CSCI 6365 Advanced Machine Learning

A Travel Agent LLM: Integrating LangChain, Knowledge Graphs, and Retrieval-Augmented Generation for Hotel and Attraction Recommendations

Shi Qiu

The George Washington University

Prof. Sardar Hamidian & Prof. Armin Mehrabian

Abstract—This project presents a conversational travel assistant integrating a Large Language Model (LLM), LangChain, a custom Knowledge Graph (KG), and Retrieval-Augmented Generation (RAG). The assistant recommends hotels, attractions, and transportation based on user queries by combining structured data from a Kaggle hotel dataset with unstructured semantic knowledge from the KG. Evaluation shows significant improvements in both hotel and non-hotel queries with KG and RAG integration, highlighting their role in enhancing the assistant's capabilities.

Keywords—*Travel Assistant, LLM, LangChain, Knowledge Graph, Retrieval-Augmented Generation, Hotel Recommendation*

1. Problem Definition and Understanding

1.1. Problem Definition

This project aims to develop a travel assistant that helps users find hotels, attractions, and transport options via natural language queries. The system:

- Classifies queries as hotel or non-hotel related.
- Uses rule-based filtering on a large CSV hotel dataset for hotel queries.
- Performs semantic searches on a custom Knowledge Graph for non-hotel queries.
- Generates responses with an LLM, integrating retrieved data and context.
- Maintains conversation memory for follow-up queries.

This ensures effective use of structured and unstructured data to enhance user satisfaction and query relevance.

1.2. Understanding the Domain and Data

Travel decisions involve factors like hotel features, location preferences, and local attractions. **Data sources:**

- **Hotel Dataset:** A large CSV from Kaggle with hotel metadata (e.g., *HotelName*, *HotelRating*, *HotelFacilities*). Used for filtering hotel-related queries.
- **Knowledge Graph:** Custom-built, encoding relationships such as `HAS_ATTRACTION`, `HAS_TRANSPORT`, and `HAS_HOTEL`. Initially based on New York City, it can be extended to other destinations.

By integrating structured hotel data and a KG for attractions and transport, the system can intelligently respond to diverse travel queries.

2. Data Preparation and Preprocessing

2.1. Data Cleaning and Handling Missing Values

The hotel CSV was large and varied. We:

- Trimmed whitespace and standardized facility names.
- Removed entries missing essential attributes for reliability.

For the KG, data cleaning ensured consistent naming for nodes and edges.

2.2. Feature Selection and Feature Engineering

Instead of extensive feature engineering, the system uses:

- Simple filters from user queries (e.g., city, star rating, facilities).
- Textual embeddings for semantic search over the KG.

We did **not** use binary features like `NEAR` for proximity. Instead, relevance is inferred through semantic similarity during retrieval.

3. Model Selection and Development

3.1. Choice of Appropriate Models

The system's capabilities derive from:

1. **LLM-based Intent Classification:** Classifies queries into hotel or non-hotel categories using LangChain.
2. **Filtering for Hotel Queries:** Applies filters on the CSV (e.g., city, star rating).
3. **Vector-based Retrieval for Non-Hotel Queries:** Uses OpenAI embeddings and FAISS to perform semantic searches over the KG.

3.2. Correct Implementation of the Process

The workflow, depicted in Figure 1, involves the following steps:

1. **User Query:** User asks a question in natural language.
2. **Intent Classification:** LLM determines if the query is hotel-related or not.
3. **Data Retrieval:**
    - **Hotel Queries:** Apply filters on the hotel CSV to find matches.
    - **Non-Hotel Queries:** Use embeddings to search the KG for relevant attractions or transport information.
4. **Answer Generation:** Combine retrieved data into a prompt for the LLM to generate a coherent response.
5. **Follow-up Support:** Maintain conversation memory to handle follow-up queries seamlessly.

This approach leverages LLMs, semantic embeddings, and rule-based filtering without complex feature engineering or ranking models.

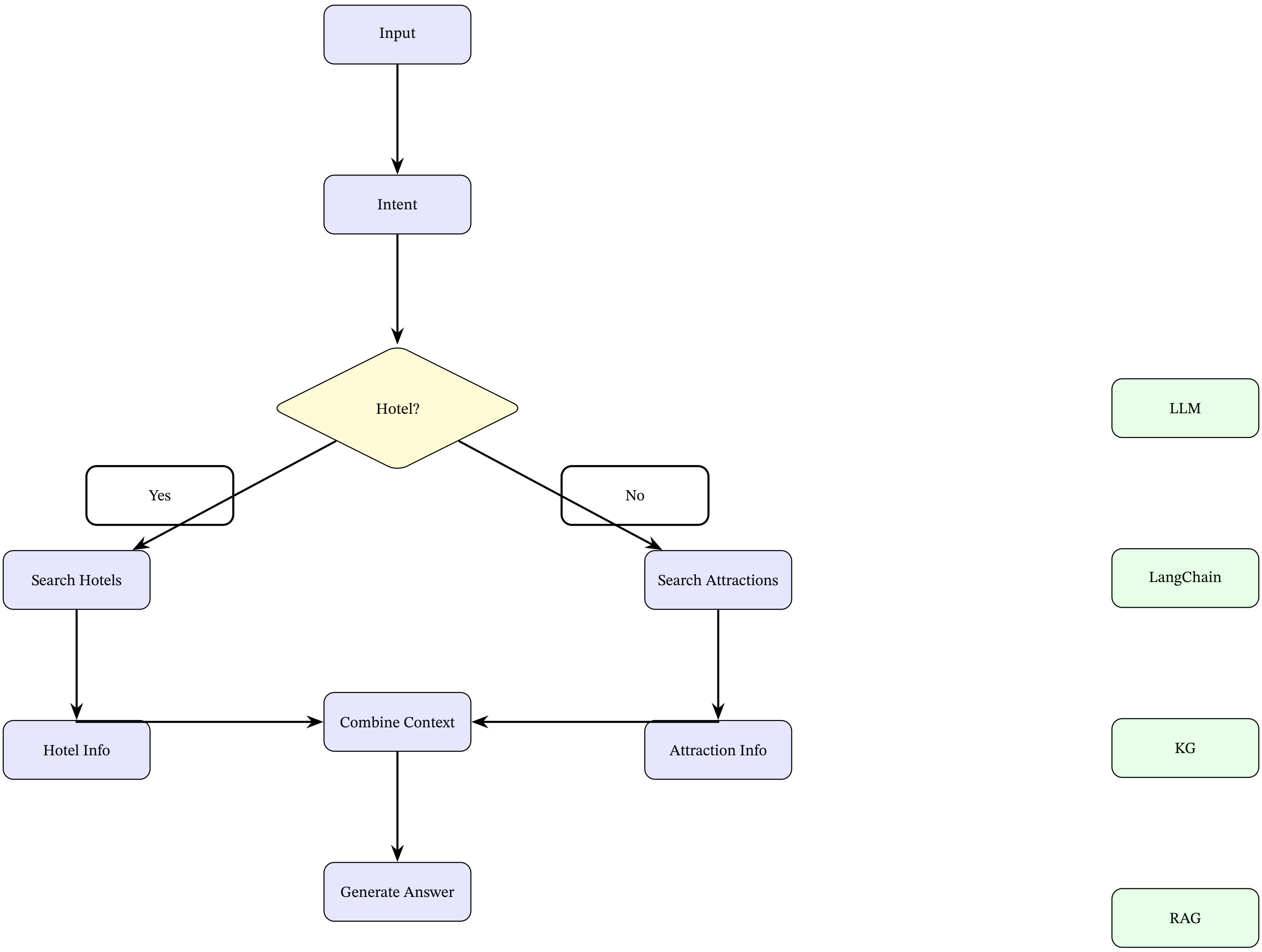

Figure 1. Integrated Workflow of the Travel Assistant System

4. Evaluation and Performance Metrics

4.1. Categorical Metrics Comparison

Table 1. Categorical Metrics Comparison Between Systems with and without RAG Integration

Metric	With RAG	Without RAG
Hotels Provided	3,994	Numerous (exact number not provided)
4-Star Hotels	490	Numerous
5-Star Hotels	89	Numerous
Correct Contact Info	Yes	No, Incorrect, Made-up
Info on Getting There	Yes	Yes
Response Cost	High	Medium
Response Time	Medium	Low

4.2. Analysis of Model Performance

Table 2. Model Performance Statistics (Last 7 Days)

Statistic	Value
Run Count	267
Total Tokens	80,001 / $1.02
Median Tokens	221
Error Rate	2%
% Streaming	0%
Latency	P50: 0.68s, P99: 10.13s

Over the past week, the system executed 267 runs, using 80,001 tokens at a cost of $1.02. The median token count was 221 per run. It maintained a 2% error rate with no streaming issues, indicating high reliability. Latency was 0.68 seconds for 50% of queries and 10.13 seconds for 99%, ensuring prompt responses. These metrics demonstrate efficient performance, with ongoing monitoring to optimize latency for outliers.

5. Conclusion

We developed an advanced travel assistant system that integrates Large Language Models (LLMs), a comprehensive hotel dataset, and a custom Knowledge Graph (KG). The system effectively distinguishes between hotel-related and non-hotel-related queries using intent classification. For hotel queries, it employs rule-based filtering on a robust CSV dataset, while for non-hotel queries, it utilizes semantic embedding-based retrieval from the KG to provide relevant recommendations.

The integration of the Knowledge Graph significantly enhances the system's ability to deliver accurate and contextually appropriate responses, especially for non-hotel inquiries. This results in improved user satisfaction and query relevance compared to systems without KG integration. Additionally, the use of Retrieval-Augmented Generation (RAG) further strengthens the assistant's response generation capabilities.

Overall, this project demonstrates the effectiveness of combining structured data, semantic knowledge, and advanced language models to create a versatile and user-friendly travel assistant. Future work could focus on expanding the Knowledge Graph to include more destinations and incorporating real-time data for dynamic recommendations.

6. References

- LangChain Documentation: https://python.langchain.com/
- OpenAI Embeddings: https://platform.openai.com/docs/guides/embeddings
- FAISS Vector Store: https://faiss.ai/
- NetworkX: https://networkx.org/
- Kaggle Hotel Dataset: https://www.kaggle.com/datasets
- Retrieval-Augmented Generation: https://arxiv.org/abs/2005.11401
- Knowledge Graphs: https://en.wikipedia.org/wiki/Knowledge_graph
- Bilibili Video: https://www.bilibili.com/video/BV1FtytYeEQM
- Tourist Attractions Prediction Paper: http://bos.itdks.com/6b6277a4f30b41aeb7a56bd5524dbe49.pdf
- Tourist Attractions Prediction with Enhanced Location Knowledge Graph: https://ieeexplore.ieee.org/document/10466836
- TravelRAG: A Tourist Attraction Retrieval Framework Based on Multi-Layer Knowledge Graph: https://www.mdpi.com/2220-9964/13/11/414#
- Agent Framework Overview by Lilian Weng: https://lilianweng.github.io/posts/2023-06-23-agent/
- LangChain RAG Tutorial: https://python.langchain.com/docs/tutorials/rag/

