pyopenjtalkによるテキスト音声変換

　目標は、デイサービスで働くロボット。voskの次は、テキスト音声変換である。1年ぐらい前は、open_jtalkしか見つけることができなかった。現在、いくつかの選択肢が存在するが、私のなかで実績のあるopen_jtalkを採用する。open_jtalkをpythonから使えるようにしたのが、pyopenjtalkである。今回は、T8のみで行う。

構成

・T8$　　Client PC CF-T8 Debian10
・pyopenjtalk# T8 Docker Container

目次

Dockerのコンテナを用意

ChatGPTの回答を参考にDokerfileを作成する。

Dockerfile

```
# ベースイメージとして軽量なPythonイメージを使用
#FROM python:3.9-slim
FROM python:3.10-slim
# 必要なパッケージをインストール
RUN apt-get update && apt-get install -y --no-install-recommends \
    build-essential \
    libsndfile1-dev \
    libpulse-dev \
    neovim \
    && apt-get clean \
    && rm -rf /var/lib/apt/lists/*
# pyopenjtalkをインストール
RUN pip install --no-cache-dir pyopenjtalk
# 作業ディレクトリを設定
WORKDIR /app
# サンプルスクリプトをコンテナにコピー
#COPY run.py /app/run.py
COPY init.vim /root/.config/nvim/init.vim
# エントリーポイントを設定
#ENTRYPOINT ["python", "run.py"]
# デフォルトのコマンドを/bin/bashに設定
CMD ["/bin/bash"]
```

なる可能性がある

docker imageの作成

上記の2つのファイルを同一のフォルダに保存し、コマンドを実行する。

```
T8:~/a-ros/Dockerfile/open_jtalk$ docker build -t pyopenjtalk-image .
```

docker containerの実行

```
T8:~$ docker run -it -e DISPLAY=$DISPLAY -v
/tmp/.X11-unix/:/tmp/.X11-unix --name pyopenjtalk --net host
pyopenjtalk-image
```

docker containerの再起動

T8:~$ docker start pyopenjtalk
T8:~$ docker attach pyopenjtalk

script実行

スクリプトファイル run01.py作成

```
import pyopenjtalk
import numpy as np
import soundfile as sf

# 音声合成を実行
text = "こんにちは、pyopenjtalkのテストです。"
waveform, sr = pyopenjtalk.tts(text)

# 音量を下げる
volume_factor = 0.00001  # 0.0〜1.0の範囲で設定
wave = waveform * volume_factor

# 音声をWAVファイルとして保存
sf.write("output.wav", wave, sr)
print("音声ファイルが生成されました: output.wav")
```

実行

pyopenjtalk:/app# python3 run01.py

結果

生成された音声ファイルをホストにコピーし、以下を実行する。良好な結果を得た。

```
T8:~/Music$ docker cp pyopenjtalk:/app/output.wav .
                    Successfully copied 430kB to /home/yo/Music/.
T8:~/Music$ aplay output.wav
再生中 WAVE 'output.wav' : Signed 16 bit Little Endian, レート 48000 Hz, モノラル
```

当初、音量を下げるプロセスがない状態で実行したところ、音声を聞き取ることができなかった。そのままでは波形が大きすぎるようだ。

今後を思案

20241222 pythonのバージョン統一

さて、各要素の作動確認を行ったので、一連の流れを繋げる必要がある。現在の状況としては、各要素について、Docker containerを作成し、独立した環境で作動確認を行っている。対話ロボットを作成するのであれば、一連の流れを同一containerで作動させることができれば、効率が良い。pythonのバージョンを確認すると。

T8	ros noetic	3.8.10
Ryzen9	ggml	3.10
Ryzen9	vosk	3.9
T8	pyopenjtalk	3.10

ros noetic containerのpythonバージョンが、3.8.10で各要素が作動するか？
以下の条件に配慮する。
1. ros_noeticのcontainerで全ての要素が実行できるか
2. 同じpython3.10で統一したcontainerで対話システムを構成する
3. ubunt(noetic)のバージョン20.04で、python3.10が使えるか？
　　geminiは、可能といっている。
4. Ryzen9で実行する必要がある

以上を整理
条件2より、　　voskがpython3.10で作動することを確認
条件1，3より、noeticのpythonを3.10で作動を確認
条件4より、　　Ryzen9で実行する。

docker container noetic ubuntuのバージョン確認

noetic yo@T8:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:　Ubuntu 20.04.6 LTS
Release:　　20.04
Codename:　　focal

docker container noetic pythonのバージョン確認

noetic yo@T8:~$ python3 --version
Python 3.8.10

20241224 Python3.8.10での動作確認

動作確認OK。これで、全ての構成要素で、ros noeticでの動作確認が取れました。
ggmlについては、もともとpythonの関与は無いので問題なし。
Ryzen9でros noeticの環境を構築し、各nodeを作成し、実行していく。

参考

[1]ChatGPT

質問

pyopnejtalkを使うための最低限の機能を有するdocker containerを作りたいので、Dockerfileを教えてください。

回答

`pyopenjtalk`はPython向けのOpen JTalkラッパーです。それを動作させるDockerコンテナを構築するために、Python環境と`pyopenjtalk`の依存関係をインストールする最低限のDockerfileを以下に示します。

```*dockerfile*
# ベースイメージとして軽量なPythonイメージを使用
FROM python:3.9-slim
# 必要なパッケージをインストール
RUN apt-get update && apt-get install -y --no-install-recommends \
    build-essential \
    libsndfile1-dev \
    libpulse-dev \
    && apt-get clean \
    && rm -rf /var/lib/apt/lists/*
# pyopenjtalkをインストール
```

RUN pip install --no-cache-dir pyopenjtalk
作業ディレクトリを設定
WORKDIR /app
サンプルスクリプトをコンテナにコピー
COPY run.py /app/run.py
エントリーポイントを設定
ENTRYPOINT ["python", "run.py"]
```

### `run.py`の例

```python
import pyopenjtalk
# サンプルテキスト
text = "こんにちは、pyopenjtalkのテストです。"
# 音声合成を実行
wave, sr = pyopenjtalk.tts(text)
# 音声をWAVファイルとして保存
import soundfile as sf
sf.write("output.wav", wave, sr)
print("音声ファイルが生成されました: output.wav")
```

ビルドと実行方法

1. Dockerイメージをビルドします:
   ```bash
   docker build -t pyopenjtalk-minimal .
   ```
2. コンテナを実行します:
   ```bash
   docker run --rm -v $(pwd):/app pyopenjtalk-minimal
   ```
3. カレントディレクトリに`output.wav`が生成されます。

このDockerfileは、`pyopenjtalk`を動作させるために必要なPython環境と依存ライブラリを最低限インストールします。また、必要に応じてPythonのバージョンや追加パッケージをカスタマイズしてください。
   ```