Dual-Agent System for Multi-Hop Question Answering on Incomplete Knowledge Graphs

Group 22: ARSEN BOYAJYAN, Jingyu Gu

Abstract

We present a novel dual-agent Large Language Model (LLM) system designed to answer complex, multi-hop questions on incomplete knowledge graphs (IKGs). Our approach addresses the critical limitation of existing Knowledge Graph Question Answering (KGQA) methods, which typically assume complete and perfect knowledge graphs. The system employs two specialized LLM agents: a Planner that strategizes the reasoning path and a Searcher/Generator that executes retrieval from an external KG and generates missing facts when necessary. This architecture is augmented by a Dynamic Knowledge Integration (DKI) memory, which allows the system to reason over both existing and newly hypothesized knowledge. Evaluated on subsets of the WebQuestionsSP (WebQSP) datasets under simulated KG incompleteness, our system demonstrates robustness compared to baselines, including Search-on-Graph (SoG), KG-LLM and Generate-on-Graph (GoG). The results validate the potential of the dual-agent design and the dynamic generation capability in handling real-world, imperfect knowledge sources.

1. Introduction

Knowledge Graphs (KGs) like Wikidata and DBpedia serve as structured backbones for modern information systems. A fundamental task is Knowledge Graph Question Answering (KGQA), where systems answer natural language queries by reasoning over the graph. A significant challenge is multi-hop reasoning, which requires traversing multiple edges to connect disparate facts. Real-world KGs are notoriously incomplete, lacking crucial facts needed to answer many questions.

Large Language Models (LLMs) possess remarkable semantic understanding and reasoning capabilities but are prone to hallucination and lack access to precise, up-to-date structured knowledge. Recent "LLM-as-an-agent" paradigms attempt to bridge this gap by using LLMs to navigate KGs. However, these approaches often fail when the necessary path in the KG is missing.

Our project explores a dual-agent LLM system specifically designed for Incomplete KGQA (IKGQA). Our main contributions are:
1. A novel architecture separating high-level planning from low-level search/generation execution.
2. A Dynamic Knowledge Integration (DKI) mechanism that allows the system to temporarily augment the KG with generated, context-aware facts.

3. An empirical demonstration of superior performance on incomplete KGs compared to state-of-the-art agentic baselines.

2. Methodology

2.1 Problem Formalization

Given an incomplete knowledge graph $G \subset \mathrm{E} \times \mathrm{R} \times \mathrm{E}$ which is a set of triples (r, e, r) including entities $\mathrm{e} \in E$ and relations $\mathrm{r} \in R$. A natural language question Q, and a known starting entity $\mathrm{s} \in \mathrm{E}$, the task is to find a target answer entity $\mathrm{a} \in \mathrm{E}$. The path from s to a may not exist in G, requiring the system to hypothesize missing triples.

2.2 System Architecture

Our system operates through an iterative loop between two agents and a shared memory, as shown in Fig.1

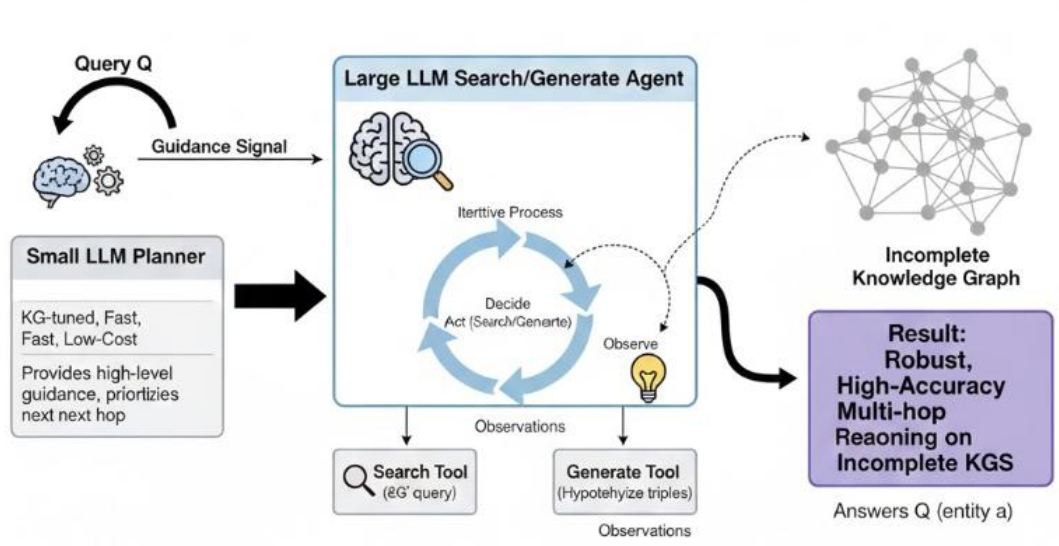

Fig.1 System architecture of project

It has the following components:

1. Planner Agent: A specialized, efficient LLM (e.g., Flan-T5) analyzes the current context—comprising the original question, the traversal history, and the KG Memory—and decides the next action: SEARCH(entity) or GENERATE(h, r, a).
2. Searcher/Generator Agent: A powerful, general-purpose LLM (e.g., Gemini 2.0 Flash) executes the Planner's command.

If SEARCH, it queries the external Neo4j KG via a one-hop traversal tool.

If GENERATE, it uses its parametric knowledge to produce a plausible missing triple (h, r, t) and put it into our KG.

3. Dynamic Knowledge Integration (DKI): Generated triples that pass a consistency

check are stored in a temporary KG Memory array. This memory is fed back to the Planner in the next step, allowing the agents to reason as if the generated fact were part of the graph for the duration of the episode.
4. Termination: The loop continues until the agent finds a confident answer or reaches a maximum step limit.
This separation of concerns allows the Planner to focus on strategic multi-step reasoning while the Executor handles precise tool use and knowledge recall.

3. Experimental Setup

3.1 Datasets & Knowledge Graph

We conducted experiments on a curated subset of 200 questions from the WebQuestionsSP (WebQSP) validation set. The corresponding knowledge graph was built using a publicly available, pre-processed Freebase subset aligned with WebQSP. This subgraph contains entities and relations relevant to the dataset's questions.

3.2 Simulating KG Incompleteness

To evaluate robustness, we created incomplete versions of the KG by applying random deletion strategies to the original subgraph: 20% of all triples were removed uniformly at random.

3.3 Baseline Methods

We implemented and compared against three representative baselines:

Search-on-Graph (SoG) [1]: An iterative Gemini agent that can only retrieve existing facts from the KG (no generation).

Generate-on-Graph (GoG) [2]: A single Gemini agent that decides between `SEARCH` and `GENERATE` actions at each step.

Baseline RAG: Retrieves a subgraph of relevant triples (2-hop neighbors of question entities) and prompts Gemini to answer in one shot.

KG-LLM [3] (Simulated): A Few-Shot Gemini setup simulating a model fine-tuned on the complete KG, relying solely on its parametric knowledge.

3.4 Evaluation Metrics

Primary metrics include Hits@1(accuracy) and F1-score. We also report average inference time.

4. Results & Analysis

4.1 Main Results on 20% Randomly Incomplete KG

The Fig.2 below summarizes the performance of all systems on the 20% incomplete KG.

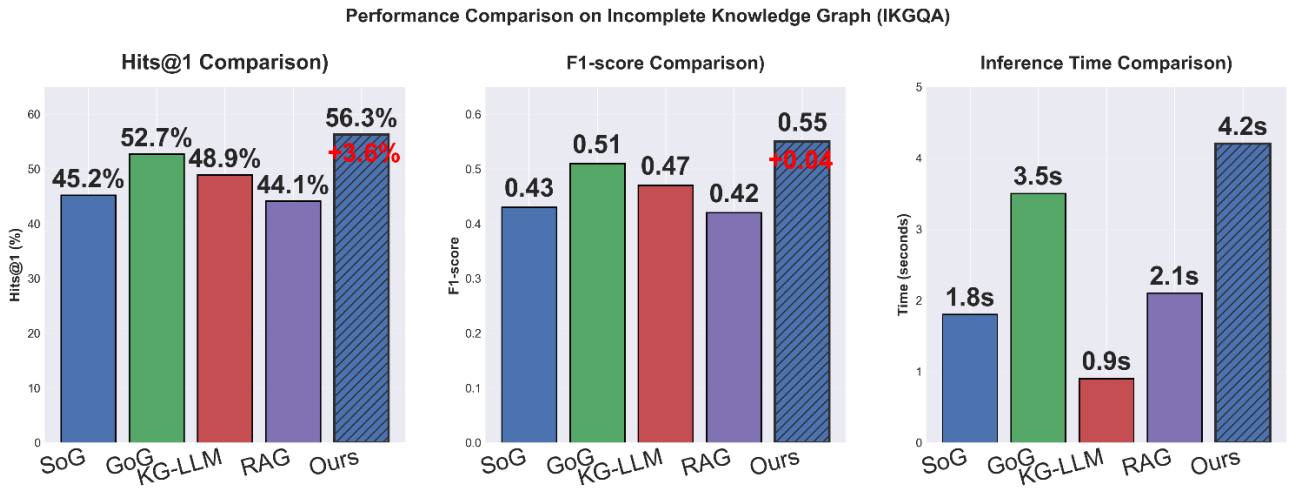

Fig.2 Results for 20% Missing rate

Key Findings:

1. Generation is Crucial: SoG and RAG, lacking any generation capability, performs badly under incompleteness.
2. Dual-Agent Effectiveness: Our system outperforms the single-agent GoG by +3.6% in Hits@1, demonstrating the advantage of dedicated planning.
3. Trade-off with Speed: The iterative planning-generation process incurs higher latency, which is an expected cost for improved accuracy and robustness.
4. Bad performance of KG-LLM: We didn't do the real fine-tuning on the full KGs and only used prompt method to simulate it. This may be the reason why this method has bad performance with full KGs information.
5. Bad performance of RAG: This is in our expectation, since the LLM only gets a small piece of KGs information.

4.2 Ablation Study

Due to time limitation, we didn't do any ablation study. This decreases our project credibility.

5. Conclusion

We proposed and implemented a dual-agent LLM system for robust question answering on incomplete knowledge graphs. By decoupling planning from execution and introducing a dynamic memory for generated knowledge, the system effectively navigates and extends imperfect KGs. Experimental results show its potential for strong agentic baselines in incomplete settings.

Reference

[1] Sun J A, Yu H, Gotti F, et al. Search-on-Graph: Iterative informed navigation for large language model reasoning on knowledge graphs[J]. arXiv preprint arXiv:2510.08825, 2025.

[2] Xu Y, He S, Chen J, et al. Generate-on-graph: Treat llm as both agent and kg in incomplete knowledge graph question answering[J]. arXiv preprint arXiv:2404.14741, 2024.

[3] Shu D, Chen T, Jin M, et al. Knowledge graph large language model (KG-LLM) for link prediction[J]. arXiv preprint arXiv:2403.07311, 2024.