Lab: Speech to Text: Conversion Techniques

Introduction

In the digital age, the ability to convert text into spoken audio has numerous applications, from making content more accessible to creating engaging multimedia experiences. OpenAI's Audio API harnesses advanced text-to-speech (TTS) technology, offering a user-friendly interface to generate realistic voiceovers. With six unique voices and support for multiple languages, this tool is ideal for various use cases, including blog narration, multilingual audio production, and real-time streaming.

Key Features

- Six Distinct Voices: Alloy, Echo, Fable, Onyx, Nova, Shimmer.
- Multiple Language Support: Follows Whisper model's language capabilities.
- Real-Time Audio Streaming: Utilizes chunk transfer encoding.
- Variety of Output Formats: MP3, Opus, AAC, FLAC.

Lab Solution

Solution notebook for this lab can be found at `~/work/nlp-generative-ai-bootcamp/Lab04/openai_audio_api.ipynb`

Initial Setup

Start by importing the necessary module and setting up your OpenAI API key:

```
import os

# Prompt for the API key
api_key = input("Enter your OpenAI API key: ")

# Set the environment variable
os.environ["OPENAI_API_KEY"] = api_key
```

Import Libraries

Once the SDK is installed, import the required libraries, prep the data and call the model:

```
from pathlib import Path
from openai import OpenAI
client = OpenAI()
speech_file_path = Path("speech.mp3")
response = client.audio.speech.create(
  model="tts-1",
  voice="alloy",
  input="My name is Dr. Lee and I am a Data Analytics Professor at Miami Dade College. I am trying this ChatGPT model for the first time and it is super cool!"
```

```
)
response.stream_to_file(speech_file_path)
from IPython.display import Audio

# Play the generated audio file
Audio("./speech.mp3")
```

Explanation

- `client.audio.speech.create()` : Main function for speech generation.
- `model` : "tts-1" for standard quality, "tts-1-hd" for high definition.
- `voice` : Choose from the six available options.
- `input` : Your text to be converted into speech.
- `response.stream_to_file()` : Saves the audio to the specified path.

Use Cases and Applications

Narrating a Blog Post

Transform your written content into an engaging audio format, making it more accessible and appealing to a broader audience.

Language Learning Tools

Create multilingual educational content to aid in language learning.

Real-Time Audio Streaming

Develop interactive applications that require live voiceover, such as virtual assistants or online gaming.

Conclusion

OpenAI's Audio API offers a versatile and powerful platform for text-to-speech applications. Whether you're a content creator, educator, or developer, this tool provides an accessible way to enhance your projects with lifelike audio.

Remember to adhere to OpenAI's Usage Policies by informing end-users that the voice they hear is AI-generated. Embrace the future of digital audio with OpenAI!

Addendum: Translating and Generating Audio in English and Spanish

Preparing the English Text

Store your English text in a variable named `my_speech`:

```
my_speech = """
My name is Dr. Lee and I am a Data Analytics Professor at Miami Dade College.
I am trying this ChatGPT model for the first time and it is super cool!
"""
```

Translating the Text to Spanish Using OpenAI API

Use the OpenAI API to translate `my_speech` to Spanish:

```
import openai

# Translate the English text to Spanish
response = openai.completions.create(
  model="gpt-3.5-turbo-instruct",
  prompt="Translate the following text to Spanish: {}".format(my_speech),
  max_tokens=100
)

# Extract the translated text
my_speech_es = response.choices[0].text.strip()
```

Generating Audio in English and Spanish

Generate audio files for both the original English text and its Spanish translation.

```
# Import the necessary library
from openai import OpenAI

client = OpenAI()

# Generate English audio
response_en = client.audio.speech.create(
    model="tts-1",
    voice="alloy",
    input=my_speech
)
response_en.stream_to_file("english_speech.mp3")

# Generate Spanish audio
response_es = client.audio.speech.create(
    model="tts-1",
    voice="alloy",  # Choose a voice suitable for Spanish
    input=my_speech_es
)
response_es.stream_to_file("spanish_speech.mp3")
```

Playing the Audio Files

Play the generated audio files using IPython's Audio class.

```
from IPython.display import Audio

# Play the English audio
Audio("english_speech.mp3")
```

```
# Play the Spanish audio
Audio("spanish_speech.mp3")
```

This process allows you to translate text into Spanish using the OpenAI API and then generate audio in both languages in a Jupyter environment. It's a practical way to create bilingual audio content for various applications, enhancing both accessibility and engagement for diverse audiences.