AWSage: An Agentic Chatbot

Tapaswi Satyapanthi
satyapanthi.t@northeastern.edu

1 Introduction

AWSage is an innovative chatbot developed to assist users with AWS Compute queries by leveraging a robust combination of Generative AI, Retrieval-Augmented Generation (RAG), and the LangChain ecosystem. Unlike traditional FAQ systems that require manual searching through disjointed information, AWSage automates this process by generating comprehensive responses from diverse data sources, including pre-existing FAQs and the internet.

2 Project Objectives

The primary objective of AWSage is to transform traditional FAQ systems into a dynamic, interactive chatbot that:

- Enhances user experience by providing instant, accurate responses to complex queries spanning multiple topics.
- Integrates advanced AI technologies to handle a broad spectrum of queries with high precision.
- Scales across various AWS services beyond Compute, aiming for comprehensive support coverage.

3 ChatBot Development Approach

3.1 Data Collection and Preprocessing

Data collection for AWSage involved scrapping AWS Compute FAQs directly from the official AWS website. This task was challenging due to the diverse formats across different FAQ pages. To ensure the quality and uniformity of the collected data:

- **Scraping Techniques**: Adaptive scraping techniques were implemented to accommodate different webpage layouts, extracting only relevant content (questions and answers).

- **Preprocessing**: The raw data underwent several preprocessing steps to normalize and clean the text. This included removing HTML tags, correcting encoding issues, and standardizing terminologies for consistent processing.

3.2 Embeddings

After preprocessing, the data was transformed into embeddings:

- **Jina AI's General Purpose Text Embeddings**: Utilized to convert the text data into 768-dimensional vectors. These embeddings are particularly effective in capturing the semantic meaning of the FAQs, which enhances the retrieval process.
- **Recursive Character Text Splitter**: Employed to split answers into smaller segments, preserving context and improving the model's ability to retrieve the most relevant segments based on the query.

3.3 Embeddings Vector Store

The embeddings were then stored in a Pinecone Vector Database, chosen for its efficiency and scalability in handling vector similarity searches:

- **Vector Database Capabilities**: Pinecone supports high-throughput queries and provides features like approximate nearest neighbor search, which is crucial for finding the most relevant FAQ entries in response to user queries.
- **Metadata Management**: Each embedding is stored with metadata, including the corresponding question, category, and subcategory, facilitating efficient and accurate retrieval.

3.4 LLM Model Selection

For generating responses and handling complex queries, AWSage utilizes the GPT-3.5-turbo model from OpenAI:

- **Model Capabilities**: GPT-3.5-turbo is renowned for its advanced natural language understanding and generation capabilities, making it an ideal choice for interactive applications like chatbots.
- **Tool Invocation**: This model excels in scenarios requiring the integration of various tools due to its optimized architecture for lower latency and higher throughput, which is critical in real-time applications.
- **Contextual Relevance**: The model's ability to maintain context over longer dialogues ensures that the chatbot can handle multi-turn interactions effectively, providing responses that are not only relevant but also contextually aware.

4 Use Case and Technology Overview

4.1 Agentic RAG Architecture

AWSage is structured around a sophisticated RAG architecture, employing Jina AI's embedding models and Pinecone's vector database for efficient information retrieval. This setup allows the chatbot to understand and retrieve highly relevant answers from the AWS Compute FAQs.

4.2 Data Handling and Internet Search Integration

The integration of Tavily Search API addresses out-of-scope queries such as real-time data requests, demonstrating the chatbot's ability to handle diverse user needs. This dual approach ensures that AWSage provides not only specific AWS-related information but also general knowledge as required.

4.3 Generative AI and LangChain Ecosystem

Utilizing the LangChain library, AWSage incorporates multiple AI tools such as:

- **Pinecone Vector Store**: Manages and retrieves document vectors for similarity comparisons.
- **Jina Embeddings**: Processes text data into embeddings, enhancing the chatbot's understanding of user queries.
- **LangGraph**: Orchestrates the flow of data and query handling, providing a seamless integration of various AI tools.

5 Key Features and Functionalities

- **Dynamic FAQ Integration**: Converts static FAQ data into a responsive chat interface.
- **Hybrid Query Handling**: Uses both embedded knowledge and live internet search to respond to queries.
- **Smart Routing**: Directs queries to the most relevant tool based on their nature, optimizing response accuracy and speed.

6 Challenges and Solutions

- **Diverse Data Formats**: Implemented adaptive scraping techniques to handle varying layouts across AWS FAQ pages.
- **Computational Demands**: Utilized cloud resources to manage the high computational needs of LLMs and data embedding processes.

7 Challenges and Solutions

- **Diverse Data Formats**: Implemented adaptive scraping techniques to handle varying layouts across AWS FAQ pages.
- **Computational Demands**: Utilized cloud resources to manage the high computational needs of LLMs and data embedding processes.

8 Fine-Tuning

For fine-tuning, we have leveraged OpenAI's `gpt-4o-mini-2024-07-18` model, which was specifically trained on the data scraped from the AWS website.

8.1 Steps to Execute Fine-Tuning Job on the Scraped Data

IMPORTANT: Ensure to generate an OpenAI API key with all the necessary privileges for fine-tuning. Lack of privileges will result in errors during the execution of the following steps:

1. **Convert the Scraped Data**: Execute `fine_tuning/create_data.py` to generate the data in a format suitable for fine-tuning as specified by OpenAI.
2. **Execute the Fine-Tuning Job**: Run all the cells in `fine_tuning/fine_tuning.ipynb`. This initiates a fine-tuning job which can be monitored via OpenAI's dashboard.
3. **Post-Fine-Tuning Configuration**: Once the fine-tuning process is complete, set `USE_FINE_TUNED_MODEL` to `TRUE` in the `.env` file to enable the chatbot to use the fine-tuned model.

NOTE: The fine-tuning process may require a significant amount of time, depending on the complexity and size of the data.

9 Conclusion

AWSage represents a significant advancement in chatbot technology, successfully integrating multiple AI disciplines to enhance user interaction with AWS services. Its ability to simplify complex data retrieval and present information in a user-friendly manner marks a pivotal shift from traditional FAQ systems to more dynamic, intelligent platforms.

10 Future Scope

- **Service Expansion**: Plans to include more AWS services, broadening the chatbot's expertise.
- **FAQs-to-Chatbot Framework**: Development of a generalized framework that allows users to transform any set of FAQs into a functional chatbot, promoting adaptability and user engagement across various domains.