AI Audio Transcription and Grading Tool

Azhar Abdool

This tool allows users to upload audio files for transcription and receive a graded analysis of the transcription's fluency, lexical diversity, and grammar using a Whisper speech-to-text model and a fine-tuned BERT model.

Setup Instructions

1. Local Installation

Prerequisites

- Python 3.8 or higher
- Pip package manager
- FFmpeg (required for processing audio files)

Steps

1. Clone the repository:

- https://github.com/azharabdool/Personal1.git

2. Install dependencies:

-pip install -r (requirements)

3. Download and set up the Whisper model:

-pip install openai-whisper

4. Download and set up the BERT model:

-pip install transformers torch

5. Ensure FFmpeg is installed:

-sudo apt update

-sudo apt install ffmpeg

6. Start the Flask app:

-python app.py

7. Access the app in your browser:

-http://127.0.0.1:5000

Source Code Overview

Components

1. **Whisper Model Integration**:
    - The OpenAI Whisper model transcribes the audio files.
    - Handles transcription errors gracefully.
2. **BERT Model Integration**:
    - Tokenizes the transcription.
    - Uses fine-tuned BERT for sequence classification to determine fluency.
3. **Scoring System**:
    - **Fluency**: Based on sentiment analysis.
    - **Lexical Diversity**: Ratio of unique words to total words.
    - **Grammar**: Placeholder fixed score.
4. **Flask Application**:
    - Routes for file upload and processing.
    - HTML templates for user interaction.

Key Files

- app.py: Main application file.
- templates/index.html: Frontend template for file uploads and displaying results.
- uploads/: Folder to store uploaded audio files.

APIs Used

- **Transformers** (HuggingFace): For BERT-based text analysis.
- **Whisper**: For transcription of audio files.
- **Pydub (mediainfo)**: For extracting audio metadata.

Challenges and Solutions

My main challenge which affected the whole direction is keeping my API keys secure, and so to be safe I had to disregard my first version of the application using openAI and used a different route to grade.

Challenge 1: Audio Processing

- **Issue**: Ensuring uploaded audio files were in the correct format (mono, 16-bit PCM, 16 kHz).
- **Solution**: Used Pydub and FFmpeg to validate and preprocess audio files.

Challenge 2: Model Integration

- **Issue**: Large model sizes and slow processing times.
- **Solution**: Used smaller versions of Whisper (base) and BERT for faster processing while maintaining reasonable accuracy.

Challenge 3: Grading System

- **Issue**: Creating an accurate scoring system.
- **Solution**: Simplified initial metrics and allowed for future extensibility.