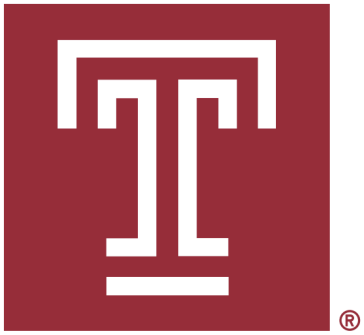

Temple University

Artificial Intelligence: Project Report

Student :
Aayush Acharya
TUID: 916538863

Submitted to :
Pei Wang

Date: December 15, 2025

1 Introduction

Large Language Models (LLMs) are trained on static corpora with a fixed knowledge cutoff, which limits their ability to reason about up-to-date facts. Retrieval-Augmented Generation (RAG) has emerged as a practical solution by grounding LLM outputs in external knowledge sources. This project explores *knowledge graphs* as a structured and interpretable retrieval substrate for RAG. Specifically, we study the problem of *graph retrieval*: how to select the most relevant nodes and edges from a large knowledge graph to support multi-hop question answering. The project proposes and implements an LLM-driven graph retriever that treats the LLM as a reasoning agent for iterative graph traversal rather than a pure text generator. Beyond the final system, this report emphasizes the learning process, design challenges, and lessons learned while building an end-to-end graph retriever for KGQA. The code and resources for this project are available in `https://github.com/aayushacharya/graph-retriever`.

2 Background

2.1 Graph Retrieval for KGQA

A graph retriever identifies and extracts a relevant subgraph from a large knowledge graph given a natural language query. Instead of operating on the entire graph, which is computationally infeasible, the retriever narrows down the search space to a query-specific context consisting of selected entities and relations. This subgraph is then used by downstream reasoning or answer-generation components.

Graph retrievers serve as a bridge between natural language queries and structured data, and they are a core component in KGQA systems, fact verification pipelines, and graph-augmented LLM frameworks.

2.2 Why Knowledge Graphs for RAG?

Compared to unstructured text, knowledge graphs offer:

- Explicit relational structure for multi-hop reasoning
- Reduced ambiguity through typed entities and relations
- Interpretability and traceability of reasoning paths
- Better support for logical and compositional queries

These properties directly address limitations observed in vector-based retrieval systems, which often retrieve semantically related but structurally irrelevant information.

3 Related Work

Traditional KGQA pipelines rely on entity linking and relation extraction followed by symbolic graph traversal. While precise, these pipelines are brittle and struggle with ambiguous or complex questions. Embedding-based retrievers project queries and nodes

into a shared semantic space [1, 4], enabling scalable similarity-based retrieval but often losing structural constraints.

More recent neural-symbolic approaches [2] integrate learning-based methods with symbolic reasoning, improving robustness on compositional queries. LLM-based systems such as text-to-SPARQL frameworks generate executable queries directly, but they are highly sensitive to syntax errors and ontology mismatches, especially in large graphs like Freebase.

This project builds on these ideas by using the LLM as a *graph traversal planner* rather than a direct query generator.

4 Learning process

Curiosity Question 1: How can LLMs trained with a knowledge cutoff provide insight on today's data? LLMs are trained on static corpora and therefore cannot directly encode facts that emerge after their training cutoff. Through this project, I learned that Retrieval-Augmented Generation (RAG) is the dominant paradigm for addressing this limitation. In particular, grounding LLM outputs in external knowledge sources allows the model to reason over up-to-date information at inference time. Among different retrieval substrates, knowledge graphs stood out due to their structured representation of entities and relations, which naturally support multi-hop and compositional reasoning.

Curiosity Question 2: How do you know which nodes and edges to retrieve from a knowledge graph? This question exposed a central challenge in graph-based RAG systems. Unlike text retrieval, graph retrieval requires selecting not just relevant entities but also the correct relational paths. Through experimentation, I found that embedding-based similarity alone is insufficient for this task. This motivated the idea of using an LLM as a reasoning agent that iteratively proposes graph traversal strategies, retrieves partial subgraphs, and reflects on what information is still missing.

Curiosity Question 3: How do you evaluate retrieval systems involving knowledge graphs? While designing the retriever, it became clear that evaluating retrieval quality in isolation is difficult. I discovered that the scientific community uses Knowledge Graph Question Answering (KGQA) benchmarks which provide an end-to-end evaluation framework, where retrieval quality is implicitly measured by downstream question answering performance.

Curiosity Question 4: What are the popular KGQA benchmarks? Through a survey of the literature, I identified several KGQA benchmarks, among which GrailQA is one of the most prominent. GrailQA is specifically designed to test generalization and multi-hop reasoning, making it well-suited for evaluating graph retrievers that aim to move beyond simple pattern matching.

Curiosity Question 5: How can an LLM translate natural language into queries without understanding the graph? A key insight from this project is that LLMs cannot reliably generate correct formal queries without explicit knowledge of the graph schema. To address this, the system provides structured context in the form of ontology

specifications, including entity types and relation identifiers. By constraining the LLM to reason only over the provided ontology, the system significantly reduces invalid queries and hallucinated relations.

5 Dataset

We use the **GrailQA** [3] dataset, a large-scale KGQA benchmark built on Freebase with approximately 64K questions. Each question is annotated with logical forms, enabling precise evaluation. GrailQA is explicitly designed to test generalization under three settings:

- **i.i.d.**: similar to training distributions
- **Compositional**: novel combinations of known patterns
- **Zero-shot**: unseen relations and structures

The dataset emphasizes multi-hop reasoning and is widely used to benchmark graph retrievers and KGQA models, making it a suitable testbed for this project.

6 Methodology

The implemented system follows an LLM-driven, iterative graph retrieval paradigm. Instead of embedding-based similarity search, the LLM generates *structural probes* that guide graph traversal. The full architecture is given in Figure 1.

6.1 Curiosity Question 2: How do you know which nodes and edges to retrieve?

This question lies at the core of graph retrieval. The project addresses it by decomposing retrieval into an iterative reasoning loop:

1. Translate the natural language question into abstract traversal plans (structural probes).
2. Execute these probes deterministically against the KG to retrieve a partial subgraph.
3. Reflect on the completeness of the retrieved subgraph.
4. Generate new probes targeting missing relations or entities.

This loop continues until sufficient information is retrieved or a depth limit is reached.

6.2 System Components

- **Structural Probe Generator**: Converts a question into schema-aware traversal instructions instead of rigid queries.
- **Reflection-Based Executor**: Executes probes as safe graph queries, ensuring retrieved facts exist in the KG.

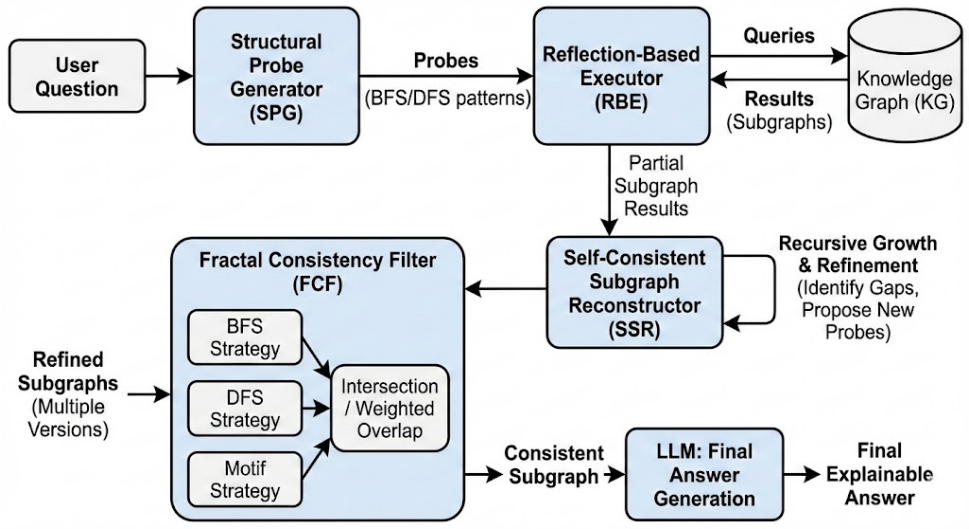

Figure 1: Proposed architecture of the graph-retriever system

- **Self-Consistent Subgraph Reconstructor**: Uses LLM reflection to identify gaps and guide further retrieval.
- **Consistency Filtering**: Runs multiple traversal strategies (e.g., BFS, DFS) and intersects results to reduce hallucination.

7 Experiments and Results

Due to time and resource constraints, full-scale quantitative evaluation on the GrailQA leaderboard was not completed. A small curated subset of 9 samples was created and evaluated on the baseline and proposed methods. The result is shown in Table 1.

A qualitative comparison was conducted against a baseline text-to-SPARQL approach. The baseline failed due to syntactic errors and incorrect relation usage, while the proposed retriever was more robust by construction, as it relied on simple, deterministic graph queries. An example of the comparison between two approaches is provided in Figure 2.

Model	EM
Gemini 2.5 pro Baseline	44.44
Proposed architecture	**55.55**

Table 1: Results on a small curated subset of questions.(Not the dev set of GrailQA)

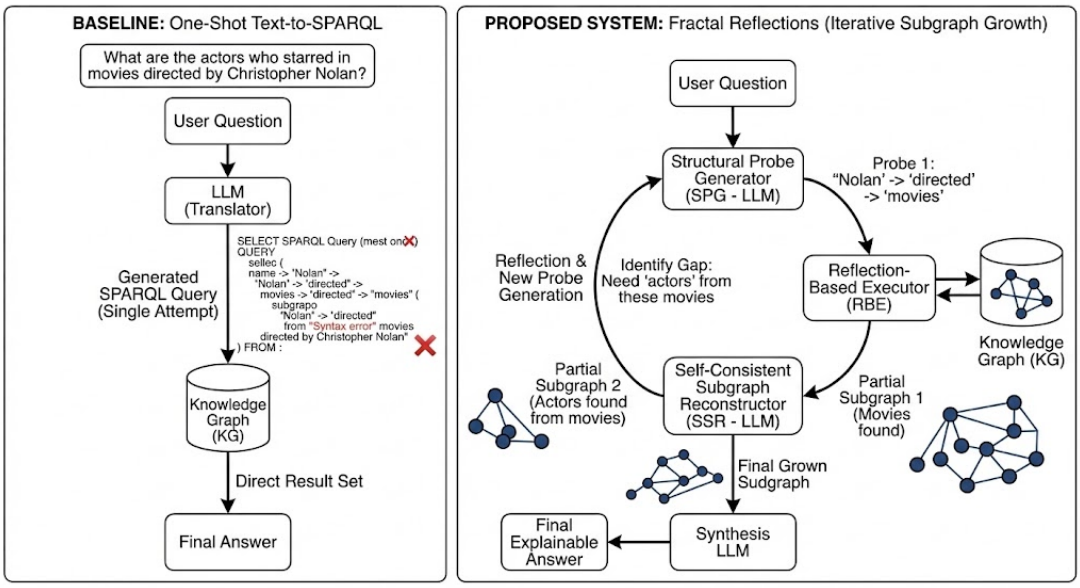

Figure 2: Qualitative comparison of the baseline approach with proposed system through an example.

8 Lessons Learned

This project was as much a learning exercise as a systems implementation. Key lessons include:

- **LLMs are better planners than executors**: Asking LLMs to generate traversal strategies is more reliable than asking them to generate full formal queries.
- **Structure matters**: Pure semantic similarity is insufficient for multi-hop reasoning over KGs.
- **Reflection improves retrieval**: Iterative self-evaluation helps identify missing information and reduces brittle failures.
- **Engineering challenges dominate**: Rate limits, ontology size, and query latency significantly shape system design.

9 Limitations

The current system lacks large-scale quantitative evaluation and is limited by rate costs of proprietary AI models. The approach also incurs higher inference cost than the baseline to multiple LLM calls per query. Additionally, the system has been tested primarily on Freebase-style graphs and may require adaptation for other KG schemas.

10 Conclusion

This project demonstrates that knowledge graphs are a powerful retrieval substrate for grounding LLMs beyond their training cutoff. By framing graph retrieval as an iterative,

LLM-guided reasoning process, the system avoids many pitfalls of direct query generation and embedding-based retrieval. The project provides a strong foundation for future work on robust, interpretable graph-augmented LLM systems.

References

[1] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26, 2013.

[2] Yu Gu, Xiang Deng, and Yu Su. Don't generate, discriminate: A proposal for grounding language models to real-world environments. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 4928–4949, 2023.

[3] Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, and Yu Su. Beyond iid: three levels of generalization for question answering on knowledge bases. In *Proceedings of the web conference 2021*, pages 3477–3488, 2021.

[4] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*, 2019.