An Audio Pipeline for Sandalwood Cultivation Resource Processing

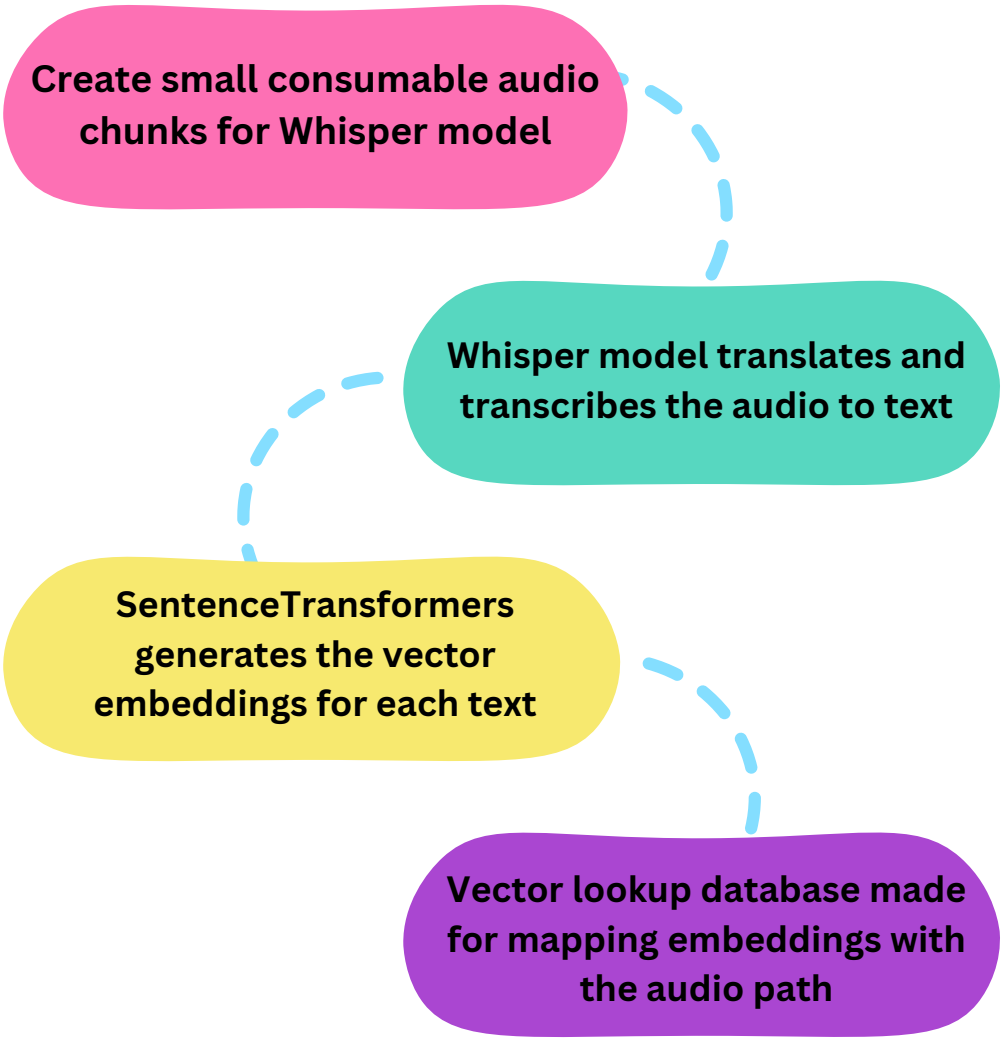
~ Shashank Shekhar Singh,

Mechanical Engineering, IIT BHU

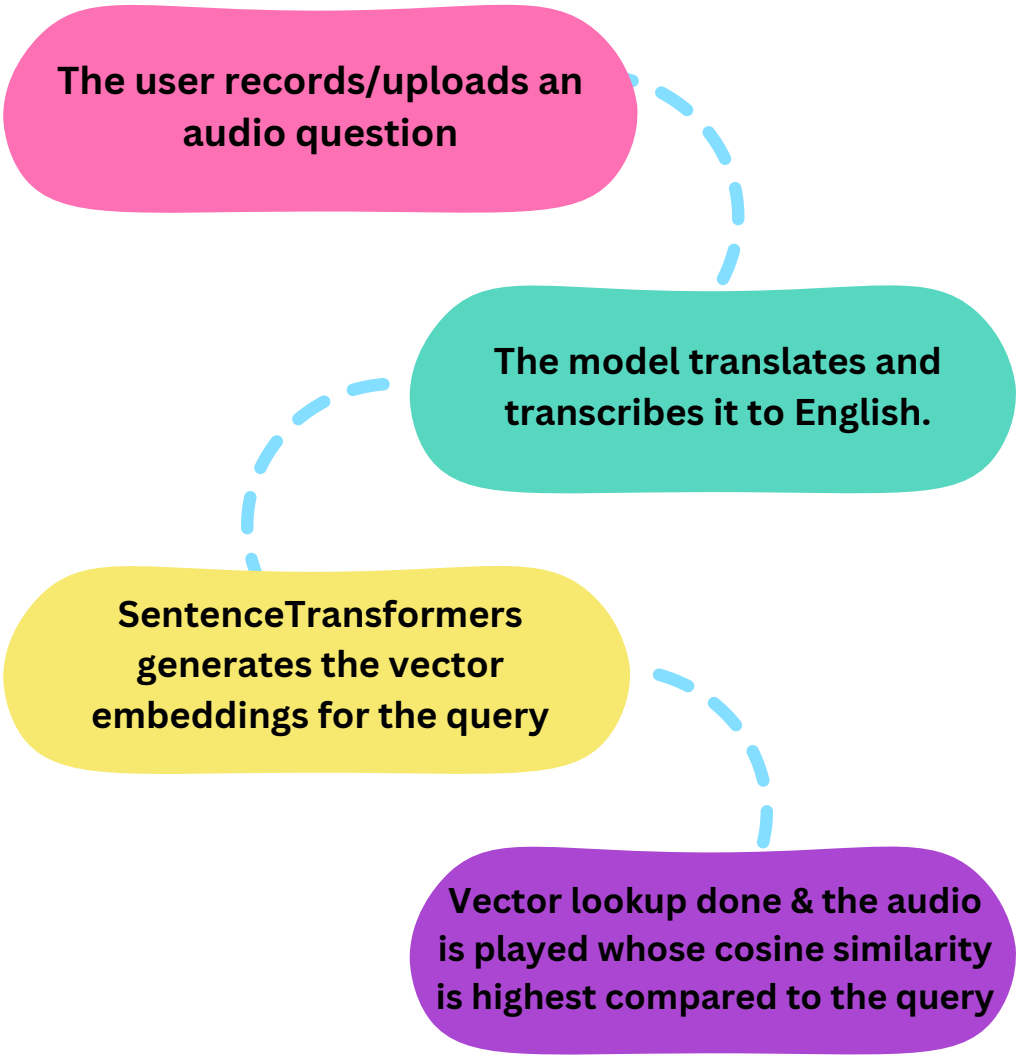
1. Introduction

Sandalwood has been integral to Indian culture, particularly in Karnataka, where it holds economic, religious, and medicinal value. Given its importance, it is critical to capture indigenous knowledge on sandalwood cultivation and conservation methods. This project aims to build a **speech-based pipeline for processing audio resources** on sandalwood cultivation, making them **accessible and queryable for users.**

The Vector Database Design



The Audio Lookup and Playback Design



2. Problem Description

The task involves:

1. Developing an Automatic Speech Recognition (ASR) model for Kannada audio resources, which are primarily in a colloquial style.
2. Implementing a speech-based Question-Answering (QA) system to enable users to ask questions and receive relevant audio segments containing the answers.

3. Dataset Description

The dataset comprises audio recordings in Kannada about sandalwood cultivation, scraped from YouTube, and supplied by the ML Fiesta Organizing Team. The colloquial nature of the recordings presents unique challenges for ASR due to informal vocabulary, idiomatic expressions, and background noise.

4. Task 1: Automatic Speech Recognition Model

4.1 Model Choice

To build the ASR model, I used **OpenAI's Whisper model (large version)** due to its multilingual support and capability to handle noisy environments. Whisper allows direct translation and transcription of audio from Kannada to English, facilitating further processing.

4.2 Model Training and Fine-tuning

Whisper model has a time limit of 30 seconds to the length of audio that can be fed in a single prompt. In order to tackle this, I divided the audio into chunks of 30 seconds each. Whisper's pretrained model was fine-tuned and tested on the Sandalwood dataset to better handle the nuances of Kannada's colloquial speech. I compared it with the other available closed source models and it was performing quite impressive.

4.3 Implementation

I used Whisper's transcribe() function to generate transcriptions for the audio files in English, because I was new to Kannada language. These transcriptions were stored along with the original audio paths, forming a database for subsequent search operations.

5. Task 2: Speech-Based Question-Answering System

5.1 Pipeline Overview

The pipeline leverages ASR to convert both the user's question and the dataset audio transcriptions into text, facilitating a text-based search to locate relevant audio segments.

5.2 Model Choice for Sentence Embedding

To find the most relevant response to a query, we used the **SentenceTransformer model ("all-MiniLM-L6-v2")** to generate embeddings for the audio chunka dataset transcriptions. Whenever we get a audio question from the user, it was passed through the Whisper Mode; for its text transcription. After this, the text user queries were passed through the SentenceTransformer model for getting the vector embeddings. This model is lightweight and effective for similarity matching.

5.3 Similarity Matching and Retrieval

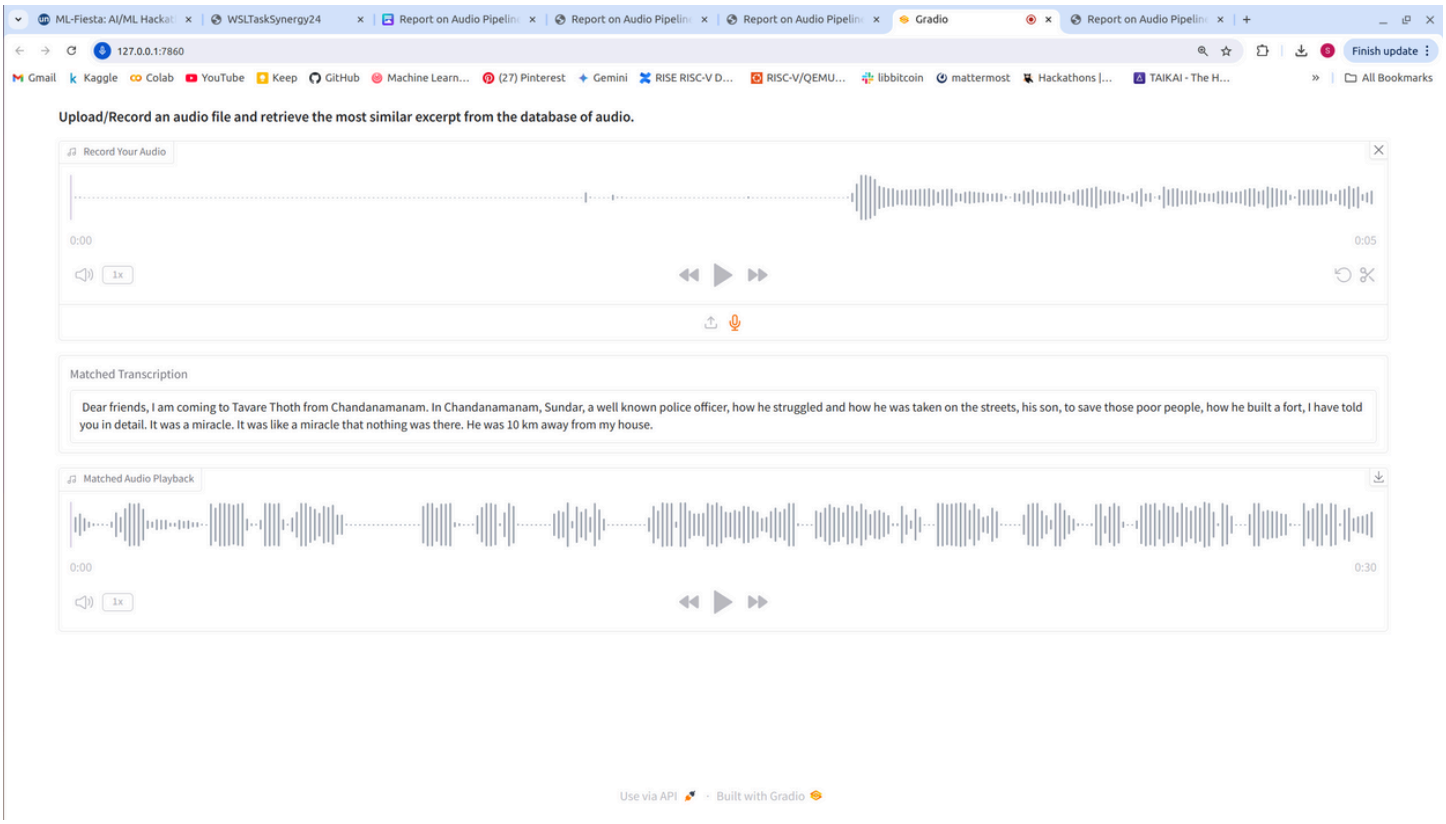
The embedding for the user's query is matched against precomputed embeddings of the audio transcriptions using cosine similarity. The segment with the highest similarity score is identified, and both its transcription and corresponding audio file path are returned to the user, on the screen.

6. Code Implementation

Below is an outline of the primary components in the code:

1. **Audio Input and Processing**:
   - The user provides input through an audio recording, which is temporarily saved and then processed for transcription.
2. **Transcription and Embedding**:
   - The ASR model transcribes the audio into text, which is then converted into an embedding for similarity matching.
3. **Gradio Interface**:
   - The Gradio library is used to create a user-friendly interface, allowing users to input audio, view transcriptions, and listen to matched audio segments.

7. Testing and Results



The system successfully transcribes and matches queries to relevant audio segments, handling the informal Kannada language effectively. While the performance is strong, future improvements may include:

- Expanding the dataset to cover more dialects and noise variations.
- Utilizing advanced language models to enhance context comprehension in QA tasks.

8. Future Work

- Implement additional noise reduction techniques to enhance transcription accuracy.
- Integrate dialect-specific fine-tuning for broader Kannada language support.