AI Salesman System Documentation

Overview

The AI Salesman system is designed to act as a cold-calling agent capable of holding meaningful conversations, understanding customer queries, and closing sales. The focus is on creating a high-performance, low-latency solution that delivers natural-sounding voice interactions, simulating human-like conversation.

The web-based demonstration showcases the system's functionalities without telephony integration. The solution uses advanced AI models for STT (Speech-to-Text), TTS (Text-to-Speech), and conversation handling with a robust backend to manage data and interactions.

Architecture

Workflow:

1. **Customer Call Initiation**: The AI Salesman begins a simulated call or receives one via the web interface.
2. **Speech-to-Text (STT)**: Real-time transcription of customer speech using Whisper Large V3 Turbo.
3. **LLM (Large Language Model)**: LLaMA 3.3 70B processes the transcribed text, decodes customer queries, and retrieves relevant data.
4. **Context Management**: Tracks and stores the conversation context for personalized interactions and smooth conversation flow.
5. **Data Retrieval**: Queries the Pinecone database for customer information and related details.
6. **Chatbot Response**: Generates responses to customer queries.
7. **Text-to-Speech (TTS)**: Converts the chatbot's response to speech using F5 TTS (running locally).
8. **Call Summary**: At the end of the call, a summary is generated and presented to the operator.

Technologies Used:

- **LLM**: LLaMA 3.3 70B
- **STT**: Whisper Large V3 Turbo
- **TTS**: F5 TTS (local deployment)
- **Backend**: FastAPI
- **Database**: Pinecone
- **Web Demo**: HTML, CSS, and JavaScript frontend

Setup Instructions

Prerequisites

1. Python 3.10+
2. Packages in requirements.txt
3. Docker (optional for containerized deployment)
4. F5 TTS installed locally
5. API keys for required services:
    - PINECONE_API
    - LLAMA_API_KEY
    - HUGGING_FACE_API
    - SAMBANOVA_API_KEY

Installation Steps:

Backend:

1. Clone the repository.

2. Install dependencies:
```
pip install -r requirements.txt
```

3. Set up environment variables in a `.env` file:
```
PINECONE_API=your_pinecone_api_key
LLAMA_API_KEY=your_llama_api_key
HUGGING_FACE_API=your_hugging_face_api_key
SAMBANOVA_API_KEY=your_sambanova_api_key
```

4. Start the backend server:
```
uvicorn web.main:app --host 0.0.0.0 -port $PORT
```

Frontend:

1. Navigate to the `web` directory.
2. No additional dependencies required for HTML, CSS, and JavaScript.
3. Run the `uvicorn` command then go to the browser to run the web demo.

F5 TTS:

1. Ensure F5 TTS is set up locally and running.
2. Configure the backend to communicate with the local TTS instance.

File-Specific Details

`main.py`

- **Purpose**: Serves as the entry point for the backend system.
- **Key Functions**:
  - Defines API routes for handling transcription requests (`/ws/speech-to-text`), chatbot interaction (`/ws/generate-response`), and TTS conversion.
  - Implements the FastAPI framework to ensure asynchronous and scalable request handling.
- **Implementation Details**:
  - Uses `uvicorn` for running the server.
  - Routes link directly to services like `stt_service.py` and `tts_service.py`.

`stt_service.py`

- **Purpose**: Processes real-time audio input to transcribe customer speech.
- **Key Functions**:
  - Utilizes Whisper Large V3 Turbo for accurate transcription.
  - Handles audio stream preprocessing and manages API calls to the Whisper model.
- **Implementation Details**:
  - Ensures low-latency transcription for real-time interactions.
  - Converts audio input into text format compatible with the chatbot.

`chatbot_service.py`

- **Purpose**: Handles natural language processing and response generation.
- **Key Functions**:
  - Integrates LLaMA 3.3 70b to generate contextually accurate and conversationally relevant responses.
  - Maintains conversation history and context for a seamless user experience.

- **Implementation Details**:
    - Supports multi-turn conversations.
    - Manages errors like unclear inputs with fallback responses.

`database_service.py`

- **Purpose**: Interfaces with Pinecone to store and retrieve customer data and conversation logs.
- **Key Functions**:
    - Manages vectorized data storage for efficient querying.
    - Retrieves context-specific information to personalize responses.
- **Implementation Details**:
    - Utilizes Pinecone's API for high-speed data indexing and lookup.
    - Ensures secure and optimized database operations.

`tts_service.py`

- **Purpose**: Converts text responses from the chatbot into natural-sounding audio.
- **Key Functions**:
    - Leverages the locally hosted F5 TTS engine for speech synthesis.
    - Processes chatbot output and generates audio streams for playback.
- **Implementation Details**:
    - Configurable to adjust voice pitch, speed, and tone.
    - Communicates with the backend through defined API routes.

`frontend/`

- **Purpose**: Provides the user interface for the web-based demonstration.
- **Components**:
    - **HTML**: Structures the layout for the web interface.
    - **CSS**: Designs the visual style and user-friendly aesthetics.
    - **JavaScript**: Implements interactive features like initiating calls, displaying transcripts, and audio playback.
- **Implementation Details**:
    - Ensures cross-browser compatibility.
    - Uses asynchronous requests to interact with the backend endpoints seamlessly.

Features

1. Real-time transcription of customer speech.
2. Intelligent query handling with a state-of-the-art LLM.

3. Context-aware conversations for personalized interaction.
4. Natural-sounding TTS output.
5. Automated call summary generation.

API References

STT Endpoint

- **Route**: `/ws/speech-to-text`
- **Method**: POST
- **Input**: Audio stream
- **Output**: Transcribed text

LLM Endpoint

- **Route**: `/ws/generate-response`
- **Method**: POST
- **Input**: Transcribed text, conversation context
- **Output**: LLM response text

TTS Endpoint

- **Route**: `/ws/generate-response`
- **Method**: POST
- **Input**: Chatbot response text
- **Output**: Audio stream

Demonstration

Visit the web-based demo at AI Salesman Demo to experience the system in action.

Future Enhancements

1. Integration with telephony systems for real-world deployment.
2. Advanced sentiment analysis for better customer interaction.
3. Additional language support for broader applicability.

Contributors

- Devesh Bhushan, Devdeep Mukherjee, Dishita Pokharna, Bhumika

---

For further questions, please contact the development team.