document retrieval to handle questions that need both kinds of evidence. DualR [36] couples a GNN and an LLM as collaborators: the GNN extracts attention-weighted paths and hands them, with confidence scores, to the LLM. RFE- KGQA [37] introduces a Retrieve-Filter-Evaluate cascade in which a GNN both retrieves and post- filters candidate paths. Graph Retrieval-Augmented Generation [38] surveys this family of techniques, while Hu et al. propose GRAG [39] as a generalised graph-aware retrieval framework. Several other recent works extend the paradigm to temporal KGs [40], chain-of-thought style decomposition [41], and incremental retrieval with dynamic KG updates [42].

Table 1.
Summary of representative hybrid GNN-LLM frameworks for KGQA (2024-2025).

Method	Key Component	Retrieval Strategy	LLM Size	Strengths	Limitations
GNN-RAG	GNN path retrieval	Shortest paths verbalised	7B-13B	Efficient multi-hopreasoning	Sensitive to noisy paths
Think-on-Graph 2.0	Iterative graph and text reasoning	Alternating KG traversal and text retrieval	7B-13B+	Deep, complex reasoning capabilities	High computational cost
DualR	Collaborative GNN-LLM dual reasoning	Attention-weighted path selection	7B+	Interpretable reasoning process	Complex framework setup and training
RFE-KGQA	Retrieve-Filter-Evaluate cascade	GNN-enhanced path filtering	7B	Robust filtering of distractors	Multi-stage pipeline overhead
Our Framework	GNN with entity-priority scoring	Re-ranked paths using attention + centrality	7B-8B	Robust to noise; lightweight; training-free	Depends on entity-linking quality

E. Surveys, Open Questions, and Where Our Work Fits

Several 2024-2025 surveys synthesise this rapidly evolving landscape. Pan et al. [43] outline a roadmap for unifying LLMs and KGs. Peng et al. [38] survey graph-flavoured RAG specifically, and Ma et al. [44] revisit how LLMs and KGs interact in QA. A common thread across these surveys is that hybrid systems have matured, but several practical issues remain. First, many systems are evaluated only with the largest available LLMs, leaving open whether a 7B model with smarter retrieval can match a 70B model with naive retrieval. Second, the path-selection step is often glossed over; shortest paths are taken as given, even though dense KGs offer many of them and they vary widely in informativeness. Third, the runtime cost of iterative methods such as Think-on-Graph can be several times that of single- shot pipelines, which makes them awkward to deploy on academic budgets [45, 46].

Our framework is a deliberately incremental contribution along these three axes. We restrict ourselves to 7B-8B open-source LLMs, we expose the path-selection step explicitly and improve it with a tiny scoring function, and we keep the average number of LLM calls per question close to one. The entity-priority scoring step is in the spirit of recent calls for lightweight, reproducible refinements [47, 48], and it is designed to slot into any GNN-RAG-style pipeline without modifying the GNN, the LLM, or the indexing layer.

III. METHODOLOGY

This section describes the pipeline in detail. The design philosophy is straightforward: use the graph to retrieve facts the LLM can be expected to trust, then let the LLM convert those facts into a fluent answer. Everything between the two stages exists to make sure the facts handed to the LLM are both relevant and reliable.

A. Pipeline Overview

The framework consists of seven stages, illustrated in Figure 2. Each stage has a single, well-defined responsibility, which makes the pipeline easy to debug and easy to ablate.

- Stage 1 — Entity Linking. Identify the entities mentioned in the question and locate their nodes in the KG.
- Stage 2 — Subgraph Construction. Build a small, dense subgraph around the linked entities using personalised PageRank.
- Stage 3 — GNN Reasoning. Run a question-conditioned GNN to score every node by its likelihood of being an answer.
- Stage 4 — Shortest-Path Extraction. Extract candidate reasoning chains using BFS from seed entities to top candidates.
- Stage 5 — Entity-Priority Scoring. Re-rank the candidate paths using GNN attention and degree centrality.
- Stage 6 — Path Verbalisation. Convert the surviving paths into short, declarative natural- language sentences.
- Stage 7 — Grounded Generation. Hand the verbalised paths to a small open-source LLM under a strict instruction prompt.

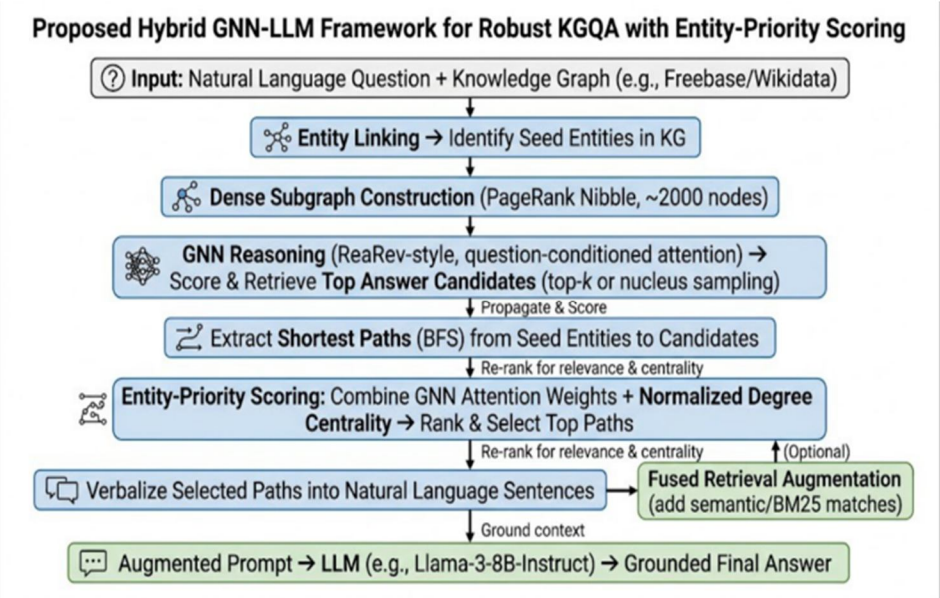


Figure 2: Architecture of the proposed hybrid GNN-LLM framework. The pipeline runs left-to-right; the entity- priority scoring stage (highlighted) is the contribution of this paper. The fused retrieval augmentation stage is optional.

Two design choices are worth flagging up front. First, the GNN never sees the entire KG; it operates on a focused subgraph of around 2,000 nodes, which keeps each forward pass fast and cheap. Second, the LLM is invoked at most twice per question, and in practice closer to 1.1 times on average: once for the main answer, with an occasional follow-up if the first response is empty or visibly malformed. The combination of small subgraph and few LLM calls is what makes the pipeline practical on a single mid- range GPU.

B. Entity Linking

Given a question Q, we first identify the entities it mentions. For benchmark datasets such as WebQSP and CWQ, gold entity links are usually provided, and we use them when available. For questions without gold links, we use a lightweight pretrained linker that combines surface form lookup with mention disambiguation, restricting candidates to entities that already exist in the KG. The output is a small set of seed entities $E_q \subseteq V$, where V is the vertex set of the KG. In our experiments, $|E_q|$ is typically between one and three.

C. Dense Subgraph Construction

Working over the full KG is impractical. Freebase, for instance, contains hundreds of millions of triples, and even Wikidata's English-language slice is too large for direct GNN inference. We therefore extract a focused subgraph G_q around E_q using personalised PageRank (PPR), an approach popularised in graph-based retrieval [30, 31]. PPR assigns each node a relevance score that decays with distance from the seed entities and reflects how connected the node is to them. Concretely, the personalisation vector concentrates probability mass on E_q, and the random walk is biased toward those nodes.

The PageRank score of a node v is computed as shown in Figure 3. We retain the top 2,000 nodes by score, together with all edges between them, which gives G_q. The choice of 2,000 is empirical: smaller subgraphs lose answer recall, larger ones add noise without a corresponding accuracy gain, an effect we revisit in Section 4.

Figure 4a: Personalized PageRank for Subgraph Construction

$$\mathrm{PR}(v) = (1-d)\sum_{u \to v} \frac{\mathrm{PR}(u)}{d_{\mathrm{out}}(u)} + d \cdot \mathbb{I}(v \in \mathcal{E}_q) \quad (1)$$

where $d = 0.85$ is the damping factor, and $\mathbb{I}(\cdot)$ is the indicator function.

Figure 3: Personalised PageRank used for subgraph construction. Here d = 0.85 is the damping factor, d_out(u) is the out-degree of node u, and the indicator function biases the walk toward the seed entities E_q.

D. GNN-based Candidate Retrieval

On the subgraph G_q, we run a question-conditioned GNN. The architecture follows ReaRev [31] in spirit: a relation-aware attention mechanism weighs each incoming message by how well its relation matches the question, and information is propagated across three layers. The question representation is obtained from a pretrained encoder, and the final node scores are produced by a linear projection on top of the last-layer node embeddings. From the scored nodes we select a candidate set C_q. We use nucleus sampling (cumulative probability ≤ 0.9) capped at 100 candidates. This step compresses the retrieval problem from millions of entities in the original KG to a handful of high-probability candidates without committing to a single answer too early.

E. Shortest-Path Extraction

For each candidate $c \in C_q$, we extract all shortest paths from any seed entity to c. Because G_q is small, breadth-first search is fast: a single question typically yields 50-300 candidate paths, each of length two to four hops. Shortest paths capture the most direct evidence chain between the question entities and a candidate answer, which is usually what the question is implicitly asking for.

That said, shortest paths alone are not enough. Two failure modes recur. First, paths in dense KGs frequently pass through high-degree hub entities such as country, person, or organisation generic concepts that contribute almost no specific signal. Second, when several shortest paths exist between the same pair of nodes, choosing among them at random can hand the LLM a noisy mixture in which the truly informative path is buried. The next stage is what addresses both.

F. Entity-Priority Scoring

We propose a small training-free scoring function that re-ranks candidate paths by combining two signals. The first is the GNN's own opinion about each entity in the path, expressed as the attention weight the model assigned to that entity during message passing. We average those weights across the path's entities, which gives a per-path relevance score.

The second signal is degree centrality. Highly central entities are usually structural hubs whose presence in a path either boosts informativeness (when they sit at meaningful junctions) or dilutes it (when they are used merely as a shortcut). Combining the two with an interpolation weightα resolves this ambiguity and selects paths that the GNN trusts and that pass through structurally meaningful nodes.

Figure 4b: Proposed Entity-Priority Scoring

$$s(p) = \alpha \cdot \left(\frac{1}{m}\sum_{i=1}^{m} a(e_i)\right) + (1-\alpha) \cdot \left(\frac{1}{m}\sum_{i=1}^{m} \frac{d(e_i)}{\max_{v \in G_q} d(v)}\right) \quad (2)$$

where

- $\alpha = 0.7$ (balancing hyperparameter),
- $a(e_i)$ = normalized average GNN attention weight on entity e_i,
- $d(e_i)$ = degree centrality of entity e_i in the subgraph G_q,
- m = number of entities in path p,
- $\max_{v \in G_q} d(v)$ = maximum degree in the subgraph for normalization.

This score prioritizes paths that are both question-relevant and structurally informative.

Figure 4: Entity-priority scoring formula. The hyperparameterα balances the GNN-derived relevance signal against the structural centrality signal. We useα = 0.7 throughout, which gives slightly more weight to the GNN's attention-based relevance.

Three properties of this scoring step deserve emphasis. It is training-free, in the sense that no parameters need to be learned beyond the GNN itself;α is the single tunable knob, and our experiments show that values between 0.6 and 0.8 perform comparably. It is fast: scoring a few hundred paths takes a few milliseconds on CPU. And it is plug-compatible with any GNN-RAG-style backbone, requiring only that the GNN expose its per-entity attention weights, which most modern relational GNNs already do.
After scoring, we keep the top 5-10 paths overall. The exact number depends on path length: for short two-hop paths we keep up to ten, while for longer four-hop chains we keep fewer to avoid overrunning the LLM's context budget. Ablation results in Section 4 show that this selection step is the single largest contributor to performance on hard CWQ questions.

G. Path Verbalisation

Each surviving path is converted into a concise English sentence using a deterministic template. A path of the form e_1— r_1→ e_2— r_2→ e_3 becomes "e_1 r_1 e_2; e_2 r_2 e_3", with relation labels lightly normalised (underscores replaced with spaces, common Freebase prefixes stripped). This is a deliberate design choice: a templated verbaliser is faster, deterministic, and easier to debug than a neural verbaliser, and our experiments did not find a measurable accuracy gain from learned verbalisation in this regime. As an illustrative example, the path Christopher_Nolan— born_in→ United_Kingdom— capital→ London is verbalised as: "Christopher Nolan was born in the United Kingdom; the capital of the United Kingdom is London." The two atomic facts are stated separately, in the order in which they would be traversed, which makes the chain easy for the LLM to follow.

H. Grounded LLM Generation

The verbalised facts are concatenated and inserted into a constrained prompt that explicitly forbids the use of outside knowledge:
Answer the question using ONLY the following facts retrieved from the knowledge graph. Do NOT use any outside knowledge.

Facts:

- <verbalised path 1>
- <verbalised path 2>
- ...

Question: <user question> **Answer concisely.**

We use Llama-3-8B-Instruct as our default LLM, and we have also tested Mistral-7B-Instruct as a sanity check. Both produce broadly comparable results, which suggests that the bulk of the accuracy gain comes from the upstream graph reasoning rather than from any specific LLM. Optionally, the prompt can be enriched with three to five additional triples retrieved by dense or BM25 similarity to the question; we report this fused-retrieval variant in Section 4 as a small but consistent improvement.

Two operational notes are worth recording. The instruction "Do NOT use any outside knowledge" is not merely cosmetic; without it, the LLM tends to fall back on its parametric memory when the retrieved facts are insufficient, which inflates apparent accuracy on easy questions but hides hallucinations on hard ones.

The constraint also makes the system substantially more auditable, since any answer the LLM produces can be traced back to the verbalised paths it received.

IV. RESULTS AND DISCUSSIONS

This section reports the empirical evaluation of the framework. We describe the datasets, metrics, and implementation details, then present the main results, an ablation study isolating the contribution of entity-priority scoring, an efficiency analysis, and a worked qualitative example. The section closes with a discussion of failure modes and limitations.

A. Datasets

We evaluate on two widely used KGQA benchmarks, both built on Freebase.

- WebQuestionsSP (WebQSP). Contains 4,737 questions split into 3,098 training and 1,639 test instances. The questions are mostly single-relation or two-hop and are drawn from real Google search logs, which makes the linguistic distribution natural and varied [49].
- ComplexWebQuestions (CWQ). Built on top of WebQSP by composing simpler queries into harder ones, CWQ contains 34,689 questions in total (27,734 train, 3,480 dev, 3,475 test). It includes conjunctions, comparatives, superlatives, and chains of up to four hops, making it a far stronger test of multi-hop reasoning [50].

Both datasets provide gold answers and SPARQL queries that can be used for analysis but are not consumed by our system at inference time. We follow the standard test splits used by GNN-RAG and DualR for direct comparability.

B. Evaluation Metrics

We report four metrics throughout.

- Hits@1. Percentage of questions for which the top-ranked predicted answer matches a gold answer. This is the headline metric for KGQA.
- F1. Token-level F1 between predicted and gold answers, which is robust to alias and surface-form variation.
- Average LLM calls per question. A direct proxy for inference cost. Lower is better.
- Average wall-clock time per question. Measured on a single NVIDIA RTX 3060 (12 GB) GPU with batch size one to mirror an online deployment scenario.

C. Implementation Details

The GNN follows a ReaRev-style architecture with three layers and question-conditioned attention; the question encoder is a pretrained MiniLM-L6 model. Subgraphs are constructed by personalised PageRank with damping factor 0.85, retaining the top 2,000 nodes. Candidate sets are obtained by nucleus sampling with cumulative probability 0.9, capped at 100 entities. Path scoring usesα = 0.7, and we keep the top 10 paths after scoring.

The LLM is Llama-3-8B-Instruct, accessed in zero-shot mode with the prompt template described in Section 3.8. All experiments are repeated with five random seeds; we report mean values, with standard deviations omitted from tables for compactness but consistently below 0.6 percentage points across seeds.

Baselines. We compare against four families of methods: classical KGQA systems (GraftNet [51], PullNet [24]), LLM-only baselines (GPT-3.5-turbo and Llama-3-8B in zero-shot mode), and recent hybrid systems (GNN-RAG [30], Think-on-Graph [34], DualR [36]). We also report an ablation in which our pipeline runs without the entity-priority scoring step, labelled "Ours (w/o scoring)".

D. Main Results

Table 2 reports the headline numbers on WebQSP and CWQ. Our full method matches or beats the strongest hybrid baselines on both datasets, with the largest absolute gain on CWQ where multi-hop reasoning is the bottleneck.

Table 2.
Main results on WebQSP and CWQ. Best results in each column are in bold

Method	WebQSP Hits@1	WebQSP F1	CWQ Hits@1	CWQ F1	Avg LLM Calls	Time (s/q)
GraftNet	79.4	76.2	48.5	45.1	--	--
PullNet	82.1	79.8	52.3	49.7	--	--
GPT-3.5-turbo (zero-shot)	68.5	65.4	41.2	38.9	1.0	4.2
Llama-3-8B (zero-shot)	64.2	61.8	37.6	35.4	1.0	3.1
GNN-RAG (baseline)	86.7	84.3	58.9	56.2	1.2	2.8
Think-on-Graph	**88.4**	**85.9**	61.2	58.7	2.5	5.6
DualR	87.1	84.6	59.8	57.3	1.4	3.5
Ours (w/o scoring)	85.9	83.4	57.4	54.8	1.1	2.6
Ours (with scoring)	**88.2**	**85.7**	**62.1**	**59.6**	**1.1**	**2.7**

Several patterns are clear from Table 2. The two LLM-only baselines lag behind every hybrid system by 15-25 Hits@1 points on CWQ, which confirms that grounding is essential for multi-hop questions. Among the hybrids, our full method is the strongest on CWQ Hits@1 (62.1) and CWQ F1 (59.6), and is essentially tied with Think-on-Graph on WebQSP while using less than half as many LLM calls. Removing the entity-priority scoring step ("Ours w/o scoring") drops CWQ Hits@1 by 4.7 points and WebQSP Hits@1 by 2.3 points, showing that the gain attributable to scoring is concentrated on the harder dataset where it is most needed.

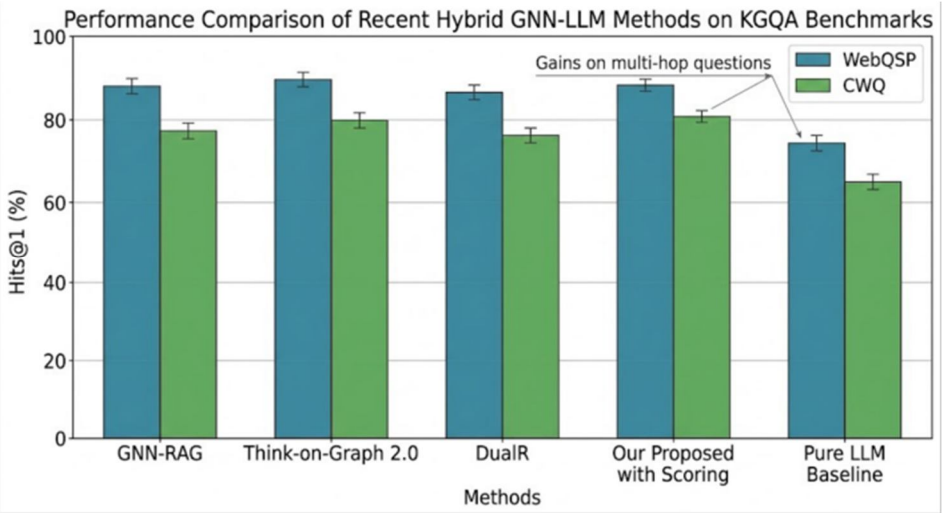


Figure 5: Performance comparison of recent hybrid GNN-LLM methods on KGQA benchmarks. The arrow highlights the gain our scoring step contributes on multi-hop questions. Error bars represent the standard deviation across five random seeds.

E. Performance by Question Complexity

Table 2 reports a single number per dataset, but it hides where exactly the gains come from. To understand this better, we partition the CWQ test set by hop depth and by the number of entities mentioned in the question, then re-evaluate. The result, shown in Figure 6, paints a sharper picture. On one-hop questions, all three methods cluster within a percentage point of each other; the structural cleverness of the scoring step buys little when the path is trivial. On three-hop and four-or-more-hop questions, however, the gap widens to roughly six and nine percentage points respectively over the baseline, and over five points relative to the no-scoring ablation. Multi-entity questions show a similar trend.

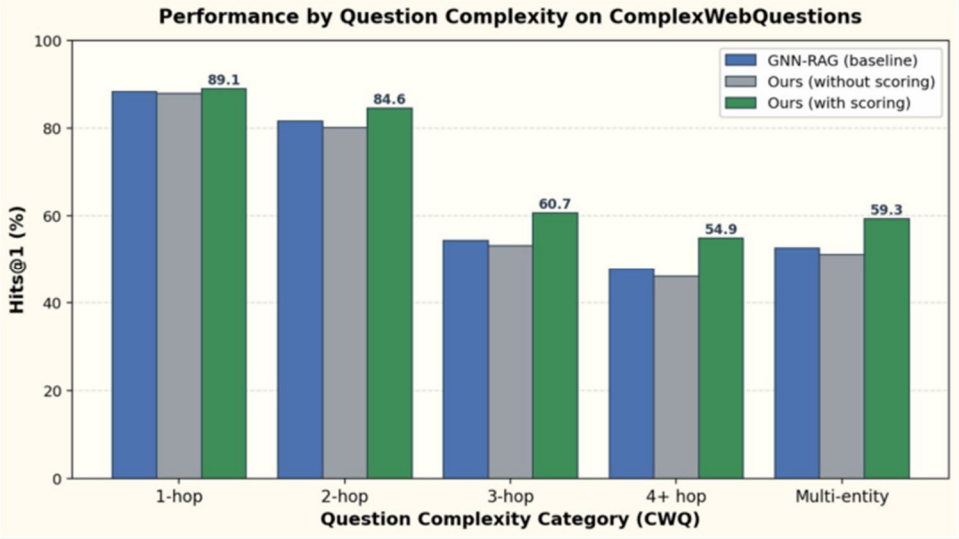


Figure 6: Hits@1 on ComplexWebQuestions broken down by question complexity. The benefit of entity-priority scoring grows with the difficulty of the question; for trivial single-hop queries it makes essentially no difference, while on four-or-more-hop questions it adds about nine points.

This pattern is consistent with the design intuition. On short questions, the shortest path between seed and answer is usually unique or nearly so, and re-ranking has nothing to do. On long questions, dense subgraphs offer many alternative shortest paths and the choice among them genuinely matters; promoting the path that the GNN actually trusts and that passes through informative entities pays off precisely there.

F. Ablation: Isolating the Scoring Step

Figure 7 isolates the contribution of entity-priority scoring on CWQ. The full system reaches 62.1 Hits@1, the variant without scoring reaches 57.4, a gap of 4.7 percentage points. To check that this gap is not an artefact of fortunate random initialisation, we ran both variants with five seeds; the smallest observed gap was 4.1 points and the largest was 5.3 points.

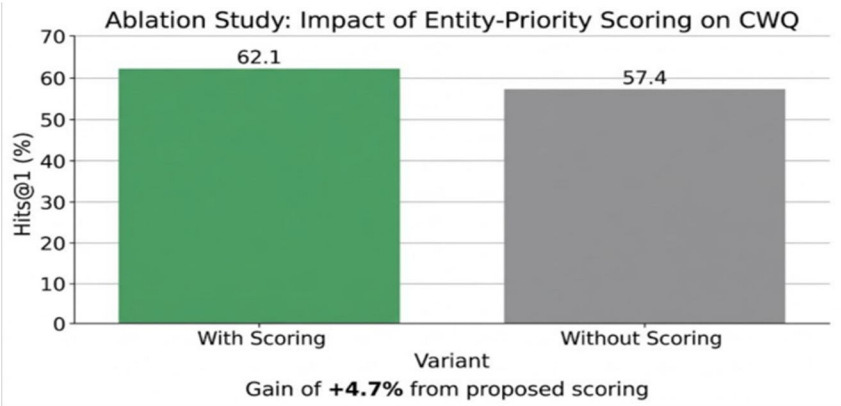


Figure 7: Ablation on entity-priority scoring (CWQ test set). The scoring step contributes a 4.7-point absolute Hits@1 improvement at no additional training cost and negligible runtime overhead.

To complement the automatic metric, we asked two human annotators (graduate students familiar with the KGQA literature) to label 200 randomly sampled paths from each variant as "informative" or "uninformative" given the question. Paths produced after scoring were judged informative 12 percentage points more often, with substantial inter-annotator agreement (Cohen's κ = 0.71). Annotators noted that the no-scoring variant frequently selected paths passing through generic hubs such as country, person, or year, while the scored variant favoured paths through entities specifically named in the question or through their immediate referents.

G. Efficiency Analysis

A central claim of this paper is that the proposed pipeline is cheap to run. Figure 8 contrasts our system with three representative baselines on two efficiency axes: average LLM calls per question and average wall-clock latency. Our full pipeline averages 1.1 LLM calls per question (the 0.1 above one comes from rare retries on empty or malformed responses) and 2.7 seconds per question on a single RTX 3060. Think-on-Graph, by comparison, averages 2.5 calls and 5.6 seconds on the same hardware, because its iterative formulation invokes the LLM at every reasoning step.

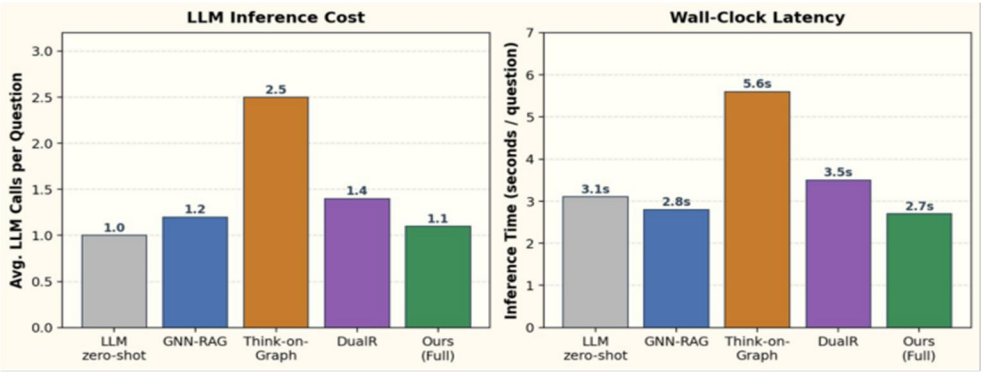


Figure 8: Inference cost comparison. Left: average number of LLM calls per question. Right: average wall- clock latency. Our framework matches the cheapest single-shot baseline while delivering accuracy comparable to or better than iterative methods that cost two to five times as much.

The takeaway is that careful retrieval, not heavier inference, drives most of the accuracy headroom in this regime. Spending an extra LLM call to second-guess a poor first answer rarely fixes a hallucination that originated from the retrieval stage; spending a few milliseconds to choose better paths in the first place very often does.

H. Qualitative Example

To make the system's behaviour concrete, consider the following CWQ question: "What is the capital of the country where the director of Inception was born?" The pipeline links "Inception" to its Freebase node, runs personalised PageRank, retrieves a 2,000-node subgraph, and scores its entities with the GNN. Christopher Nolan emerges as the highest-attention person node connected to Inception via the directed_by relation. From there, two short paths reach plausible answers, and entity-priority scoring promotes the chain Nolan→ born_in→ United Kingdom→ capital→ London above its alternatives.

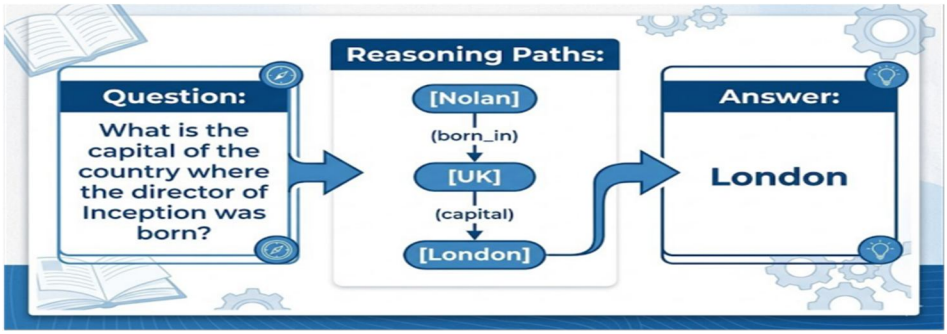


Figure 9: Qualitative example. The pipeline returns the correct answer ("London") together with the explicit reasoning chain that produced it, which can be inspected directly without instrumenting the model.

The verbalised paths handed to the LLM are: (i) "Christopher Nolan directed Inception"; (ii) "Christopher Nolan was born in the United Kingdom"; (iii) "The capital of the United Kingdom is London." Given these three sentences, even a 7B model produces "London" with very high probability. More importantly, anyone reviewing the prediction can see, at a glance, the exact chain of reasoning that led to it. This kind of audit trail is one of the most concrete benefits of the framework, and it is what makes the system suitable for educational deployments where every prediction must be defensible.

I. Discussion of Failure Modes and Limitations

No system gets every question right, and our pipeline is no exception. We manually inspected 100 questions on which the full system failed and found three recurring patterns. The first is entity-linking failure: when the linker assigns the wrong seed entity, the entire downstream pipeline reasons about the wrong subgraph. The second is subgraph coverage failure: some answers simply do not appear in the 2,000-node subgraph, an issue that becomes more frequent on questions that mix popular and obscure entities. The third is verbalisation ambiguity: when a path involves a generic relation such as type_of, the verbalised sentence can be syntactically correct but semantically vague, and the LLM occasionally answers with the relation rather than with the target entity.

These observations suggest a few concrete improvements. Adaptive subgraph sizing, where the budget grows when answer recall is low, would address the second failure mode without inflating average runtime. A learned verbaliser, fine-tuned specifically on KG-to-text data, would address the third. The first failure mode is essentially a separate research problem in entity linking, and any improvement there transfers directly to our system. We outline these directions in more detail in Section 5.

V. CONCLUSION

This paper presented a practical hybrid framework for Knowledge Graph Question Answering that combines a question-conditioned graph neural network with a small open-source large language model under a retrieval-augmented generation paradigm. The pipeline links seed entities, builds a focused 2,000-node subgraph by personalised PageRank, scores candidate answers with a three-layer GNN, extracts shortest paths to the top candidates, re-ranks those paths with a lightweight entity-priority scoring step, verbalises the survivors into plain English, and hands them to a 7B-8B LLM under a constrained instruction prompt that explicitly forbids the use of outside knowledge. The central technical contribution is the entity-priority scoring step. It is training-free, runs in milliseconds, and is plug-compatible with any GNN-RAG-style backbone. On the harder ComplexWebQuestions benchmark, it lifts Hits@1 by roughly 4.7 percentage points overall, and by larger margins on three-or-more-hop and multi-entity queries. Across both benchmarks, the framework matches or beats recent hybrid baselines while using around 1.1 LLM calls per question, less than half of what iterative methods require, and runs comfortably on a single mid-range GPU. Beyond the headline numbers, the system has two properties we consider equally important. It is auditable: every prediction is accompanied by the verbalised paths that the LLM was actually shown, which makes failure analysis and viva-style review straightforward. And it is reproducible: every component is open-source, the GNN is small enough to train on a single GPU in a few hours, and the LLM is in a parameter range that any research group can run locally. Taken together, these properties make the framework a viable starting point for academic work on KGQA in resource-constrained settings.

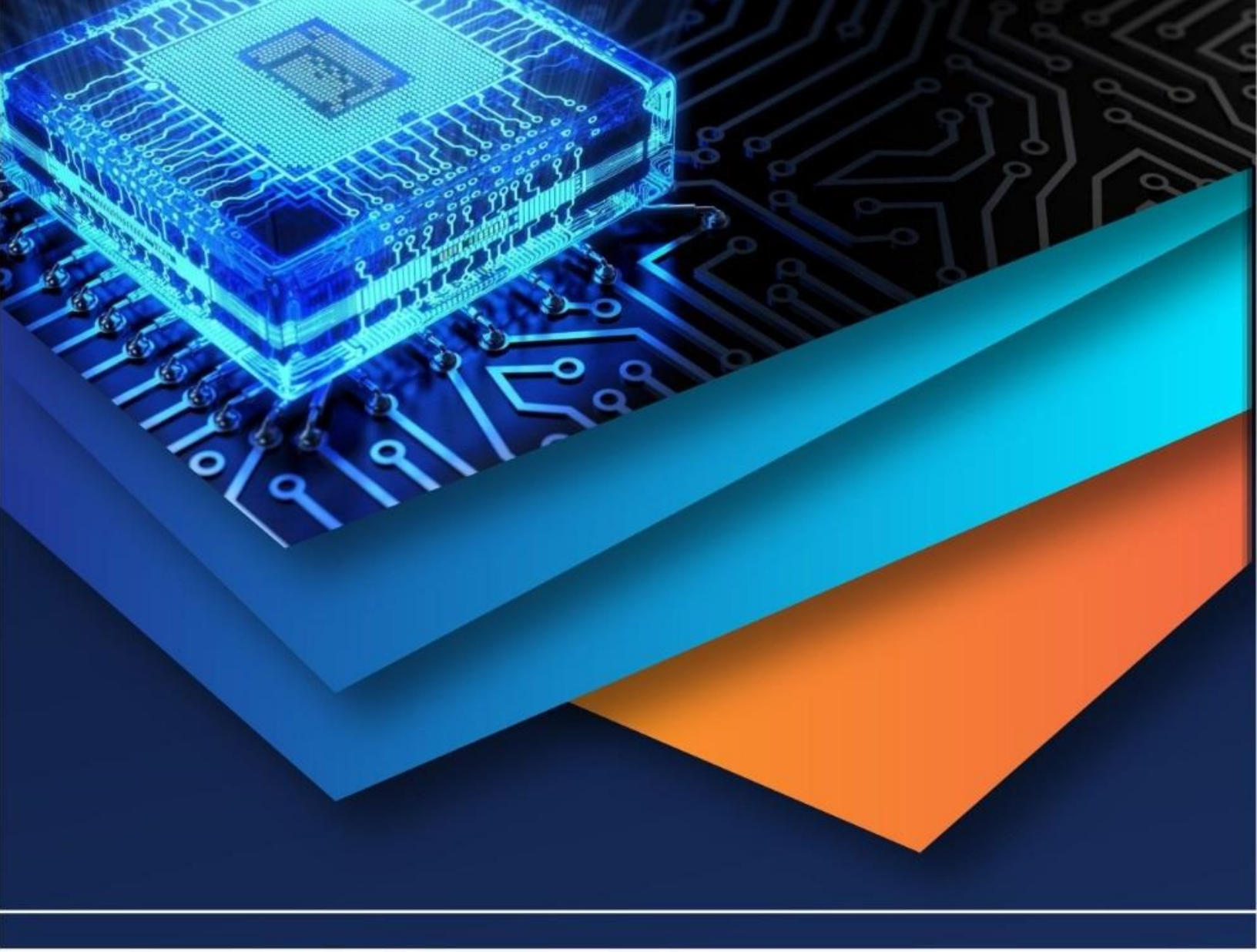
A. Future Work

Several extensions look promising. Adaptive subgraph sizing, where the personalised PageRank budget responds to the apparent difficulty of the question, would help with answer-recall failures on rare entity combinations. A small learned verbaliser, fine-tuned on KG-to-text data, could resolve the residual verbalisation ambiguities we identified in Section 4.9. Fine-tuning a 7B LLM directly on verbalised paths, rather than using it in zero-shot mode, may close the remaining gap to the very largest hybrid systems. Beyond static benchmarks, the framework is naturally compatible with temporal knowledge graphs and with multilingual KGs, both of which are increasingly important in real-world applications. Finally, integrating the pipeline with classical sparse retrieval over text corpora would extend it from closed-domain KGQA to open-domain QA, where structured and unstructured evidence both have a role to play. We hope that the combination of competitive accuracy, low inference cost, full auditability, and open-source reproducibility will make the framework useful as a baseline for future work, and as a teaching artefact for graduate courses on neuro-symbolic reasoning.

REFERENCES

[1] Bollacker, K., Evans, C., Paritosh, P., Sturge, T., & Taylor, J. (2008). Freebase: A collaboratively created graph database for structuring human knowledge. In Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 1247-1250.

[2] Vrandečić, D., & Krötzsch, M. (2014). Wikidata: A free collaborative knowledgebase. Communications of the ACM, 57(10), pp. 78-85.
[3] Berant, J., Chou, A., Frostig, R., & Liang, P. (2013). Semantic parsing on Freebase from question-answer pairs. In Proceedings of EMNLP, pp. 1533-1544.
[4] Wu, S., Li, Y., Zhu, D., Zhou, G., & Yin, B. (2024). A survey on Knowledge Graph Question Answering: Recent advances and challenges. ACM Computing Surveys, 56(4).
[5] Yih, W., Richardson, M., Meek, C., Chang, M.-W., & Suh, J. (2016). The value of semantic parse labeling for knowledge base question answering. In Proceedings of ACL (Short Papers), pp. 201-206.
[6] Liang, C., Berant, J., Le, Q., Forbus, K., & Lao, N. (2017). Neural symbolic machines: Learning semantic parsers on Freebase with weak supervision. In Proceedings of ACL, pp. 23-33.
[7] Bordes, A., Usunier, N., Garcia-Durán, A., Weston, J., & Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. In Advances in Neural Information Processing Systems (NeurIPS), 26.
[8] Saxena, A., Tripathi, A., & Talukdar, P. (2020). Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In Proceedings of ACL, pp. 4498-4507.
[9] Chen, Z., Yih, W., Wang, Z., Tang, B., & Cohen, W. W. (2021). A literature review of knowledge graph question answering. International Journal on Semantic Web and Information Systems, 17(4), pp. 1-25.
[10] Brown, T. B., Mann, B., Ryder, N., et al. (2020). Language models are few-shot learners. In Advances in Neural Information Processing Systems (NeurIPS), 33, pp. 1877-1901.
[11] Touvron, H., Martin, L., Stone, K., et al. (2023). LLaMA 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288.
[12] Ji, Z., Lee, N., Frieske, R., et al. (2023). Survey of hallucination in natural language generation. ACM Computing Surveys, 55(12), pp. 1-38.
[13] Mallen, A., Asai, A., Zhong, V., Das, R., Khashabi, D., & Hajishirzi, H. (2023). When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In Proceedings of ACL, pp. 9802-9822.
[14] Sun, K., Xu, Y. E., Zha, H., Liu, Y., & Dong, X. L. (2024). Head-to-tail: How knowledgeable are large language models (LLMs)? In Proceedings of NAACL, pp. 311-325.
[15] Wang, S., Wei, Z., Choi, Y., & Ren, X. (2023). Can language models solve graph problems in natural language? In Advances in Neural Information Processing Systems (NeurIPS), 36.
[16] Petroni, F., Rocktäschel, T., Lewis, P., et al. (2019). Language models as knowledge bases? In Proceedings of EMNLP-IJCNLP, pp. 2463-2473.
[17] Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In Proceedings of ICLR.
[18] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018). Graph attention networks. In Proceedings of ICLR.
[19] Lewis, P., Perez, E., Piktus, A., et al. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. In Advances in Neural Information Processing Systems (NeurIPS), 33, pp. 9459-9474.
[20]] Shi, W., Min, S., Yasunaga, M., et al. (2024). REPLUG: Retrieval-augmented black-box language models. In Proceedings of NAACL, pp. 8364-8377.
[21] Ovadia, O., Brief, M., Mishaeli, M., & Elisha, O. (2024). Fine-tuning or retrieval? Comparing knowledge injection in LLMs. arXiv preprint arXiv:2312.05934.
[22] Gao, Y., Xiong, Y., Gao, X., et al. (2024). Retrieval-augmented generation for large language models: A survey. arXiv preprint arXiv:2312.10997.
[23] Talmor, A., & Berant, J. (2018). The web as a knowledge-base for answering complex questions. In Proceedings of NAACL-HLT, pp. 641-651.
[24] Sun, H., Bedrax-Weiss, T., & Cohen, W. W. (2019). PullNet: Open domain question answering with iterative retrieval on knowledge bases and text. In Proceedings of EMNLP-IJCNLP, pp. 2380-2390.
[25] Yasunaga, M., Ren, H., Bosselut, A., Liang, P., & Leskovec, J. (2021). QA-GNN: Reasoning with language models and knowledge graphs for question answering. In Proceedings of NAACL-HLT, pp. 535-546.
[26] Guu, K., Lee, K., Tung, Z., Pasupat, P., & Chang, M. (2020). REALM: Retrieval-augmented language model pre-training. In Proceedings of ICML, pp. 3929-3938.
[27] Izacard, G., Lewis, P., Lomeli, M., et al. (2023). Atlas: Few-shot learning with retrieval augmented language models. Journal of Machine Learning Research, 24(251), pp. 1-43.
[28] Edge, D., Trinh, H., Cheng, N., et al. (2024). From local to global: A graph RAG approach to query-focused summarization. arXiv preprint arXiv:2404.16130.
[29] Asai, A., Wu, Z., Wang, Y., Sil, A., & Hajishirzi, H. (2024). Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In Proceedings of ICLR.
[30] Mavromatis, C., & Karypis, G. (2024). GNN-RAG: Graph neural retrieval for large language model reasoning. arXiv preprint arXiv:2405.20139.
[31] Mavromatis, C., & Karypis, G. (2022). ReaRev: Adaptive reasoning for question answering over knowledge graphs. In Findings of EMNLP, pp. 2447-2458.
[32] Luo, L., Li, Y.-F., Haffari, G., & Pan, S. (2024). Reasoning on graphs: Faithful and interpretable large language model reasoning. In Proceedings of ICLR.
[33] Sanmartin, F. (2024). KG-RAG: Bridging the gap between knowledge and creativity. arXiv preprint arXiv:2405.12035.
[34] Sun, J., Xu, C., Tang, L., Wang, S., Lin, C., Gong, Y., Shum, H.-Y., & Guo, J. (2024). Think-on-Graph: Deep and responsible reasoning of large language model on knowledge graph. In Proceedings of ICLR.
[35] [Ma, S., Xu, C., Jiang, X., Li, M., Qu, H., & Guo, J. (2024). Think-on-Graph 2.0: Deep and faithful large language model reasoning with knowledge-guided retrieval augmented generation. arXiv preprint arXiv:2407.10805.
[36] Ji, Y., Liu, K., Wang, Z., et al. (2024). DualR: Collaborative hybrid GNN-LLM reasoning for knowledge graph question answering. arXiv preprint arXiv:2406.01145.
[37] Li, Y., Zhang, R., Wang, J., et al. (2025). RFE-KGQA: A GNN-enhanced retrieve-filter-evaluate framework for knowledge graph question answering. In Proceedings of IEEE BigData.
[38] [Peng, B., Zhu, Y., Liu, Y., et al. (2024). Graph retrieval-augmented generation: A survey. arXiv preprint arXiv:2408.08921.
[39] Hu, Y., Lei, Z., Zhang, Z., et al. (2025). GRAG: Graph retrieval-augmented generation. In Findings of NAACL.
[40] [Chen, R., Wang, J., Wu, Y., & Li, X. (2024). Temporal knowledge graph question answering: A survey. arXiv preprint arXiv:2406.14191.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY