Asistente de Atención Telefónica Automatizado

Este proyecto implementa un sistema completo cliente-servidor para simular una llamada telefónica automatizada utilizando modelos de voz, lenguaje natural y síntesis de texto a voz, completamente offline (sin depender de servicios externos).

Arquitectura del Sistema

La solución se basa en dos componentes principales:

1. Cliente (cliente.py)

- Graba audio del usuario localmente (4 segundos por turno).
- Envía el audio al servidor por WebSocket.
- Reproduce la respuesta de audio recibida del servidor.

2. Servidor (servidor2.py)

- Despliega un servicio WebSocket usando FastAPI.
- Convierte audio a texto con Whisper (STT).
- Procesa el texto con LLaMA 2 (7B) para generar una respuesta conversacional.
- Convierte texto a audio con TTS (Tacotron2-DDC).
- Envía el audio de vuelta al cliente.
- Gestiona contexto por llamada para mantener el flujo conversacional.

Modelos Locales Utilizados

Funcionalidad	Modelo	Fuente / Framework
STT	whisper-small	OpenAI Whisper
LLM	meta-llama/Llama-2-7b-chat-hf	HuggingFace Transformers
TTS	tts_models/en/ljspeech/tacotron2-DDC	Coqui TTS

Todos los modelos son cargados localmente, evitando llamadas a servicios en la nube.

Flujo y Gestión de Contenido

Flujo de una llamada:

- El servidor envía un mensaje de bienvenida sintetizado.
- El cliente graba un turno de usuario (4s), lo envía al servidor.

- El servidor transcribe, genera texto de respuesta (usando contexto), sintetiza audio y lo retorna.
- El cliente reproduce la respuesta.
- El ciclo se repite hasta que se cierre la conexión.

Gestión de contexto:
- Se mantiene un historial breve (history) por llamada (call_id), limitado a los últimos 2 intercambios.
- Permite respuestas coherentes y evita repeticiones.
- Se incluye detección de temas repetidos y manejo con una respuesta predeterminada.

Respuestas predefinidas:
- Algunas preguntas frecuentes como "where are you located" tienen respuestas instantáneas sin pasar por el modelo LLM.
- Mejora el rendimiento y reduce la carga computacional.

Ejecución

Servidor:

```
python servidor2.py
```

Cliente:

```
python cliente.py
```

Nota: Asegúrate de tener tu micrófono y altavoces habilitados. Ambos scripts deben correr en la misma máquina o estar conectados por red.

Requisitos

Python 3.8+

Dependencias:
- fastapi, websockets, uvicorn
- whisper, transformers, torch
- TTS, soundfile, librosa, numpy, sounddevice