LangGraph RAG agent - Travel planner advisor

Matteo Farè ID: 989345

University of Milan, Department of Computer Science.

Contributing authors: matteo.fare1@studenti.unimi.it;

Abstract

This project builds a RAG agent using LangChain, LangGraph, and an open-source LLM from Ollama. It retrieves travel-related information by crawling a website, stores the data in a Chroma vector database, and uses Nomic embeddings for document processing. The result is a travel planner advisor that provides personalized travel suggestions.

1 Introduction

The field of **artificial intelligence** and **natural language processing (NLP)** has seen rapid advancements, with **large language models (LLMs)** becoming highly proficient in **text generation** and understanding.

This study explores the development of a **Retrieval-Augmented Generation (RAG) agent** system. The core libraries employed are **LangChain** and **LangGraph** for the the system's architecture, while the Llama3.2 model from **Ollama** serves as the adopted LLM, providing the main NLP and conversational capabilities. The system integrates **Nomic** for encoding textual data into dense vector representations, and stores retrieved documents in a **Chroma** vector database. These technologies allow the system to efficiently handle user queries and generate personalized answers.

The goal is to build a simple **travel advisor**[1] that can adapt to user needs, suggesting destinations, itineraries, and tips based on real-world data.

2 Data, Embedding, Vector Storage, and Language Model

2.1 Dataset

The **dataset** used for this project is built starting from the website *"Italia.it"*[1], which provides exhaustive information about travel destinations, itineraries, local history and culture highlights in the italian country. A custom **web-crawling function** was developed to extract this information, focusing on scraping **travel-related pages**. Specifically, ten pages are retrieved for each region of Italy to ensure that a sufficient amount of information is available for various locations, covering at least the most renowned destinations.

2.2 Nomic Embed

Once extracted from the web, the data underwent an embedding process to prepare it for **efficient retrieval**. Embeddings are **dense vector representations** of the textual data, allowing the system to compare and search for similar content effectively. In this project, **Nomic embed**, an **open-source embedding model**, it is used to generate **vector embeddings** for each document.

2.3 Chroma DB

To store and manage these embeddings, it is employed **Chroma DB**. Chroma is a powerful **open-source** AI application database used to store and manage high-dimensional embeddings of documents. Its use enables fast and efficient retrieval of **relevant** documents based on their **semantic similarity** to user queries.

Documents in Chroma are represented as vectors, generated by Nomic embeddings. These vectors allow the system to perform **similarity searches**, fetching the most relevant documents in response to a query. By storing the data in this format, this technology ensures that the retrieval process is both scalable and optimized for high performance, even when dealing with large datasets.

The integration of Chroma, with the rest of the RAG system is **seamless**. Whenever a user query is received, it retrieves the **top-k** closest documents (in this case, k was set to 5), which are then used in the various steps by the system.

2.4 Llama Language Model

The Large Language Model Meta AI, or Llama, is a **transformer-based model** developed by Meta, optimized for a wide range of NLP tasks such as **text generation**, **conversational AI**, and **contextual analysis**. Its architecture employs a **self-attention mechanism**, which enables it to understand and generate complex, contextually coherent responses by analyzing input sequences.

[1] Website home page: https://www.italia.it/en

3 LangChain and LangGraph

3.1 LangChain

LangChain is a framework designed to integrate language models with **external tools** like APIs, databases, or vector stores. It allows to build systems that leverage LLMs in combination with retrieval or computation engines, enhancing the model's capabilities with external, **real-world data**.

3.2 LangGraph

LangGraph extends LangChain by enabling the creation of **graph-based workflows**, where **tasks** (nodes) and their **connections** (edges) are explicitly defined. This structure introduces flexibility into complex decision-making processes, where the system can take **different paths** depending on the data retrieved or the nature of the query.

4 Retrieval-Augmented Generation (RAG)

4.1 Standard RAG

Retrieval-Augmented Generation (RAG) is a hybrid approach combining document retrieval with generative language models. In a typical RAG system, a query is processed in two stages:

- **Retrieval**: The system fetches relevant documents or information from a database or external source based on the user's query.

- **Generation**: A language model then generates a response by incorporating both the retrieved documents and its internal knowledge.

This approach enhances the **quality of the responses** by grounding them in **factual**, up-to-date data, addressing one of the key limitations of language models.

4.2 RAG Agents

A **RAG agent** takes the standard RAG concept a step further by introducing decision-making capabilities within the system. Unlike a basic RAG framework, which sequentially retrieves documents and generates responses, a RAG agent **controls the flow of operations** dynamically. The agent can:

- **Analyze query**: the system evaluates whether the query is relevant. If not, the agent can redirect the query to another process or refine the user's question.

- **Select data path**: depending on the query and the retrieved data, the agent decides which set of documents is most appropriate for response generation, or if additional retrieval is necessary.

- **Evaluate answers**: the RAG agent can check for hallucinations (when a model generates incorrect information) and try alternative strategies to correct such errors.

- **Adaptive workflows**: RAG agents can alter their behavior dynamically based on the specific requirements of each query, unlike a standard RAG system, which follows a fixed retrieval-generation pipeline.

5 Graph Structure and System Components

The system is built as a **graph-based workflow** using **LangGraph**. Each node in the graph represents a distinct task, and edges between nodes determine the flow of execution based on the outcomes of these tasks. The key components, nodes, and edges of the graph represented in the following **Figure** 1 are described below.

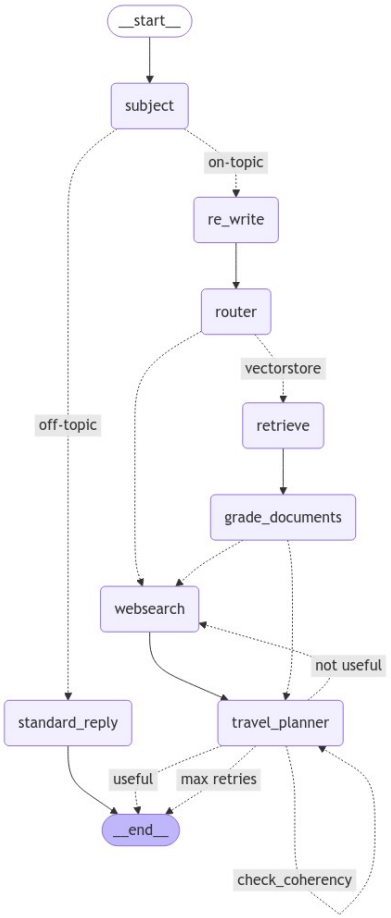

Fig. 1 Graph: travel planner advisor

5.1 Components

The various components defining the tasks are implemented using classes that describe the behavior in distinct stages. These classes are structured with the **LangChain** framework that is used to build structures defined as **chains**, which are composed of linked prompt templates and LLM functions.

For example, the `QueryIsRelevant` class determines whether the user's question is related to the topic of travel or not. The use of `with_structured_output` method define a version of the language model where the output conforms to the structure defined in the class. By taking this last element and chaining it with the designed **prompt template**, a chain is defined in which the output of the left side (the prompt template) is fed as input into the right side (the LLM with structured output).

5.2 Nodes

The system's flow is organized through a series of **node functions**, each representing a distinct task within the state-based graph. The nodes work in sequence, forming a **pipeline** from query input to final answer generation:

- **subject**: this node is responsible for determining the subject matter of the user's query. When a query is received, this function evaluates whether the question is about a relevant topic, like travel, or if it is off-topic.

- **standard_reply**: if the query is off-topic, this node provides a standard response, informing the user that the system can only assist with travel-related queries.

- **router**: this node determines whether the user's query should be routed to a document-based retrieval system (RAG) or if the system needs to perform a web search. It relies on a model that analyzes the query and outputs the most appropriate path. The result is stored in the state as the route key, guiding the system to either retrieve documents or perform a web search.

- **rewrite_query**: if the router takes through the vectorstore (i.e., the dataset), this node takes the user's original query and attempts to rephrase or refine it to create a more effective search query. This is particularly useful if the original query is too vague or complex.

- **retrieve**: once the query is refined, the retrieve node fetches relevant documents from the dataset. This node updates the state by appending the relevant documents, which will later be graded for their usefulness.

- **grade_documents**: this node evaluates whether the retrieved documents are relevant to the query. If fewer than half of the documents are relevant, it sets the state to initiate a web search.

- **planner**: this node is responsible for generating a travel plan based on the user's question and relevant documents. It uses the documents fetched earlier and combines them with the query context to form a coherent travel itinerary. If the plan isn't grounded in the documents or doesn't answer the user's question, the system may enter a retry loop.

- **web_search**: if the query cannot be answered using the dataset, this node initiates a web search to gather the necessary information.

5.3 Edges

The edges between the nodes represent the **flow of information**, ensuring that queries move through the graph in a logical order.

- **route_subject**: this edge decides whether the query is about travel or some other subject. Based on the subject score from the subject node, it routes the query to either the `standard_reply` node (if off-topic) or the `rewrite_query` node (if on-topic).

- **route_question**: this edge determines whether the query should trigger a web search or document retrieval from the vectorstore. The decision is based on the outcome of the router node, which evaluates where the best source of information lies for the given query.

- **decide_to_plan**: after the `grade_documents` node evaluates the relevance of the documents, this edge determines whether the system should proceed to planning or initiate a web search. If the documents are not relevant enough, it triggers the `web_search` node; otherwise, it moves to `planner` node.

- **check_and_answer**: this edge determines whether the output generated by the planner is grounded in the documents and coherent with the user's question. If the plan is deemed satisfactory, the flow ends. If it is not, the system may re-enter a retry loop or trigger a web search.

5.4 Graph Workflow Summary

The graph workflow starts by determining the relevance of the query's subject using the subject node. Based on this, the query is either rewritten for clarity or routed to a standard reply if it's off-topic.

Next, the router node decides whether to retrieve documents from a vector store or perform a web search. The retrieved documents are graded for relevance, determining whether they are useful for generating a travel plan. If relevant documents are found, the planner node define a plan. If not, the system falls back to a web search.

Throughout the process, the state of the graph is updated maintaining context at each step to ensure informed decision-making.

6 Results

This section presents various results obtained from test runs with the defined RAG agent. Each example demonstrates the **agent's behavior** when interacting with different types of queries and contexts, effectively showcasing the adaptive capability of the RAG framework. These results, reported in the **Appendix A**, illustrate how the agent processes travel-related inquiries, accesses relevant information from the vector store or web search, and responds appropriately based on the query's specific requirements.

7 Conclusion

This study presents the development of a **Retrieval-Augmented Generation (RAG)** agent system for a travel planning application. The system integrates the **Llama3.2** model from Ollama with **LangChain** and **LangGraph**, enabling it to handle user queries efficiently and generate personalized travel advice. By utilizing **Nomic embeddings** and **Chroma** for vector storage, the system ensures high-quality retrieval and semantic search of travel-related documents.

The **graph-based workflow**, implemented with LangGraph, allows for dynamic decision-making at every stage of the query handling process. This design supports complex operations such as query relevance analysis, document retrieval, planning, and response evaluation. The adaptability of the **RAG agent** enhances user interaction, making the system capable of offering relevant, **fact-grounded** travel recommendations based on real-time data and user preferences.

Appendix A Results

Query (1):

Hi, I'd like to plan a vacation in Italy, in the Sardinia region. What do you suggest me?

Path (1):

—ESTABLISH SUBJECT—
===SUBJECT: TRAVEL===
—RE-WRITE QUESTION—
—ESTABLISH ROUTE—
===ROUTE: RAG===
—RETRIEVE DOCS—
—CHECK DOCUMENT RELEVANCE—
—DOCUMENT RELEVANCE—
—DOCUMENT RELEVANCE—
—DOCUMENT RELEVANCE—
—DOCUMENT RELEVANCE—
—DOCUMENT RELEVANCE—
===EVALUATE DOCUMENTS===
===DECISION: GO TO PLANNER===
—PLANNER—
===CHECK COHERENCY===
===DECISION: PLANNER IS COHERENT WITH THE DOCUMENTS===
===EVALUATE PLANNER ANSWER===
===DECISION: PLANNER ADDRESSES QUESTION===

Answer (1):

Sardinia is a beautiful island in the Mediterranean Sea, known for its stunning beaches, crystal-clear waters, and rich history. Here are some suggestions for your vacation:

Must-see destinations:
1. Costa Smeralda: A picturesque coastline with turquoise waters and white-sand beaches.
2. **Cagliari**: The capital city of Sardinia, known for its historic center and Roman amphitheater.
3. **Alghero**: A charming coastal town with a beautiful old town.
4. **Chia**: A popular beach resort town on the south coast.

Beach activities:
1. **Snorkeling and diving**: Explore the island's incredible marine life and coral reefs.
2. **Surfing and windsurfing**: Take advantage of Sardinia's consistent winds and waves.
3. **Beach relaxation**: Enjoy the sun, sand, and sea at one of the many beautiful beaches.

Cultural experiences:
1. **Visit ancient ruins**: Explore the island's rich history at sites like Tharros, Nora, and Su Nuraxi di Barumini.
2. **Wine tasting**: Sample Sardinia's famous Cannonau wine and learn about the island's wine production.
3. **Traditional cuisine**: Try local specialties like suckling pig, seafood risotto, and seada (fried dough balls).

Outdoor activities:
1. **Hiking and trekking**: Explore the island's scenic trails and enjoy breathtaking views.
2. **Cycling**: Rent a bike and ride through the countryside or along the coast.
3. **Kayaking and paddleboarding**: Paddle through the crystal-clear waters and explore the coastline.

Accommodation:
1. **Luxury resorts**: Choose from high-end hotels and resorts in popular destinations like Costa Smeralda and Alghero.
2. **B&Bs and agritourism**: Experience local hospitality at family-run B&Bs or farm stays.
3. **Self-catering apartments**: Rent a self-catering apartment or villa for a more independent stay.

Tips:
1. **Best time to visit**: June to September for warm weather, or April to May and September to October for milder temperatures.
2. **Getting around**: Rent a car or use public transportation to explore the island.
3. **Language**: Italian is the official language, but many locals speak Sardinian and English.

I hope these suggestions help you plan an unforgettable vacation in Sardinia!

Query (2):

Which cities in New Zealand offer the best adventure activities, from bungee jumping to glacier hikes?

Path (2):

—ESTABLISH SUBJECT—
===SUBJECT: TRAVEL===
—RE-WRITE QUESTION—
—ESTABLISH ROUTE—
===ROUTE: WEB SEARCH===
—WEB-SEARCH—
—PLANNER—
===CHECK COHERENCY===
===DECISION: PLANNER IS COHERENT WITH THE DOCUMENTS===
===EVALUATE PLANNER ANSWER===
===DECISION: PLANNER ADDRESSES QUESTION===

Answer (2):

Hello there, fellow travel enthusiast! I'm so excited to help you plan your next adventure in New Zealand. You're looking for some of the best cities to experience thrilling activities like bungee jumping and glacier hiking, right?

Well, let me tell you, Queenstown is often referred to as the adventure capital of the world, and for good reason! It's a paradise for thrill-seekers and outdoor enthusiasts. You'll find plenty of opportunities for bungee jumping, skydiving, white-water rafting, and more. The stunning scenery and breathtaking views make it an absolute must-visit destination.

One of the most iconic activities in Queenstown is bungee jumping off the Kawarau Bridge, which is indeed the highest commercial bungee jump in the world! You'll feel like you're flying as you take the leap of faith from 43 meters above the river. And if that's not enough, there are plenty of other adventure activities to choose from, such as hiking, mountain biking, and paragliding.

Now, I know what you're thinking: "What about Wanaka" Ah, my friend, Wanaka is a close second when it comes to adventure capital status. While Queenstown might have the quantity of activities, Wanaka has the quality and scenery on its side. You'll find plenty of opportunities for hiking, skiing, and snowboarding in the winter months, as well as stunning views of Lake Wanaka and the surrounding mountains.
If you're looking for a more unique adventure experience, Glacier Country is definitely worth considering. Heli-hiking on glaciers like Fox or Franz Josef is an unforgettable experience that will leave you breathless (pun intended!). You'll get to explore the breathtaking landscapes and witness the stunning beauty of these natural wonders up close.

Now, let's talk about itineraries! If you have 3 days in Queenstown, I recommend starting with a bungee jumping session off the Kawarau Bridge. Then, spend the next day exploring the scenic trails of the Ben Lomond Reserve or taking a hike to the top of Bob's Peak for panoramic views of the town and surrounding mountains.

For a week-long trip, here's how you could plan each day:
Day 1: Arrival in Queenstown, explore the town, and take a sunset cruise on Lake Wakatipu.
Day 2: Bungee jumping off the Kawarau Bridge, followed by a hike to the top of Bob's Peak for stunning views.
Day 3: White-water rafting or kayaking on the Shotover River, then relax with a scenic flight over Milford Sound.
Day 4: Heli-hiking on Fox Glacier, followed by a visit to the nearby Franz Josef Glacier for some glacier hiking and ice climbing.
Day 5: Explore Wanaka's scenic trails, visit the famous Wanaka Tree, and enjoy dinner at one of the town's top restaurants.
Day 6: Take a scenic flight over Mount Cook National Park, then hike to the summit of Mount Aspiring for breathtaking views.
Day 7: Return to Queenstown for some last-minute shopping or exploring before departing.

Now, let's talk about local cuisine! You have to try some of New Zealand's famous dishes like fish and chips, lamb shanks, or a traditional Kiwi breakfast with eggs, bacon, and sausages. And don't forget to indulge in some delicious craft beer or wine from the region!

Tips for travelers: Don't forget to bring your sense of adventure (and a good camera!) and be prepared for unpredictable weather conditions. Also, make sure to book your activities and accommodations well in advance, especially during peak season.

I hope you have an amazing time exploring New Zealand's adventure capital! Safe travels!

Query (3):

Who won the 2024 Olympics?

Path (3):

—ESTABLISH SUBJECT—
===SUBJECT: OTHER===

Answer (3):

Hello! I'm your travel assistant, and I can help with questions related to travel.

References

[1] Xie, J., Zhang, K., Chen, J., Zhu, T., Lou, R., Tian, Y., Xiao, Y., Su, Y.: TravelPlanner: A Benchmark for Real-World Planning with Language Agents (2024). https://doi.org/10.48550/arXiv.2402.01622

[2] LLM-Agents-Papers. GitHub (2023). https://github.com/AGI-Edgerunners/LLM-Agents-Papers.git