AI Developer Homework Assignment

Problem

We are a customer support automation team, and our goal is to help users get their answers before reaching an agent. We have a lot of guides on how to troubleshoot the problems users are facing, however sometimes they might be difficult to read.

Your goal is to create a chatbot that leverages Retrieval-Augmented Generation (RAG) techniques to provide relevant responses to user queries. The chatbot should be able to handle an **initial user query** and a **follow-up question** by retrieving useful information from our Help Center documentation and generating coherent responses.

** This task is given to show your understanding of LLM and RAG technologies, we will not use your solutions for our products.*

Data

You are provided with a couple of files:

- **hc_articles.json** - this file will contain NordVPN Help Center articles on troubleshooting, set-up, maintenance and other important information.
- **expected_answers.py -** this file will contain 2 example conversations and answers that we would like to get from the chatbot. Along the answers we will add the article where this information is present.

Tasks

1. Populate a database with provided Help Center articles for retrieval. Feel free to use any document preprocessing techniques you deem appropriate.
2. Integrate a Large Language Model API to generate responses.
3. Create a RAG pipeline. Ensure the RAG system is functional and able to accurately utilize the context from the database for responses.
4. Focus on creating a fully working chatbot that we can interact with. The quality of the responses should be your second priority.
5. Make sure that the system asks for the **country** and **device** of the user before troubleshooting, if the user is experiencing **connectivity issues**.
6. Evaluate how accurately the system is able to respond, you can use our provided **expected_answers.py** file. But feel free to create your own test cases too.

Documentation of design decisions and assumptions must be provided in the code itself and the README file.

Requirements

- Use Python 3 in this project.
- Make sure to provide setup instructions and additional documentation if necessary for evaluation and deployment.
- Submit your solution as a private GitHub repository and invite donatas.vaiciukevicius@cybercare.cc as a collaborator.

Recommendations

- Walk us through your thought process in the comments of your code or README.
- There should be an easy way to run your solution. README is a perfect place to provide instructions for how to run your solution.
- We recommend using **Hugging Face free LLM API** for your LLM and embedding needs, however you're free to use whatever you want.
- Consider free databases like **Chroma, Elasticsearch, Weaviate, Milvus**, etc.
- For the user interface, we recommend using **chainlit** for easy integration.
- Do not focus on the latency of the system, it is ok if the Hugging Face LLM API takes some time to give an answer. This is expected.

Evaluation Criteria

The solution is evaluated based on these criteria (in order):

1. Effective use of retrieval techniques and their integration with the chatbot.
2. Clarity and completeness of the documentation and report.
3. Quality of the code.
4. Chatbot evaluation techniques.
5. Chatbot correctness.