AI Pyramid - CosyVoice2 音色クローン

CosyVoice2 は、大規模言語モデル（LLM）に基づいた高品質な音声合成システムであり、自然で滑らかな音声を合成することができます。本ドキュメントでは、OpenAI API と互換性のある完全な呼び出し方法を提供します。ユーザーは対応する StackFlow ソフトウェアパッケージをインストールするだけで、すぐに使用を開始できます。

1. 準備

AI Pyramid ソフトウェアパッケージの更新を参考に、以下の依存パッケージとモデルのインストールを完了させてください。

```
apt update
```

コア依存パッケージのインストール:

```
apt install lib-llm llm-sys llm-cosy-voice llm-openai-api
```

CosyVoice2 モデルのインストール:

```
apt install llm-model-cosyvoice2-0.5b-ax650
```

モデル更新のヒント

新しいモデルをインストールするたびに、モデルリストを更新するために `systemctl restart llm-openai-api` コマンドを手動で実行する必要があります。

パフォーマンスについて

CosyVoice2 は高性能なニューラルネットワーク音声生成モデルです。自然で滑らかな音声を合成できますが、リソースが限られたデバイスでは以下の制限があります：最大生成音声長は 27 秒です。また、初回モデル読み込みには時間がかかる場合があります。アプリケーションのシナリオに合わせて、音声の長さを適切に調整してください。

2. 基本的な呼び出し例

Curl を使用した呼び出し

```
curl http://127.0.0.1:8000/v1/audio/speech \
  -H "Content-Type: application/json" \
  -d '{
    "model": "CosyVoice2-0.5B-ax650",
    "response_format": "wav",
    "input": "名にし負はば、いざこと問はむ、都鳥。わが思ふ人は、ありやなしやと、住の江の岸による浪、よるさへや、夢の通ひ路、人目
  }' \
  -o output.wav
```

Python を使用した呼び出し

```
from pathlib import Path
from openai import OpenAI

client = OpenAI(
    api_key="sk-",
    base_url="http://127.0.0.1:8000/v1"
)

speech_file_path = Path(__file__).parent / "output.wav"
with client.audio.speech.with_streaming_response.create(
  model="CosyVoice2-0.5B-ax650",
  voice="prompt_data",
  response_format="wav",
  input='名にし負はば、いざこと問はむ、都鳥。わが思ふ人は、ありやなしやと、住の江の岸による浪、よるさへや、夢の通ひ路、人目よくら
) as response:
  response.stream_to_file(speech_file_path)
```

3. 音色クローン

3.1 クローン用スクリプトの取得

以下のいずれかの方法で CosyVoice2 クローン用スクリプトを取得してください。

方法 1：手動ダウンロード

CosyVoice2 スクリプトリポジトリ にアクセスしてダウンロードし、AI Pyramid デバイスにアップロードします。

方法 2：コマンドラインによるクローン

> **依存関係のチェック**
>
> システムに **git lfs** がインストールされていない場合は、Git LFS インストールガイドを参照してインストールしてください。

```
git clone --recurse-submodules https://huggingface.co/M5Stack/CosyVoice2-scripts
```

3.2 ディレクトリ構造の説明

クローン完了後のディレクトリ構造は以下の通りです。

```
root@m5stack-AI-Pyramid:~/CosyVoice2-scripts# ls -lh
total 28K
drwxr-xr-x 2 root root 4.0K Jan  9 10:26 asset
drwxr-xr-x 2 root root 4.0K Jan  9 10:26 CosyVoice-BlankEN
drwxr-xr-x 2 root root 4.0K Jan  9 10:27 frontend-onnx
drwxr-xr-x 3 root root 4.0K Jan  9 10:26 pengzhendong
-rw-r--r-- 1 root root   24 Jan  9 10:26 README.md
-rw-r--r-- 1 root root  103 Jan  9 10:26 requirements.txt
drwxr-xr-x 3 root root 4.0K Jan  9 10:26 scripts
```

3.3 音声サンプルの処理

ステップ 1：仮想環境の作成

CosyVoice2-scripts ディレクトリに移動してください。

```
cd CosyVoice2-scripts/
```

初回操作

初めて Python 仮想環境を作成する場合は、`apt install python3.10-venv` を実行する必要があります。

```
python3 -m venv cosyvoice
```

ステップ 2：仮想環境のアクティベート

```
source cosyvoice/bin/activate
```

ステップ 3：依存パッケージのインストール

```
pip3 install torch torchaudio --index-url https://download.pytorch.org/whl/cpu
```

```
pip install -r requirements.txt
```

ステップ 4：処理スクリプトの実行

音色処理スクリプトを実行し、音色特徴ファイルを生成します。

```
python3 scripts/process_prompt.py --prompt_text asset/zh_woman1.txt --prompt_speech asset/zh_woman1.wav --ou
```

スクリプト実行成功時の出力例:

```
(cosyvoice) root@m5stack-AI-Pyramid:~/CosyVoice2-scripts# python3 scripts/process_prompt.py --prompt_text as
2026-01-09 10:41:18.655905428 [W:onnxruntime:Default, device_discovery.cc:164 DiscoverDevicesForPlatform] GP
prompt_text 希望你以后能够做的比我还好呦。
fmax 8000
prompt speech token size: torch.Size([1, 87])
```

3.4 音色をモデルディレクトリにデプロイ

処理済みの音色特徴ファイルをモデルデータディレクトリにコピーします。

```
cp -r zh_woman1 /opt/m5stack/data/CosyVoice2-0.5B-ax650/
```

新しい音色設定を読み込むためにモデルサービスを再起動します。

```
systemctl restart llm-sys
```

音色の置換について

デフォルトのクローン音色を置換する場合は、`/opt/m5stack/data/models/mode_CosyVoice2-0.5B-ax650.json` ファイル内の `prompt_dir` フィールドを新しい音色ディレクトリに変更してください。音色を置換するたびに、モデルサービスを再初期化する必要があります。

4. クローンした音色での呼び出し

Curl を使用した呼び出し

```
curl http://127.0.0.1:8000/v1/audio/speech \
  -H "Content-Type: application/json" \
  -d '{
    "model": "CosyVoice2-0.5B-ax650",
    "voice": "zh_woman1",
    "response_format": "wav",
    "input": "名にし負はば、いざこと問はむ、都鳥。わが思ふ人は、ありやなしやと、住の江の岸による浪、よるさへや、夢の通ひ路、人目
  }' \
  -o output.wav
```

Python を使用した呼び出し

```
from pathlib import Path
from openai import OpenAI

client = OpenAI(
    api_key="sk-",
    base_url="http://127.0.0.1:8000/v1"
)

speech_file_path = Path(__file__).parent / "output.wav"
with client.audio.speech.with_streaming_response.create(
  model="CosyVoice2-0.5B-ax650",
  voice="zh_woman1",
  response_format="wav",
  input='名にし負はば、いざこと問はむ、都鳥。わが思ふ人は、ありやなしやと、住の江の岸による浪、よるさへや、夢の通ひ路、人目よくら
) as response:
  response.stream_to_file(speech_file_path)
```

事例

実行例では、Open AIの依存ライブラリをインストールし、Ollamaサービスを再起動する必要があります。

```
pip3 install openai
```

```
systemctl restart llm-*
```

```python
# main.py

from pathlib import Path
from openai import OpenAI
import subprocess


def main():
    # Initialize the OpenAI client
    client = OpenAI(
        api_key="sk-",  # Replace with your actual API key
        base_url="http://127.0.0.1:8000/v1"
    )

    # Temporary file paths
    base_dir = Path(__file__).parent
    raw_audio_path = base_dir / "temp_raw.wav"
    transcoded_audio_path = base_dir / "temp_48k_stereo.wav"

    print("=== Interactive Speech Synthesis Mode ===")
    print("Enter text and press Enter to generate speech.")
    print("Type 'quit' or 'exit' to stop.\n")

    while True:
        # 1. Read user input
        input_text = input("Enter text (quit/exit to stop): ").strip()

        # Exit condition
        if input_text.lower() in ["quit", "exit"]:
            print("Exiting program...")
            break

        if not input_text:
            print("Error: Input text cannot be empty.\n")
            continue

        try:
            # 2. Generate raw audio from the TTS API
            print("Generating speech...")
            with client.audio.speech.with_streaming_response.create(
                model="CosyVoice2-0.5B-ax650",
                voice="zh_woman1",
                response_format="wav",
                input=input_text,
            ) as response:
                response.stream_to_file(raw_audio_path)

            # 3. Transcode to 48 kHz stereo WAV using FFmpeg
            print("Transcoding audio...")
            ffmpeg_cmd = [
                "ffmpeg",
                "-y",                    # Overwrite output file if it exists
                "-i", str(raw_audio_path),
                "-ar", "48000",          # Set sample rate to 48 kHz
                "-ac", "2",              # Set channel count to stereo
```

```
                "-f", "wav",
                str(transcoded_audio_path)
            ]

            # Run transcoding
            # Remove stdout/stderr redirection if you need FFmpeg logs for debugging
            subprocess.run(
                ffmpeg_cmd,
                check=True,
                stdout=subprocess.DEVNULL,
                stderr=subprocess.DEVNULL
            )

            # 4. Play the transcoded audio with tinyplay
            print("Playing audio...\n")
            tinyplay_cmd = ["tinyplay", str(transcoded_audio_path)]
            subprocess.run(tinyplay_cmd, check=True)

        except subprocess.CalledProcessError as e:
            print(
                f"Command execution failed. Please make sure FFmpeg and tinyplay "
                f"are installed and available in PATH: {e}\n"
            )
        except Exception as e:
            print(f"An error occurred: {e}\n")
        finally:
            # Remove temporary files
            raw_audio_path.unlink(missing_ok=True)
            transcoded_audio_path.unlink(missing_ok=True)


if __name__ == "__main__":
    main()
```