Project Report

On

Multilingual Speech Recognition Model for RAG without Training

Submitted by

Kota Prathyusha

Email:

Prathyureddykota05@gmail.com

Phone No:

8790402811

CERTIFICATE

This is to certify that, KOTA PRATHYUSHA has successfully completed the Project Report on "**Multilingual Speech Recognition Model for RAG without Training**" for partial fulfilment of the project co-ordinator.

Date:/04/24. **Project Co-Ordinator**

INDEX

1. ABSTRACT

To build a multilingual speech recognition model without training, using a pre-trained multilingual speech recognition model, such as Multilingual Whisper, to enable RAG to perform tasks in multiple languages. Generate textual descriptions based on detected language from audio.

2. INTRODUCTION

Nowadays we always hear about new technology that improves our lifestyle, that makes our life easier. Technology has revolutionized the human mankind. Human race has put a gear in technology and they are not in a mood to move the pedals away from this gear. There is huge research on various technology sector such as Artificial Intelligence.

In the ever-evolving landscape of technological innovation, few advancements have captured our imagination and transformed our daily lives quite like speech recognition. From virtual assistants that respond to our every command to transcribing our spoken words into written text with uncanny accuracy, speech recognition technology has revolutionized human-machine interaction and fundamentally changed the way we communicate and interact with digital devices.

At the forefront of this technological revolution lies a relentless pursuit of enhancing the capabilities and efficiency of speech recognition systems. Researchers and engineers around the globe are tirelessly pushing the boundaries of what's possible, leveraging state-of-the-art deep learning models and sophisticated algorithms to unlock new levels of accuracy, speed, and adaptability.

One such ground breaking advancement in the realm of speech recognition is the development of cutting-edge models like the Whisper Large v3 by OpenAI. Built upon the foundation of deep learning and leveraging the latest advancements in natural language processing, the Whisper Large v3 model represents a quantum leap forward in the field of Multilingual Speech Recognition Model for RAG without Training.

With its unparalleled ability to understand and transcribe speech in multiple languages with exceptional accuracy and efficiency, the Whisper Large v3 model stands as a testament to the power of modern AI technologies in transforming the way we interact with and harness the potential of spoken language.

In this project, we embark on a journey to explore and harness the capabilities of the Whisper Large v3 model for speech recognition. Our goal is to leverage this state-of-the-art technology to develop a robust and versatile speech recognition system that caters to the diverse needs and requirements of users across various domains and industries.

Through a combination of rigorous research, experimentation, and real-world testing, we aim to showcase the immense potential of the Whisper Large v3 model in facilitating seamless communication, breaking down linguistic barriers, and empowering individuals to interact with digital systems in more intuitive and natural ways.

Our goal is to make communication easier for everyone, no matter what language they speak. With our system, you can talk to your devices more naturally and get things done faster.

4.METHODOLOGY

4.1 System requirements

4.1.1Hardware Requirements:

- **CPU:** Intel Core i5 with multiple cores and high clock speeds.
- **GPU:** NVIDIA Tesla V100 with a minimum of 8 GB VRAM (preferablymore).
- **RAM:** 8 GB or more DDR4 RAM.
- **Storage:** SSD with a minimum of 500 GB
- **Mouse and Keyboard:** Any standard mouse and keyboard should suffice.

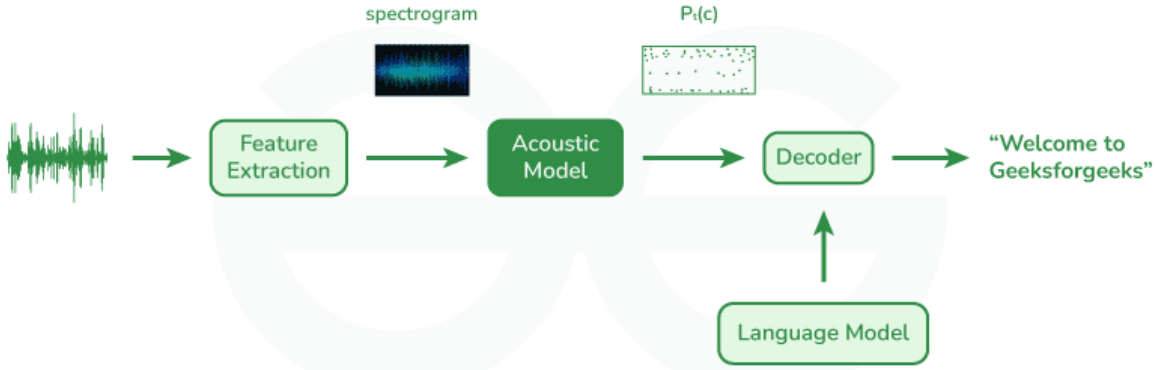
4.1.2Software Requirements:

- Operating System: Windows 10
- Coding Language: Python 3.7

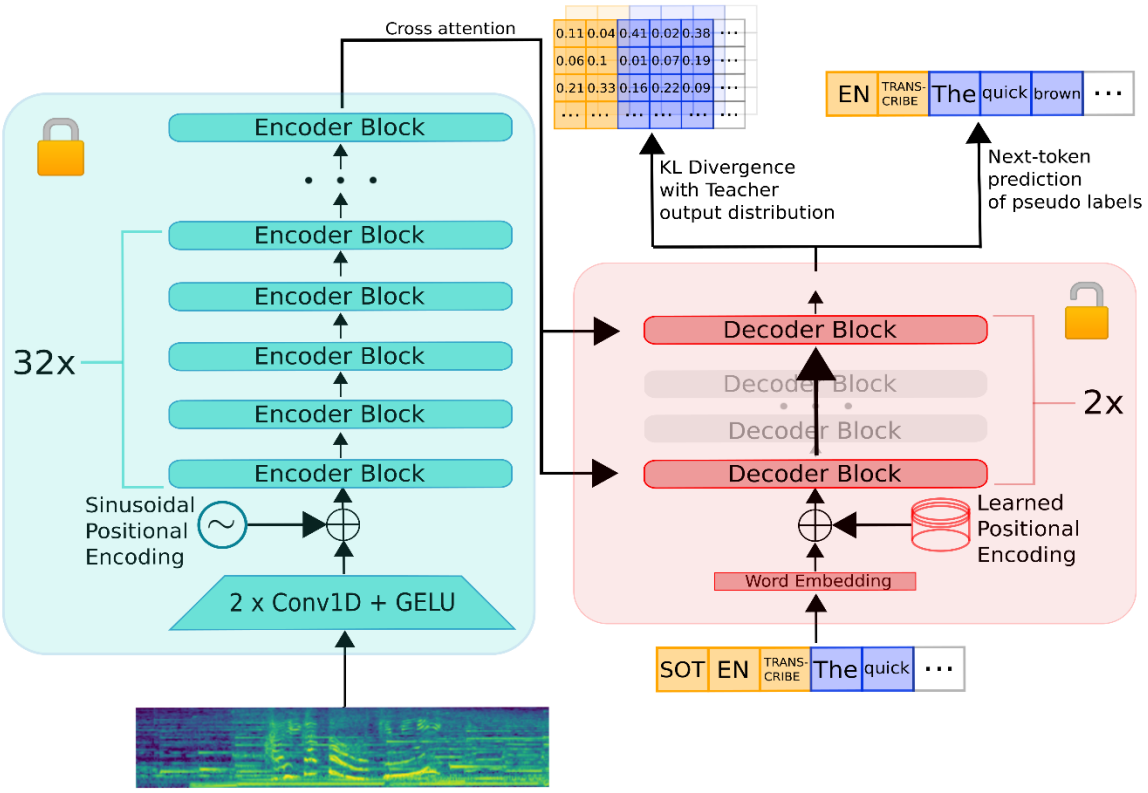
5. SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE

Multilingual speech recognition architecture:



Whisper large v3 architecture:



6. SOFTWARE IMPLEMENTATION

Step 1: Setup Environment

1. Install necessary libraries and dependencies, including transformers, datasets, and any other required packages.
2. Set up a development environment using Python and preferred IDE or code editor.

Step 2: Data Collection and Pre-processing

1. Identify and collect speech data in multiple languages for training and testing the model.
2. Preprocess the speech data, including noise removal, normalization, and feature extraction.
3. Prepare text transcripts corresponding to the speech data.

Step 3: Model Training

1. Load the Whisper Large v3 model using the transformers library.
2. Fine-tune the model on the collected multilingual speech dataset.
3. Train the model to map speech inputs to corresponding text outputs.

Step 4: Model Evaluation

1. Evaluate the trained model's performance on a separate test dataset.
2. Use metrics such as word error rate (WER) or accuracy to measure the model's accuracy and efficiency.

Step 5: Integration with Speech Recognition System

1. Develop a user-friendly interface for interacting with the speech recognition system.
2. Implement functionality to capture live speech input from a microphone or process pre-recorded audio files.
3. Integrate the trained Whisper Large v3 model into the system to perform speech-to-text conversion.

Step 6: Deployment

1. Deploy the software implementation on a suitable platform, such as a local machine or cloud server.
2. Ensure scalability, reliability, and security of the deployed system.
3. Provide documentation and instructions for users to access and utilize the speech recognition system.

Step 7: Testing and Validation

1. Conduct comprehensive testing to ensure the software implementation functions as expected.
2. Validate the accuracy and performance of the speech recognition system across different languages and use cases.
3. Address any bugs or issues identified during testing and validation.

Step 8: Maintenance and Updates

1. Monitor the performance of the deployed system and collect user feedback.
2. Implement necessary updates, improvements, and bug fixes based on user feedback and emerging requirements.
3. Maintain documentation and provide ongoing support to users as need

7. OUTPUT SCREENS

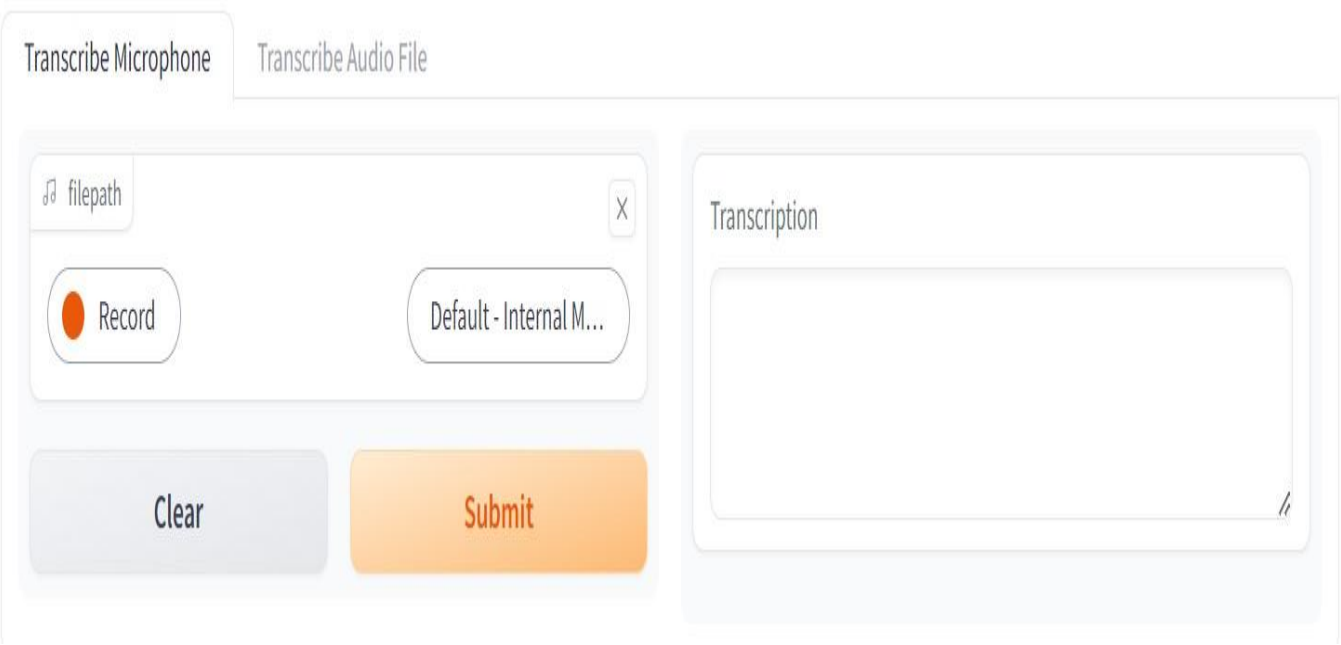


Fig1: speech to text interface

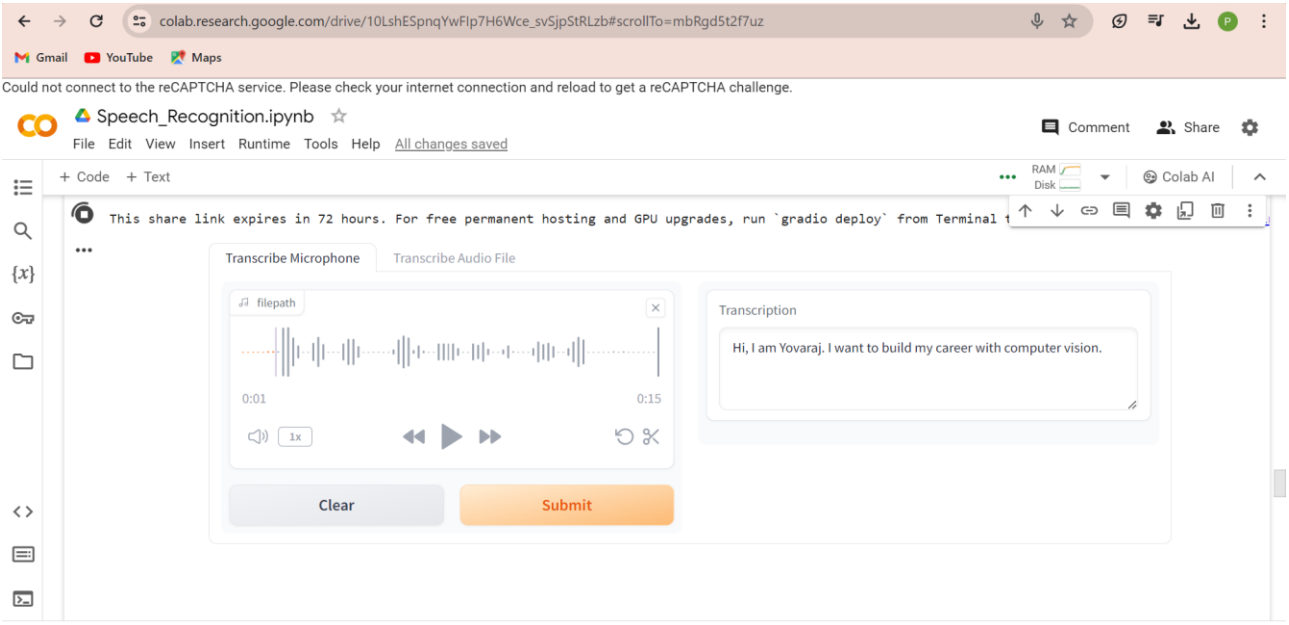


Fig2: Transcribed microphone

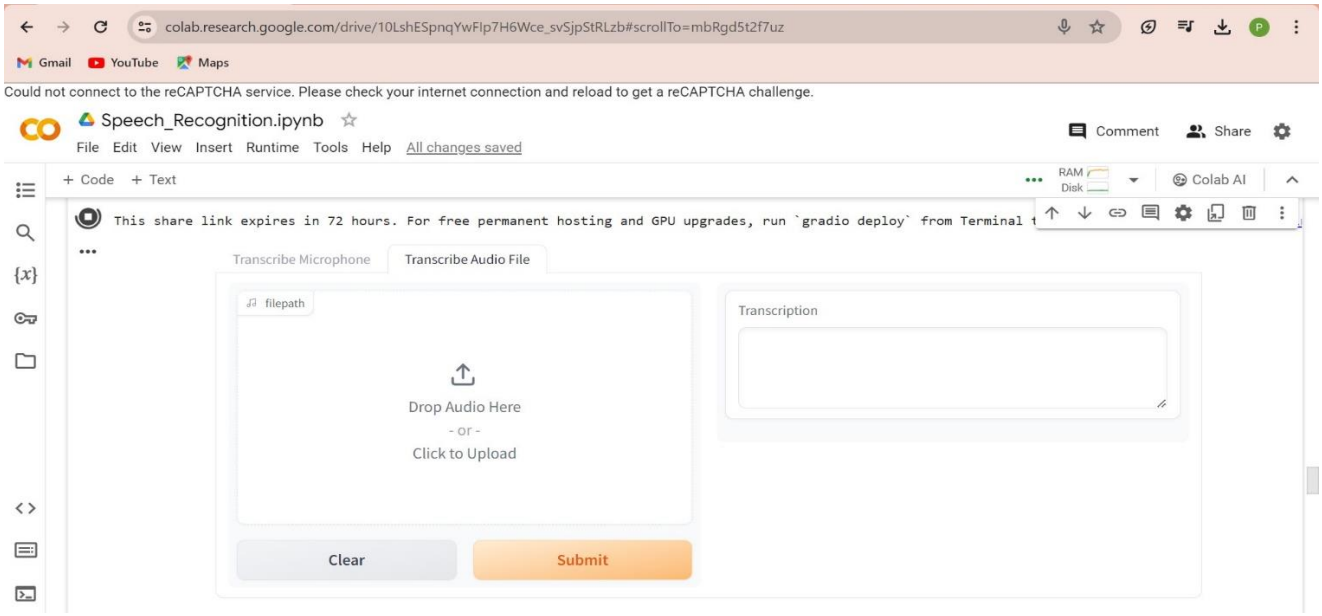


Fig3: Transcibe Audio file

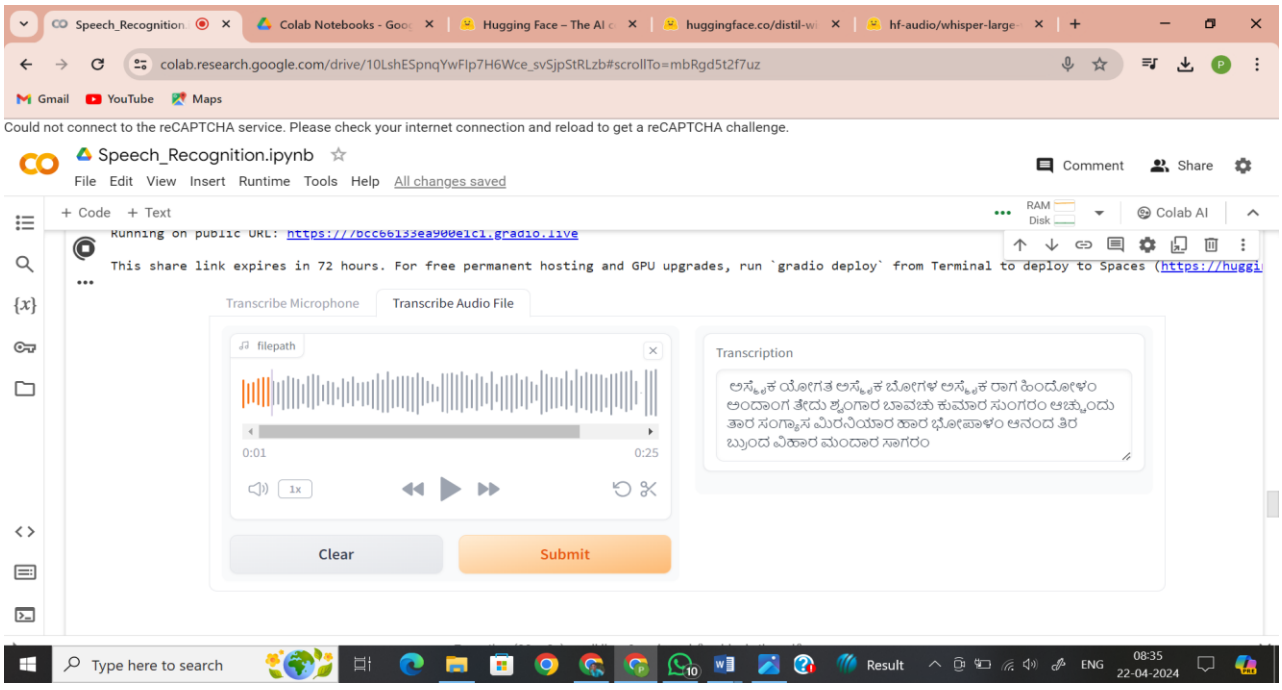


Fig4:Transcribe Audio file give results

8. CONCLUSION

the implementation of Multilingual Speech Recognition using the Whisper Large v3 model represents a significant advancement in the field of automatic speech recognition. By leveraging cutting-edge technologies such as the Hugging Face Transformers library and Gradio for interface design, this model offers a versatile and user-friendly solution for transcribing speech in multiple languages.

The model's robust features, including support for microphone input, file upload, and long-form transcription, enable seamless interaction and accommodate a wide range of user needs. Additionally, the Whisper Large v3 model's multilingual support ensures accessibility and applicability across diverse linguistic contexts, further enhancing its utility and versatility.

Through clear documentation, comprehensive usage guidelines, and acknowledgments of contributions and dependencies, this project fosters transparency, collaboration, and community engagement. Furthermore, the open-source licensing and acknowledgment of contributors underscore the commitment to fostering an inclusive and collaborative development environment.

In summary, Multilingual Speech Recognition using the Whisper Large v3 model stands as a testament to the advancements in speech recognition technology and its potential to revolutionize communication and accessibility across linguistic barriers. With its robust functionality, user-friendly interface, and broad language support, this model paves the way for enhanced accessibility, efficiency, and inclusivity in speech recognition applications.

8.REFERENCES

Links:

- https://huggingface.co/tasks/automatic-speech-recognition
- https://huggingface.co/openai/whisper-large-v3
- https://learn.deeplearning.ai/courses/open-source-models-hugging-face/lesson/7/automatic-speech-recognition
- https://www.youtube.com/watch?v=ywIyc8l1K1Q

9. APPENDIX

```
!pip install transformers

!pip install -U datasets

!pip install gradio

!pip install librosa

!pip install soundfile

from datasets import load_dataset

dataset=load_dataset("librispeech_asr",split='train.clean.100',streaming=True,
trust_remote_code=True)

example = next(iter(dataset))

dataset_head = dataset.take(5)
list(dataset_head)

list(dataset_head) [2]

example

from IPython.display import Audio as IPythonAudio
IPythonAudio(example["audio"]
["array"],rate=example["audio"]["sampling_rate"])

from transformers import pipeline
asr = pipeline(task="automatic-speech-recognition",
               model="openai/whisper-large-v3")

asr.feature_extractor.sampling_rate

example ['audio'] ['sampling_rate']

asr(example["audio"]["array"])

example["text"]

import os
import gradio as gr

demo = gr.Blocks()
```

```
def transcribe_speech(filepath):
    if filepath is None:
        gr.Warning("No audio found, please retry.")
        return ""
    output = asr(filepath)
    return output["text"]

mic_transcribe = gr.Interface(
    fn=transcribe_speech,
    inputs=gr.Audio(sources="microphone",
                    type="filepath"),
    outputs=gr.Textbox(label="Transcription",
                       lines=3),
    allow_flagging="never")

file_transcribe = gr.Interface(
    fn=transcribe_speech,
    inputs=gr.Audio(sources="upload",
                    type="filepath"),
    outputs=gr.Textbox(label="Transcription",
                       lines=3),
    allow_flagging="never",
)

with demo:
    gr.TabbedInterface(
        [mic_transcribe,
         file_transcribe],
        ["Transcribe Microphone",
         "Transcribe Audio File"],
    )

demo.launch(debug=True)

demo.close()

import soundfile as sf
import io

audio, sampling_rate = sf.read('/gudilo_badilo_song.mp3')

sampling_rate

asr.feature_extractor.sampling_rate

audio.shape
```

```
import numpy as np

audio_transposed = np.transpose(audio)

audio_transposed.shape

import librosa

audio_mono = librosa.to_mono(audio_transposed)

IPythonAudio(audio_mono,
             rate=sampling_rate)

asr(audio_mono)

sampling_rate

asr.feature_extractor.sampling_rate

audio_16KHz = librosa.resample(audio_mono,
                               orig_sr=sampling_rate,
                               target_sr=16000)

asr(
    audio_16KHz,
    chunk_length_s=30, # 30 seconds
    batch_size=4,
    return_timestamps=True,
)["chunks"]

import gradio as gr
demo = gr.Blocks()

def transcribe_long_form(filepath):
    if filepath is None:
        gr.Warning("No audio found, please retry.")
        return ""
    output = asr(
      filepath,
      max_new_tokens=256,
      chunk_length_s=30,
      batch_size=8,
    )
    return output["text"]

mic_transcribe = gr.Interface(
    fn=transcribe_long_form,
    inputs=gr.Audio(sources="microphone",
```

```
                    type="filepath"),
    outputs=gr.Textbox(label="Transcription",
                       lines=3),
    allow_flagging="never")

file_transcribe = gr.Interface(
    fn=transcribe_long_form,
    inputs=gr.Audio(sources="upload",
                    type="filepath"),
    outputs=gr.Textbox(label="Transcription",
                       lines=3),
    allow_flagging="never",
)

with demo:
    gr.TabbedInterface(
        [mic_transcribe,
         file_transcribe],
        ["Transcribe Microphone",
         "Transcribe Audio File"],
    )
demo.launch(debug=True)

demo.close()
```

RESOURCE:

- https://github.com/pandluruyuvaraj/multilingual-speech-recognition