Adaptive Recommendation Chatbot with RAG and Vector Database

Developed a Chatbot for Therapists using Vector Embeddings and Pinecone

Introduction

The goal of this project was to implement a chatbot designed to lighten therapists' administrative load so they can spend more quality time with patients. It integrates into a therapist's workflow and simplifies pulling up need-to-know details from previous sessions through natural questions by storing therapist-patient conversations as vector embeddings in Pinecone for efficient retrieval. The chatbot leverages OpenAI's text-embedding-ada-002 model for embedding generation and utilizes Retrieval-Augmented Generation (RAG) for generating contextually accurate responses based on user queries.This report details the approach taken, challenges faced, and the solutions implemented.

Approach

1. Data Collection and Preprocessing:

- I started by collecting therapist-patient conversations from a CSV file.
- The text data was cleaned to remove any missing values and irrelevant information.
- Example data:

  Client_ID,Age, Gender, Transcript

  1, John Doe, 35, Male,"Therapist: How are you feeling today? Patient: I'm feeling anxious."

  2, Jaxonina Peter, 25, Female, "Therapist: Can you tell me more about that? Patient: I've been having trouble sleeping."

2. Generating Embeddings:

- We used the ‘text-embedding-ada-002’ model to convert each conversation into a vector representation.
- This model from OpenAI provides high-quality embeddings suitable for semantic similarity tasks.

3. Storing Embeddings in Pinecone:

- We initialized Pinecone, a scalable vector database, using an API key.
- The conversation embeddings were stored in Pinecone, allowing for efficient retrieval based on similarity.
- We created a mapping between conversation IDs and their text, saving this mapping to a JSON file for later use.

4. Querying the Vector Database:

- A function was implemented to convert user queries into vectors and perform a similarity search in Pinecone.
- The top relevant conversations were retrieved based on the similarity score.
- OpenAI's language model was used to generate responses by combining the retrieved conversation snippets with the user query.

- A Streamlit web application was set up to provide an interactive user interface for therapists.

Challenges Faced and Solutions

1. Data Cleaning and Preprocessing:
- **Challenge**: Ensuring data quality and consistency was crucial for generating accurate embeddings.
- **Solution**: Implemented deep data cleaning processes to handle missing values and irrelevant information.

2. Choosing the Right Embedding Model:
- **Challenge**: Selecting an embedding model that balances performance and efficiency.
- **Solution**: After evaluating several models, I chose ‘text-embedding-ada-002’ for its high-quality embeddings.

3. Scalability of Storage:
- **Challenge**: Efficiently storing and retrieving large amounts of vector data.
- **Solution**: Pinecone was chosen for its scalability and speed, making it suitable for handling high-dimensional vector data.

4. Generating Contextually Accurate Responses:
- **Challenge**: Ensuring the chatbot provides contextually relevant and accurate responses.
- **Solution**: By combining similarity search results with OpenAI's language model, I achieved high-quality responses tailored to user queries.

5. User Interface Design:
- **Challenge**: Creating an intuitive and easy-to-use interface for therapists.
- **Solution**: Streamlit was used to build a simple and interactive web application, allowing therapists to input queries and receive responses seamlessly.

Conclusion

This project successfully developed a chatbot that helps therapists query patient-related conversations efficiently. By leveraging vector embeddings and a vector database, the chatbot provides contextually accurate and relevant information quickly. The challenges faced were addressed with appropriate solutions, ensuring the robustness and scalability of the system.

Future Work
- **Expand Dataset**: Incorporate more diverse conversation data to improve the chatbot's knowledge base.
- **Refine Embedding Models**: Explore and integrate more advanced embedding models for even better performance.

- **Enhance User Interface**: Continuously improve the UI based on user feedback to make it more user-friendly and intuitive.

Acknowledgments

I would like to thank the developers of OpenAI, Pinecone, and Streamlit for their powerful tools and libraries that made this project possible.

References

- OpenAI: https://www.openai.com/
- Pinecone: https://www.pinecone.io/
- Streamlit: https://www.streamlit.io/