Audio Transcriber & Summarizer using Streamlit

1. Introduction

With the rapid growth of digital communication and multimedia content, audio-based information has become a major part of modern life. Lectures, interviews, podcasts, and online discussions are often recorded for future reference, but extracting meaningful insights from hours of audio data can be challenging. Manual transcription is time-consuming, and summarizing large text transcripts can also be tedious.

To overcome these challenges, this project — **"Audio Transcriber & Summarizer using Streamlit"** — presents an end-to-end solution that can automatically transcribe speech into text and summarize the resulting transcript. The app combines state-of-the-art AI models with a user-friendly interface built entirely in Python using Streamlit. It is efficient, accurate, and suitable for both students and professionals who need to process audio content quickly.

2. Objectives

The objectives of this project are as follows:

1. To develop a lightweight, web-based interface for uploading and processing audio files.
2. To implement automatic speech recognition using OpenAI's Whisper model.
3. To generate concise summaries of long transcripts using the BART-large-CNN model from Hugging Face.
4. To enable users to download both the transcript and summarized text for offline use.
5. To demonstrate how modern AI tools can simplify audio-to-text workflows.
6. To provide a clear and visually appealing interface using only Streamlit components.

3. Methodology

This system integrates Natural Language Processing (NLP) and Speech Recognition within a single Python-based application. The methodology can be broken down into the following steps:

Step 1 — Audio Upload:

Users upload audio files in supported formats (MP3, WAV, M4A). Streamlit's built-in uploader simplifies this process. The file is temporarily stored on the system for processing.

Step 2 — Speech Recognition:

The uploaded audio file is passed to the Whisper model, which converts spoken words into written text. Whisper is a deep learning model trained on diverse datasets and can handle accents, background noise, and multiple languages effectively.

Step 3 — Text Summarization:

The transcribed text is summarized using Facebook's BART-large-CNN model. This model follows an encoder-decoder architecture and is optimized for generating short, meaningful summaries from long pieces of text.

Step 4 — Output Display and Download:

The transcript and summary are displayed on the Streamlit interface. Users can copy or download both results as text files for documentation or sharing.

4. Implementation

The project is implemented in Python using the Streamlit framework for the interface and the Hugging Face and Whisper libraries for AI functionalities. Below is the complete code for the application.

```
import streamlit as st
import whisper
from transformers import pipeline
from pathlib import Path

# Page configuration
st.set_page_config(page_title="Audio Transcriber & Summarizer", layout="centered")

st.title("■ Audio Transcriber & Summarizer")
st.write("Upload an audio file to convert speech into text and get a summarized version.")

uploaded_audio = st.file_uploader("■ Upload Audio File", type=["mp3", "wav", "m4a"])

if uploaded_audio is not None:
    st.audio(uploaded_audio, format="audio/mp3")
    st.success("■ File uploaded successfully!")

    st.info("Loading Whisper model... Please wait.")
    model = whisper.load_model("base")

    audio_path = Path("temp_audio.mp3")
    with open(audio_path, "wb") as f:
        f.write(uploaded_audio.read())

    st.info("Transcribing the audio... this may take a moment.")
    result = model.transcribe(str(audio_path))
    transcript = result["text"]

    st.subheader("■ Transcript")
    st.text_area("Full Transcript", transcript, height=200)

    st.info("Summarizing the text... please wait.")
    summarizer = pipeline("summarization", model="facebook/bart-large-cnn")

    if len(transcript.split()) > 700:
        transcript = " ".join(transcript.split()[:700])

    summary = summarizer(transcript, max_length=130, min_length=40, do_sample=False)[0]['summary_tex

    st.subheader("■ Summary")
    st.write(summary)

    st.download_button("■ Download Transcript", transcript, file_name="transcript.txt")
    st.download_button("■ Download Summary", summary, file_name="summary.txt")
```

5. Results and Discussion

The application performs speech-to-text and summarization efficiently for a variety of audio formats. The Whisper model offers high accuracy in transcribing even low-quality recordings. The BART summarizer produces grammatically correct and concise summaries.

Sample Output:

- Transcript: Displays the complete converted text from the uploaded audio.
- Summary: Displays a shorter version that highlights key points.

The Streamlit interface provides an intuitive workflow, and download buttons make it convenient to save results. Performance depends on the hardware configuration; GPU usage speeds up both transcription and summarization significantly.

6. Conclusion

The “Audio Transcriber & Summarizer using Streamlit” project demonstrates how artificial intelligence can simplify real-world problems. By integrating Whisper and BART, this app provides a complete audio-to-summary pipeline accessible through a simple web interface.

Future Scope:

1. Integration with multilingual Whisper models to support non-English speech.
2. Addition of sentiment analysis on transcripts.
3. Option for PDF or Word output format.
4. Deployment on cloud platforms like Streamlit Cloud or Hugging Face Spaces.

This project showcases how open-source AI tools and frameworks can work together to create professional-grade applications that bridge the gap between speech and written language.