Exploring Hybrid Reasoning with Knowledge graphs in LLMs

MTD 702 - MSc Thesis II Report

Debargha Ghosh
Entry No. 2022MAS7143

Project Supervisor : Dr. BS Panda

Department of Mathematics
Indian Institute of Technology Delhi

1 Introduction

In recent years, the field of question-answering systems has advanced significantly with the development of Retrieval-Augmented Generation (RAG) models, which combine large language models (LLMs) with retrieval mechanisms to access relevant external information. However, while RAG models have demonstrated effectiveness in leveraging external data for generating accurate and contextually appropriate responses, they are often limited by a lack of deeper reasoning capabilities over structured information. This limitation can result in responses that miss complex relationships or nuanced insights embedded within data.

To address this, integrating knowledge graph reasoning into RAG systems has emerged as a promising approach. Knowledge graphs (KGs) offer structured representations of entities and their interconnections, allowing for advanced reasoning and inference beyond simple retrieval. By enhancing RAG with a knowledge graph reasoning layer, it becomes possible to harness the graph's relational structure to derive insights that consider both direct and indirect connections, enabling a richer understanding of the data. This thesis explores the integration of knowledge graph reasoning within RAG to improve the accuracy, interpretability, and reasoning depth of responses in complex question-answering tasks. Through this approach, the research aims to demonstrate how a hybrid model leveraging both unstructured and structured data sources can achieve enhanced performance in information retrieval and question-answering applications

1.1 LLM Knowledge Limitations

- **Hallucination:** Hallucination refers to instances when the model generates information that appears factual but is actually incorrect or fabricated. These inaccuracies stem from limitations in the model's understanding or gaps in its training data, resulting in responses that seem plausible but are ultimately inaccurate.

- **Knowledge Cutoff:** Knowledge cutoff refers to the point in time after which a model has no information. The model cannot access or understand events, discoveries, or data occurring after this date, limiting its responses on recent topics.

RAG offers a technique to mitigate these challenges. It enables you to provide the model with access to external data sources, mitigating hallucinations by introducing factual context and overcoming knowledge cut-offs by incorporating current information.

1.2 Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) is a versatile framework that grants Large Language Models (LLMs) access to external data beyond their training data. It enables LLMs to tap into external data sources, including knowledge bases, documents, databases, and the internet, during runtime. To efficiently retrieve information from documents, a common practice is to index them in a vector database, using embedding vectors that capture semantic meaning. Additionally, chunking is applied to large documents to improve relevance and reduce noise, ultimately enhancing the model's responses by providing specific contextual information.

1.2.1 The Framework

There are two key models within RAG framework:

- **RAG-Sequence Model:** This model utilizes the same retrieved document to generate the complete sequence. It treats the retrieved document as a single latent variable that is marginalized to get the sequence-to-sequence probability via a top-K approximation. In this approach, the top K documents are retrieved using the retriever, and the generator computes the output sequence probability for each document. These probabilities are then combined through marginalization.

- **RAG-Token Model:** In this model, a different latent document can be drawn for each target token, allowing the generator to choose content from several documents when producing an answer. Similar to the RAG-Sequence model, it retrieves the top K documents and then generates a distribution for the next output token for each document. This process is repeated for each output token, with marginalization occurring accordingly.

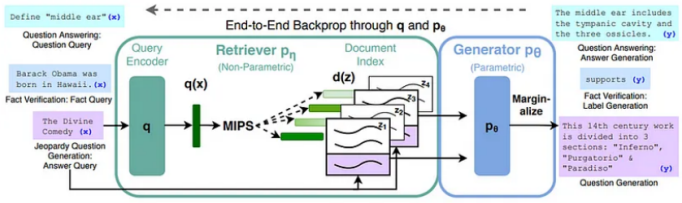
RAG-Token vs. RAG-Sequence

$$p_{\text{RAG-Token}}(y|x) = \prod_{i}^{N} \sum_{z \in \text{top-}k(p(\cdot|x))} p_\eta(z_i|x) p_\theta(y_i|x, z_i, y_{1:i-1}).$$

$$p_{\text{RAG-Sequence}}(y|x) = \sum_{z \in \text{top-}k(p(\cdot|x))} p_\eta(z|x) \prod_{i}^{N} p_\theta(y_i|x, z, y_{1:i-1}).$$

1.2.2 Components of RAG

- **Retriever:** The Retriever (DPR), which retrieves relevant documents based on a query and a document index. The retrieval component is based on Dense Passage Retrieval (DPR) and employs a bi-encoder architecture with dense and query representations derived from BERT. The top-K documents with the highest prior probability are selected, making use of a Maximum Inner Product Search (MIPS) algorithm.
- **Generator:** The Generator component, based on BART-large, is responsible for generating sequences. It combines input with retrieved content simply by concatenating them.



1.2.3 Training

During training, both the retriever and generator are trained jointly without direct supervision on which document to retrieve. The training objective minimizes the negative marginal log-likelihood of each target.

1.2.4 Decoding

In the decoding phase, RAG-Sequence and RAG-Token models require different approaches. RAG-Token uses standard autoregressive sequence generation with a beam decoder. In contrast, RAG-Sequence runs beam search for each document and scores hypotheses using the generator probabilities.

1.2.5 Rag Implementation

Implementing RAG workflows can be complex, involving multiple steps from accepting the user prompt to querying the database, chunking documents, and training the retrieval and generation processes. So, instead of training, we will be using pre-trained SOTA open source models like llama3 from ollama. Frameworks like LangChain simplify this process by providing modular components to work with LLMs and augmentation techniques like RAG.

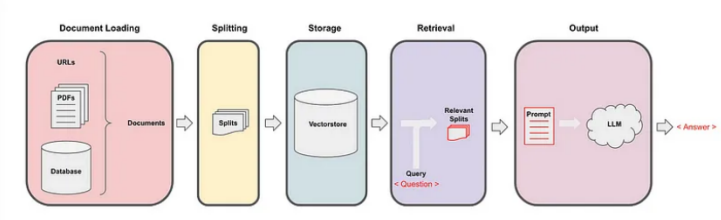


Figure 1: RAG workflow using LangChain

2 Knowledge Graphs

Knowledge graphs are structured representations of information where entities, concepts, and their relationships are organized as nodes and edges in a graph format. These graphs enable systems to understand and navigate the relationships between different pieces of information, facilitating a deeper understanding of context and meaning.

2.1 Structure of Knowledge Graphs

In a knowledge graph, entities (e.g., people, organizations, locations) are represented as nodes, while the connections between these entities (e.g., "works for," "located in") are represented as edges. This structure creates an interconnected web of knowledge, allowing for the modeling of complex relationships within data. Knowledge graphs are often designed to capture domain-specific knowledge or cover broad knowledge bases (such as Wikidata or Freebase), with the goal of supporting reasoning and inference over structured data.

2.2 Building Knowledge Graphs from Text

To construct a knowledge graph from unstructured text, an extraction process is required. This typically involves two key steps:
1. **Entity Recognition and Linking**: Identifying important entities within the text and linking them to standardized representations (e.g., recognizing "Apple" as a company).
2. **Relationship Extraction**: Detecting relationships between entities and organizing them as directed edges. This process may use dependency parsing, transformer-based language models, or specialized relation extraction models.

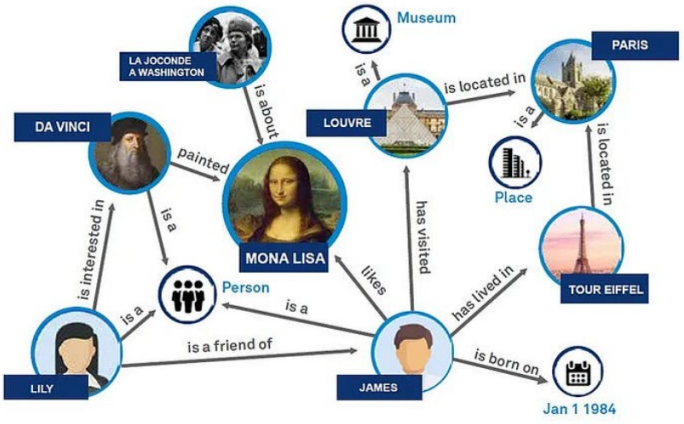


Figure 2: Example of knowledge graph

2.3 Current Limitation of RAG

In relationship-heavy datasets, where the structure and context of connections between entities are critical, traditional RAG models face several limitations:
1. **Limited Multi-Hop Reasoning:** RAG struggles with complex, multi-step relationships, often retrieving only directly related documents without following deeper relational paths.
2.**Shallow Semantic Understanding:** Dense embeddings lack the specificity to capture nuanced relationship types (e.g., "founded by"), making responses less accurate for queries requiring precise connections.
3.**Inability to Prioritize Relationships:** Due to high-dimensional vector representation, RAG may retrieve content based on similarity alone, missing context-specific priorities (e.g., personal vs. professional links).
4.**Flat Document Structure Reliance:** RAG flattens hierarchical data, losing relational depth critical in structured, relationship-heavy contexts.
5.**High Hallucination Risk:** RAG may generate plausible-sounding but incorrect answers, especially on complex, multi-relational queries.

3 Hybrid Retrieval for RAG

We will be using wikipedia dataset of Elizabeth1, a relationship heavy dataset where traditional RAG fails in even some of the 2 hop question-answering tasks, and combination of multiple pdfs of criminal investigation reports. Our solution enhanced the answer generation on both the datasets using structured reasoning through graphs.

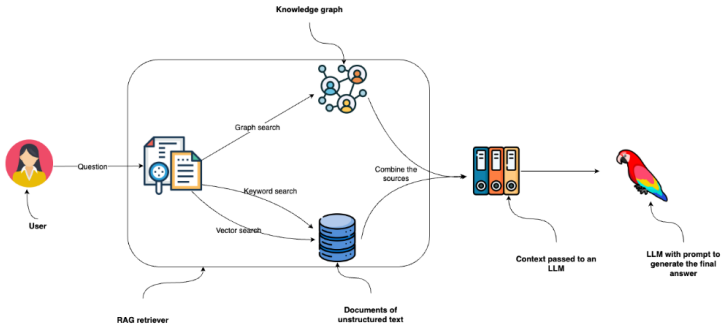


Figure 3: Workflow

Algorithm 1 Hybrid RAG-based Retrieval System

Input: Raw data, D, and user query, Q

Chunking and Splitting

Segment data D into smaller chunks for efficient processing.

Create chunks based on document structure or maximum token length limits.

Knowledge Graph Creation, G

Use LLM-based graph transformer to generate a knowledge graph from the chunks.

Ingest the generated knowledge graph into a graph database.

Hybrid Retriever Module

Initialize *Unstructured Retriever*:

- Retrieve relevant nodes by generating embeddings for graph nodes using LLM.

Initialize *Structured Retriever*:

- Detect entities in the user query Q.
- For each detected entity, apply a Cypher query template to retrieve relevant multi-hop neighborhoods(subgraph) from G.

RAG Chain

Query Rewriting mechanism to enable conversational follow-up questions.

Prompt LLM combining the context from the hybrid retriever with Q to guide the response generation.

Invoke RAG chain to generate response to the query.

if additional query, Q^* exists **then**

Retrieve previous interactions from chat history to maintain continuity.

Re-run the hybrid retriever, incorporating both Q^* and chat history.

Re-run the RAG chain with Q^*.

else

Generate a response based only on the initial query context without additional history.

end if

4 KG and reasoning path in wikidata

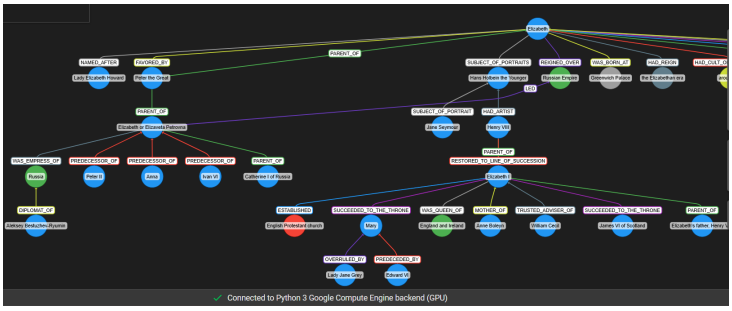


Figure 4: Snapshot of KG Generated

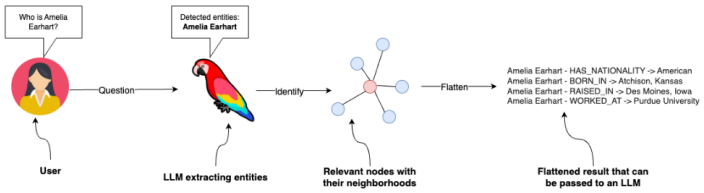


Figure 5: Graph Retriever

The graph retriever starts by identifying relevant entities in the input. For simplicity, we instruct the LLM to identify people, organizations, and locations.

The structured retriever function starts by detecting entities in the user question. Next, it iterates over the detected entities and uses a Cypher template to retrieve the neighborhood of relevant nodes.

```
print(structured_retriever("Who is Elizabeth I?"))

WARNING:neo4j.notifications:Received notification from DBMS server: {severity: WARNING} {code: Neo.ClientNotification.Statement.FeatureDeprecat
Elizabeth I - WAS_QUEEN_OF -> England and Ireland
Elizabeth I - SUCCEEDED_TO_THE_THRONE -> Mary
Elizabeth I - ESTABLISHED -> English Protestant church
Henry VIII - PARENT_OF -> Elizabeth I
Elizabeth's father, Henry VIII - PARENT_OF -> Elizabeth I
Anne Boleyn - MOTHER_OF -> Elizabeth I
Henry VIII - RESTORED_TO_LINE_OF_SUCCESSION -> Elizabeth I
James VI of Scotland - SUCCEEDED_TO_THE_THRONE -> Elizabeth I
William Cecil - TRUSTED_ADVISER_OF -> Elizabeth I
Queen Elizabeth I - PORTRAYED_IN -> English royal portraits
Queen Elizabeth I - CONTAINS_SYMBOLIC_OBJECTS -> roses and prayer books
Queen Elizabeth I - LAYERED_ICONOGRAPHY -> globes, crowns, swords and columns
Queen Elizabeth I - REPRESENTED_IN -> classical allusions
Queen Elizabeth I - SAT_FOR_ARTISTS -> Hilliard, Cornelis Ketel, Federico Zuccaro or Zuccari, Isaac Oliver, and Marcus Gheeraerts the Younger
Nicholas Hilliard - APPOINTED_OFFICIAL_MINIATURIST -> Queen Elizabeth I
```

Figure 6: Relationship rich context extracted

5 Future Work

- Novel Techniques for Evaluation of Knowledge graph construction.
- Learning the representation of the context of optimal response to complex queries using GNNs.

6 References

1. **Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks** - Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, Douwe Kiela
https://arxiv.org/abs/2005.11401

2. **From Local to Global: A Graph RAG Approach to Query-Focused Summarization** - Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Jonathan Larson
https://arxiv.org/abs/2404.16130