Ex.No.7 Exploring Prompting Techniques for AI Audio Generation

Name : Justine Irudhayaraj J
Register Number: 212221080033

Aim:
The experiment aims to explore how different prompt styles influence AI-generated audio, such as music, sound effects, and speech. It focuses on understanding prompt engineering and optimizing prompt design for better and more relevant outputs.

Softwares Required:

- **Python (3.8+)** and an IDE (Jupyter, VS Code).
- **Libraries:** requests, openai, torchaudio (optional for playback).
- **APIs:** OpenAI (Whisper for speech synthesis), Google Cloud Text-to-Speech, Hugging Face
- (MusicGen for music, Sound Effects models).
- **API Keys:** Required for OpenAI, Google Cloud, and Hugging Face.

Key Concepts:

- **Prompt Engineering:** Crafting input prompts that guide AI models to generate desired audio outputs.
- **Audio Generation:** Using AI models to create speech, music, or sound effects based on given prompts.
- **Prompt Optimization:** Refining prompt inputs for improved quality and control over the audio output.

Experiment Design:

- **Experiment 1: Speech Generation**

Prompts Used:

- Basic Prompt: "Say hello in a friendly tone."
- Detailed Prompt: "Generate a professional greeting message for a virtual assistant."
- Contextual Prompt: "Speak as if you are introducing a new product at a tech conference."

Code:

```
import requests
API_KEY = "your_google_api_key"
url = "https://texttospeech.googleapis.com/v1/text:synthesize"
def generate_speech(prompt):
headers = {"Authorization": f"Bearer {API_KEY}"}
payload = {
"input": {"text": prompt},
"voice": {"languageCode": "en-US", "name": "en-US-Wavenet-D"},
"audioConfig": {"audioEncoding": "MP3"}
```

```
}r
esponse = requests.post(url, headers=headers, json=payload)
if response.status_code == 200:
with open("speech.mp3", "wb") as file:
file.write(response.content)
print("Speech generated: speech.mp3")
else:
print("Error in generation:", response.json())
generate_speech("Speak as if you are introducing a new product at a tech conference.")
```

Experiment 2: Music Generation Using MusicGenCode:

```
import requests
API_KEY = "your_huggingface_api_key"
url = "https://api-inference.huggingface.co/models/facebook/musicgen"
def generate_music(prompt):
headers = {"Authorization": f"Bearer {API_KEY}"}
payload = {"inputs": prompt}
response = requests.post(url, headers=headers, json=payload)
if response.status_code == 200:
audio_url = response.json().get("audio_url", "No URL")
print("Generated Music URL:", audio_url)
else:
print("Error in generation:", response.json())
# Example usage
generate_music("Compose a relaxing acoustic guitar tune with ambient background sounds.")
```

Experiment 3: Sound Effect GenerationCode:

```
def generate_sound_effect(prompt):
url = "https://api-inference.huggingface.co/models/sound-effect-model"
headers = {"Authorization": f"Bearer your_huggingface_api_key"}
payload = {"inputs": prompt}
response = requests.post(url, headers=headers, json=payload)
if response.status_code == 200:
audio_url = response.json().get("audio_url", "No URL")
print("Sound Effect URL:", audio_url)
else:
print("Error in generation:", response.json())
# Example usage
generate_sound_effect("Soft rain on a metal roof at night.")
```

Output and Result:

1. **Speech Generation:** Detailed prompts produce clearer and more expressive speech.
2. **Music Generation:** Genre and mood-specific prompts yield more stylistically accurate music.
3. **Sound Effect Generation:** Context-rich prompts create more vivid and realistic sounds.

Conclusion:

This experiment demonstrates that prompt specificity and structure significantly influence thequality and relevance of AI-generated audio. Effective prompt engineering can produce high-quality outputs tailored to speech, music, or sound effects.