COMP-SCI 5588
Data Science Capstone
Week5 Handson

Professor: Dr Yugyung Lee
Term name: Bug Killers
Feb 21, 2025

Content

Links

GitHub：https://github.com/nanxuanhui/DSCapstone.git
Demo:https://drive.google.com/drive/folders/1RG_ps9vVFRKLJbTQaOvPFUdxfO3pnVbS?usp=sharing

Individual Contributions

1. **Team Members**

Name	Contributions
Saniya Pandita	Frontend Developer: Developed the user interface using Streamlit, ensuring an intuitive and user-friendly design.
Jayadithya Nalajala	Project Lead: Responsible for overseeing the project, coordinating tasks, and integrating various components.
Sai Jahnavi Devabhakthuni	Backend Developer: Implemented FastAPI for the backend, handled API endpoints, and optimized performance.
Hui Jin	NLP Engineer: Integrated Whisper for speech recognition and fine-tuned LangChain models for improved responses.

Techniques Implemented and Impact

1. **Speech and Audio Processing with Whisper**
   - Used OpenAI's Whisper model for speech-to-text transcription.
   - Summarized transcribed text using Hugging Face's philschmid/bart-large-cnn-samsum model.
   - Functionality: Transcribes audio files (e.g., doctor-patient conversations) and generates concise summaries for caregivers.
2. **Real-Time AI Deployment with FastAPI & Streamlit**
   - Built a FastAPI backend to serve the Whisper and summarization models.
   - Created a Streamlit frontend for uploading audio files and viewing transcriptions and summaries.
   - Functionality: Provides a user-friendly interface for caregivers to interact with the system.

3. **AutoGPT and AI Agents Using LangChain**
    - Implemented a LangChain-based AI agent for conversational assistance.
    - Integrated SerpAPI for web search to fetch real-time information (e.g., medication details, news).
    - Functionality: Acts as a virtual assistant for elderly users, answering questions and providing reminders.

Functionalities Implemented

1. **Speech-to-Text Transcription**
    - Used OpenAI Whisper model to transcribe voice commands into text.
    - Functionality: Allows users to speak naturally and have their commands converted into text for processing.
2. **Real-Time Information Retrieval**
    - Integrated SerpAPI to fetch real-time information (e.g., news, weather) based on user queries.
    - Functionality: Users can ask, "What's the news today?" and receive up-to-date information.
3. **Conversational AI**
    - Implemented a LangChain-based AI agent to handle conversational queries.
    - Functionality: Acts as a virtual assistant, answering questions and providing reminders.
4. **Summarization**
    - Used Hugging Face's philschmid/bart-large-cnn-samsum model to summarize long responses (e.g., news articles).
    - Functionality: Provides concise summaries for easier understanding.
5. **User Interface**
    - Built a Streamlit frontend for uploading audio files and viewing AI-generated responses.
    - Functionality: Provides a simple and intuitive interface for caregivers and users.

Results and Findings

1. **Speech and Audio Processing**
    - Whisper accurately transcribed audio files, even with background noise.
    - Summarization provided concise bullet points, making it easier for caregivers to review key information.
2. **Real-Time Deployment**
    - FastAPI successfully served the models, and Streamlit provided an intuitive interface.

- Users could upload audio files and receive transcriptions and summaries in real time.

3. AI Agents

- The LangChain-based agent effectively answered questions and fetched external data.
- However, we faced challenges with OpenAI API rate limits and switched to Hugging Face's open-source models.

Challenges

1. OpenAI API Rate Limits

- Our OpenAI API credits were exhausted during testing, limiting our ability to use GPT-based models.
- Solution: Switched to Hugging Face's open-source models for summarization and conversational AI.

2. Gemini API Rate Limits

- Attempted to use the Gemini API as an alternative but encountered rate limits.
- Solution: Continued with Hugging Face models, which provided reliable performance.

3. Audio File Upload Issues

- Initially faced 400 Bad Request errors when uploading audio files to the FastAPI backend.
- Solution: Debugged the FastAPI endpoint to ensure proper file handling and validation.

4. Model Latency

- The Whisper model had high latency for large audio files.
- Solution: Optimized the model by truncating long audio files and processing them in chunks.

Future Improvements

- Enhancing Multilingual Support – Implement additional language models to support a broader range of users.
- Improved Context Awareness – Integrate memory-based dialogue management to retain context over multiple interactions.
- Mobile Application Integration – Develop a mobile-friendly version using React Native or Flutter.
- Cloud Deployment – Migrate to AWS/GCP for better scalability and accessibility.
- Advanced Personalization – Implement user profiling to tailor responses based on individual preferences and past interactions.

- Security Enhancements – Introduce authentication mechanisms to ensure secure access and prevent API misuse.