BOTTELCO DOCS

Made By: Julian Támara Córdoba

Date: June 07th, 2024

Purpose: AI Developer technical test

1. Introduction

The main objective of this project is to develop a bot for the telecommunications industry that can handle business processes such as order management and technical support. The bot uses generative artificial intelligence to interact with users naturally and efficiently, maintaining conversation context and operating across multiple channels (omnichannel).

2. Tech Stack

Generative AI

- Model: GPT-3.5-turbo.
    - Reasons to choose GPT-3.5-turbo: This model was selected for its advanced capability to understand and generate natural language text. It is suitable for handling complex queries and maintaining conversation context effectively.

Programming Languages

- Backend: Python
- Frontend: React

Frameworks & Libraries

- Backend:
    - FastAPI: For developing the server and handling routes.
    - Uvicorn: For running the ASGI server.
    - LangChain: For implementing text retrieval and generation chains.
    - Chroma: For implementing text retrieval and generation chains.
    - CSV: For implementing text retrieval and generation chains.
    - Dotenv: For implementing text retrieval and generation chains.
- Frontend:
    - React: For creating interactive user interfaces.
    - Axios: For creating interactive user interfaces.
    - React-router-dom: For creating interactive user interfaces.

- Storage:
    - CSV Files: For storing data on products, users, and purchases.
- Infrastructure
    - Deployment: Azure App Services
    - Infrastructure Choice for Deployment:

        For deploying our chatbot application, we selected Azure App Services due to its seamless support for both the React frontend and Python/FastAPI backend. This platform offers simplified configuration, built-in CI/CD integration, and automatic scaling, which allows us to focus on development without worrying about infrastructure management. The fully managed environment and robust monitoring tools further enhance its suitability for our needs.

    - Implementing CI/CD with Azure App Services:

        To implement CI/CD with Azure App Services, we link our source control repository (e.g., GitHub, Azure Repos) directly to the App Service. This setup allows for automatic deployments whenever changes are pushed to the main branch. We use Azure DevOps or GitHub Actions to define our CI/CD pipelines, ensuring continuous integration and deployment. This process involves building the frontend and backend, running tests, and deploying the application seamlessly, ensuring that the latest updates are always live.

3. Halucination Management::

- Information Retrieval String (RAG): Implementation of strings that combine language generation with retrieval of relevant documents.
- Historical Context: Use of a history-aware retriever to contextualize questions based on chat history.

4. General System Architecture

- The software architecture of our chatbot application is designed to ensure seamless interaction between the frontend and backend components while maintaining clear separation of concerns.

- Frontend (React): The React frontend serves as the user interface, handling all user interactions, displaying messages, and providing a responsive design for user engagement.

- Backend (Python/FastAPI): The backend, built with FastAPI, consists of multiple services that handle specific business logic:

  ChatService: Orchestrates chat interactions, ensuring context is maintained and delegating tasks to the KnowledgeService.

  KnowledgeService: Employs LangChain for advanced retrieval and generation of responses, integrating data from the product and purchase CSV files.

  UserService: Manages user authentication, registration, and user-related data.

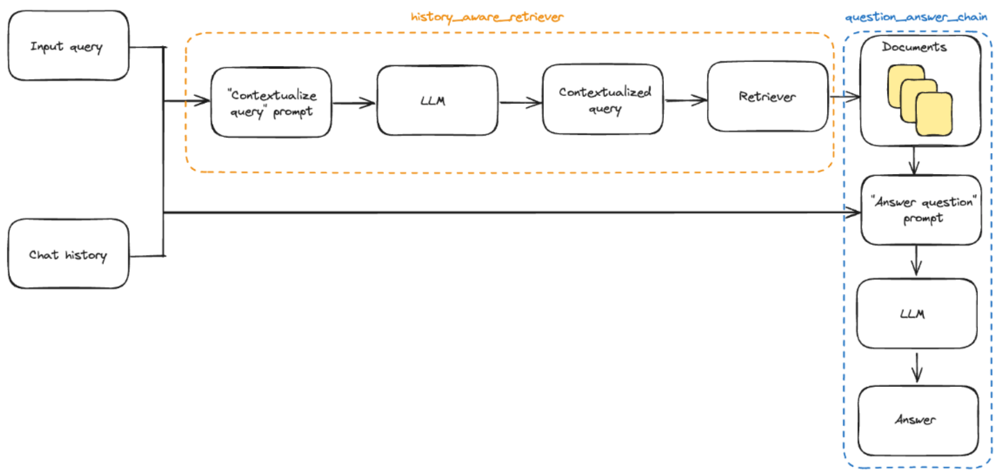
  ProductService: Handles all product-related data, including retrieving and updating product information.

  PurchaseService: Manages purchase transactions, recording them in the purchase history.

  Each service interacts with the respective models representing users, products, and purchases, ensuring data consistency and integrity.

5. AI Model Architecture

- Bot Architecture with LangChain:



- Input Query: The user's query is taken as input.

- Contextualize Query Prompt: A prompt is used to contextualize the user's query.

- LLM (Language Model): The GPT-3.5-turbo language model is used to reformulate the query if necessary.

- Contextualized query: The contextualized query is generated.

- Retriever: Relevant documents are retrieved using a vector retrieval system.

- Answer Question Prompt: A prompt is used to formulate the response based on the retrieved documents.

- LLM (Language Model): The language model generates the final response.

- Answer: The response is returned to the user.

This architecture ensures that the bot provides accurate and relevant answers by leveraging the LangChain framework for effective retrieval and generation of text. The use of GPT-3.5-turbo allows for advanced natural language understanding and generation, while the retrieval system ensures that responses are grounded in relevant and up-to-date information.

6. Omnichannel

Our telecom bot project is designed with the potential for an omnichannel experience, currently available on the web. We propose expanding to include voice via Twilio, and messaging through WhatsApp and Telegram using their respective APIs. This will allow users to interact with the bot across their preferred platforms.

To maintain conversation context across multiple channels, we will implement a centralized message routing system using LangChain's Retrieval-Augmented Generation (RAG) technique. This system will log all user interactions in a centralized database, ensuring that context is preserved and synchronized in real-time. This approach provides a seamless and consistent user experience, regardless of the communication channel.

7. Future Improvements

- Enhancing the bot's natural language understanding capabilities.

- Expanding the range of supported channels for a more comprehensive omnichannel experience.

- Implementing more advanced analytics for better user interaction insights.

8. Appendices

https://python.langchain.com/v0.2/docs/tutorials/qa_chat_history/#tying-it-together

https://github.com/juliantaco23/BotTelco