Company Challenge: Marbet

Marbet is a leading German event management agency specializing in innovative event strategies for renowned brands and companies. Established in 1996 and backed by the Würth Group, Marbet delivers large-scale events and business travel solutions, particularly in Germany and Spain. With around 150 professionals across offices in Germany and Barcelona, Marbet is known for its sustainable and cutting-edge approaches to event planning.

To enhance the guest experience during incentive trips, Marbet aims to develop an AI chatbot that provides instant and accurate assistance to travelers. The chatbot will answer queries about schedules, activities, travel requirements, ship services, and policies, making essential information easily accessible and improving overall satisfaction.

Your challenge is to design and build a RAG-enhanced chatbot capable of providing event assistance based on Marbet's documents. The chatbot must be developed from scratch using Python, LangChain, and Ollama, and should demonstrate the ability to retrieve and deliver event-specific information in an accurate and helpful manner.

The next sections of this document present some notes and resources related to the challenge. Please read this entire document before starting the challenge.

1. Communication Related to this Challenge

A dedicated Teams channel ("Marbet Week 9/10 Challenge") has been created for this challenge. Please use this channel for any questions, issues, or discussions related to the challenge.

The challenge organizers, Mekselina Doğanç and Edirlei Soares de Lima, will actively monitor the channel and provide assistance when needed.

We strongly encourage you to share your results and exchange ideas with other groups/students through the channel. Collaborating, discussing concepts, and sharing useful materials is highly encouraged.

2. Tools and Methods

For this challenge, you are expected to use the following tools:

- Python
- LangChain (https://github.com/langchain-ai/langchain)

- Ollama (https://ollama.com/)

An instance of Ollama is running on the ADS&AI server: http://194.171.191.226:3061

You can use the Ollama Python library (https://github.com/ollama/ollama-python) to interact with the server. For example, you can list the models available on the server by using the following code in a Jupyter notebook:

```
from ollama import Client
ollama_client = Client(host="http://194.171.191.226:3061")
ollama_client.list()
```

You can also download new models to the server:

```
ollama_client.pull("llama3.2")
```

To interact with a model, you can use the chat function:

```
response = ollama_client.chat(model="llama3.1:8b", messages=[
  {
    "role": "user",
    "content": "Why is the sky blue?",
  },
])
print(response["message"]["content"])
```

The Ollama server can also be used by LangChain:

```
from langchain_ollama import ChatOllama
llm = ChatOllama(
     base_url="http://194.171.191.226:3061",
     model="llama3.1:8b"
)
messages = [
    (
        "system",
        "You are a helpful assistant that translates English
         to Dutch. Translate the user sentence.",
    ),
    ("human", "I love programming."),
]
ai_msg = llm.invoke(messages)
ai_msg
```

Please notice that the Ollama server is only accessible when connected to the BUas network. To access it remotely, you need to connect to the VPN. You can find instructions on how to set up the VPN in the following link:

https://adsai.buas.nl/Study%20Content/Data%20Engineering/assets%2FADSAI%20Remote%20VPN.pdf

3. Understanding the Event Documents

The chatbot is expected to use Retrieval-Augmented Generation (RAG) to formulate responses based on the following provided documents:

- **Activities Overview**
  - Daily schedules of tours, excursions, and leisure activities.
  - Descriptions, locations, and special instructions for each event.
- **Packing List**
  - Checklist of required travel documents, clothing recommendations, and personal items.
- **Scenic Eclipse A-Z Guide**
  - Ship policies, onboard services, dining, entertainment, and safety regulations.
- **Tutorials and Additional Documents**
  - Tutorial on the visa application process, guest WiFi access, and SPA services.

Your first task is to analyze the provided documents and structure the data to optimize retrieval efficiency. Since the format and structure of these files may vary from one event to another, it is crucial to establish a standardized approach for data formatting and organization. More information about RAG and how to structure documents for it will be provided in the next section.

To get started, identify key categories of information (e.g., schedules, policies, service details) and organize the content accordingly. Be mindful that some documents may contain images, such as charts or scanned pages, which may require conversion to text.

Lastly, document your process for structuring and formatting the data so that it can be replicated and adapted for future events.

4. Retrieval-Augmented Generation (RAG)

To build an effective event assistant chatbot, it is crucial to understand how to use RAG to provide accurate and contextually relevant responses. RAG combines the power of large language models (LLMs) with external knowledge sources, allowing the chatbot to retrieve information from structured documents when generating answers. This approach ensures that responses are accurate, up-to-date, and aligned with the provided event documents.

Since implementing RAG-based chatbots can be complex, we have compiled a set of resources to help you get started. These resources will introduce you to key concepts, practical examples, and best practices for implementing RAG using LangChain and Ollama. You will also find tutorials on integrating RAG pipelines with Python, managing document indexing, and ensuring efficient data retrieval.

Please take the time to review these resources before starting the chatbot implementation. They will provide you with the foundational knowledge needed to complete the challenge successfully.

- Build a Chatbot: https://python.langchain.com/docs/tutorials/chatbot/
- Build a Retrieval Augmented Generation (RAG) App: Part 1: https://python.langchain.com/docs/tutorials/rag/
- Build a Retrieval Augmented Generation (RAG) App: Part 2: https://python.langchain.com/docs/tutorials/qa_chat_history/
- How to Build a Local AI Agent With Python (Ollama, LangChain & RAG): https://www.youtube.com/watch?v=E4l91XKQSgw
- Reliable, fully local RAG agents with LLaMA3.2-3b: https://www.youtube.com/watch?v=bq1Plo2RhYI
- RAG From Scratch: https://www.youtube.com/watch?v=wd7TZ4w1mSw&list=PLfaIDFEXuae2LXbO1_PKyVJiQ23ZztA0x

5. Chatbot Design & Prompt Engineering

Your chatbot must be designed to:

- Provide concise and accurate answers based on the event documents.
- Understand natural language variations in user queries.
- Offer clarification or suggest relevant topics when needed.

When designing the system prompt, try to define key chatbot attributes, such as:

- Identity: "You are an AI event assistant for an incentive trip."
- Tone: Professional, helpful, and concise.
- Knowledge Source: Answers are derived solely from provided documents.

Use prompt engineering to guide chatbot behavior, ensuring responses remain fact-based and relevant to user queries.

6. Testing & Evaluation

To test the chatbot's performance, try to develop a set of test cases to assess:

- Accuracy of responses based on real attendee questions.
- Efficiency in retrieving information from structured data.
- Handling of ambiguous or complex queries.

If certain queries fail, analyze the knowledge structure and improve document formatting or chatbot prompts accordingly. You can also explore other techniques to improve RAG, such as knowledge graphs. You can refer back to the kick-off slides for some desired query examples.

7. Final Deliverables

At the end of the project, each team must submit:

1. **Source code:** a link to your BUas GitHub repo with the source code of the chatbot.
2. **A structured report** detailing:
   - **Introduction**: Overview of objectives and chatbot functionality.
   - **Chatbot Design & Prompt Engineering**: Decisions on chatbot behavior and structuring prompts.
   - **Knowledge Base Structuring**: How documents were formatted and optimized.
   - **Testing & Results**: Key findings, examples of test cases, and refinements.
   - **Conclusion & Future Improvements**: Summary and recommendations for enhancements.

3. **A live presentation** demonstrating chatbot performance. This presentation will take place in week 10. The exact date and duration of the final presentation will be announced in the Teams channel of the challenge.

When you complete the challenge, you must submit the report and the link to your source code on the Brightspace assignment of the challenge:

Brightspace → ADSAI General Information 2024-2025 → Course Elements → Assignments → Marbet

The deliverables must be submitted individually by all students.

Good luck!