COMPLETE WORKING CODE

Build Speech recognition pipelines with OpenAI Gemini

Learn to transcribe, translate, and analyze audio files using OpenAI Whisper and Google Gemini

Transcription | Translation | Summarization

Naved Khan
Senior Gen-AI Engineer

+ Follow

What is Speech Recognition?

1

Understanding the Basics

Speech recognition converts spoken language into text. It's the foundation for building intelligent audio processing pipelines.

Key Terms

→ Transcription — Converting speech to text in the same language

→ Translation — Converting speech to text in a different language

→ Summarization — Creating concise summaries from transcripts

What We'll Build

→ Audio transcription using OpenAI Whisper

→ Audio translation with Gemini

→ Transcript summarization with GPT-4

→ Complete working code examples

Why This Matters

Speech recognition powers voice assistants, meeting notes, content creation, and multilingual communication systems.

Naved Khan
Senior Gen-AI Engineer

+ Follow

Gemini: Step 1 - Setup & Installation

setup.py

```
!pip install google-genai -q

from google.colab import userdata
gemini_key = userdata.get('GEMINI')

from google import genai
```

What's Happening Here?

→ Install google-genai — Installs Google's Generative AI Python package

→ Get API key — Retrieves your Gemini API key from Colab userdata

→ Import genai — Imports the Google Generative AI client library

→ Ready to create the Gemini client and process audio files

Naved Khan

Senior Gen-AI Engineer

+ Follow

Gemini: Step 2 - Upload Audio File

upload.py

```python
from google.colab import files

uploaded = files.upload()
audio_filename = list(uploaded.keys())[0]

client = genai.Client(api_key = gemini_key)

myfile = client.files.upload(file=audio_filename)
```

How This Works

→ Upload file — Uses Colab's file upload interface to get audio file

→ Get filename — Extracts the uploaded file name from the dictionary

→ Create client — Initializes Gemini client with your API key

→ Upload to Gemini — Uploads the audio file to Gemini's file storage

→ The file is now ready for processing

Naved Khan
Senior Gen-AI Engineer

+ Follow

Gemini: Step 3 - Transcribe & Translate

4

translate.py

```
response = client.models.generate_content(
    model="gemini-2.5-flash",
    contents=["Give me the Translation of the audio file",
myfile]
)

print(response.text)
```

What This Does

→ Generate content — Sends audio file and translation request to Gemini

→ Model selection — Uses gemini-2.5-flash for fast processing

→ Translation prompt — Requests translation of the audio content

→ Print result — Displays the translated text output

→ Gemini automatically transcribes and translates in one step

Naved Khan
Senior Gen-AI Engineer

+ Follow

OpenAI: Step 1 - Setup & Transcription

5

transcribe.py

```
!pip install openai -q

from google.colab import userdata
openai_key = userdata.get('OPENAI')

from google.colab import files
uploaded = files.upload()
audio_filename = list(uploaded.keys())[0]

from openai import OpenAI
client = OpenAI(api_key = openai_key)

with open(audio_filename,"rb") as audio:
    transcription = client.audio.transcriptions.create(
        model="whisper-1",
        file=audio
    )
    print(transcription.text)
```

Key Steps

→ Setup & Upload — Install OpenAI package, get API key, and upload audio file

→ Create client — Initialize OpenAI client with your API key

→ Transcribe — Use Whisper-1 model to convert speech to text

Naved Khan

+ Follow

OpenAI: Step 2 - Translation

translate.py

```python
with open(audio_filename,"rb") as audio:
    translation = client.audio.translations.create(
        model="whisper-1",
        file=audio,
    )
    print(translation.text)
```

Translation Process

→ Open audio file — Reads the audio file in binary mode

→ Create translation — Uses Whisper-1 translations API

→ Auto-detect language — Automatically detects source language

→ Translate to English — Converts speech to English text

→ Print result — Displays the translated text

→ Perfect for multilingual audio content

Naved Khan

Senior Gen-AI Engineer

+ Follow

OpenAI: Step 3 - Summarization

summarize.py

```
response = client.chat.completions.create(
    model="gpt-4",
    messages = [
        {
            "role" : "system",
            "content": "Summarize the transcript in bullet
points"
        },
        {
            "role" : "user",
            "content": transcription.text
        }
    ]
)

print(response.choices[0].message.content)
```

Summarization Flow

→ Chat completions — Uses GPT-4 with system prompt to summarize in bullet points

→ Pass transcript — Sends transcribed text as user content

→ Get & display — Retrieves and prints the bullet-point summary

Naved Khan

+ Follow

Want more content like this?

Naved Khan

Senior Gen-AI Engineer

Tap that follow button and stay in the loop!

❤️ Like 💬 Comment 🔁 Share 🧨 Save