Spleeterによる音源分離を利用した音声認識及び翻訳機能の実装

3年C組　岡本 晃朋
指導教諭　藤野 智美

１．要約

本研究では、私が参加したNICT(情報通信研究機構)主催の「多言語音声翻訳ハッカソン」の内容と開発したシステムについて報告する。翻訳機の対象を話者同士以外にも発展させ、楽曲中のボーカルに対しても翻訳する機能の開発を目指した。その手法としてSpleeterを使用し、曲中の歌詞を抽出後、その歌詞に対して翻訳を行うシステムを目指した。

キーワード　Spleeter, 音声分離, 音声認識,　Python,　PHP

２．研究の背景と目的

多言語音声翻訳ハッカソンに出場するにあたって、「言葉の壁」をなくすことに貢献する新しいアイデア、及び翻訳機を作成する必要があった。普通の翻訳機では面白くないと思うので、翻訳対象を他のもの、例えば楽曲中の歌詞などにも向ければ、会話以外の用途にも翻訳機を使用できると考えた。そこで、楽曲中の歌詞を認識し、その翻訳を行うことを目標とした。

３．研究内容

３．１　翻訳機①の作成

今回のハッカソンではNICTが開発した多言語音声翻訳機能(音声認識、翻訳、音声合成)を、「共通API」により利用できる「サンドボックスサーバー」を開放していただいた。そこで、最初に共通APIである“mimi®API　Platform”(以降、API)に慣れることを目標に、単純な翻訳機を作成してみることにした。

３．１．１　翻訳機①の概要

APIは、実行の際はプロトコルを用いてデータを転送するライブラリと、コマンドラインツールを提供するcURLにて行っている。ただ、cURLを使用したCUIでの実行は難しいかと思うので、現在最も多く使用されている汎用スクリプト言語である「PHP」に書き換えて行う。また、複数デバイスでの使用を想定し、HTMLにて構築する。

３．１．２　翻訳機①の実装

翻訳機の仕組みなどをここに示す。使用した言語は、HTML, CSS, JavaScript, PHPである。

翻訳機の大まかな実装方法としては、HTML上で翻訳対象をJavaScriptにて取得する。その情報をAPIにわたし、Resultを HTML 上に表示するという仕組みである。また、APIにて行う動作としては、次の3つの動作が主である。

・token の取得（code1）
・翻訳
・音声合成

HTML 上から取得した文章を翻訳し、音声合成するのに token が必要である。操作面についてはよりわかりやすく、直感的に操作できることを目標として、図のような UI を作成した。

図:作成した翻訳機

翻訳機に対する工夫点として、token 作成回数を減らすために、token の使用期限を確認する動作を実装した。また、この時点での問題点としては、翻訳を実施するときに言語判別ができないため、翻訳前に入力言語の指定をしないといけないということがあげられる。

３．１．３ 翻訳機①の課題

現時点で考えた問題点として、この翻訳機はオリジナリティに欠けていること、UI がスマートフォンなどの縦長の使用のみを想定しているため、タブレットや PC ではデザインが崩れてしまうという点があげられる。

３．２ 翻訳機②の作成

新たな翻訳機として、楽曲中の歌詞を抽出し、それに対して翻訳が可能な翻訳機②を作成することにした。

３．２．１ 翻訳機②の概要

翻訳機②の作成方法として、Spleeter というライブラリを使用する。これにより、楽曲中から Vocal 部分だけを抽出し、抽出部分に対して翻訳を行う。

３．２．２ Spleeter とは

Spleeter とは、音声分離技術によってさまざまな曲のオーディオデータを楽器やグループごとの最大 4 つの stem データに分離できるライブラリある。ドラムやベースの分離精度はあまり高くないが、Vocal の分離精度は高い。

３．２．３ 翻訳機②の実装

目的にあった翻訳機を作成するにあたり、Spleeter は python にて使えるモジュールなので、cURL 文を Request 文に書き直して使用する。ただし、後に thon で構築することにより、Android や IOS での動作は難しいと考えるため、現時点ではすぐに使えるようなものではないと考える。言語は Python を使用した。

cURL から python を使用した requests 文に変換するにあたっては、すべての値をそのまま構文に当てはめるだけで行った。実際には下記のコード①のように、response を json 形式にしている。

```
curl --request POST ¥
        --url https://auth.mimi.fd.ai/v2/token ¥
        --header 'Accept: application/json' ¥
        --header 'Content-Type: Content-Type'
        --data
grant_type=https://auth.mimi.fd.ai/grant_type/clie
nt_credentials ¥
        --data scope=Scope ¥
        --data client_id=client-id ¥
        --data client_secret=client_secret
```

```
import requests

url = "https://auth.mimi.fd.ai/v2/token"
payload = payload
headers = {
    "Accept": "application/json",
    "Content-Type": "Content-Type"
}
response = requests.request("POST", url,
data=payload, headers=headers)

print(response.text)
```

コード①：cURL と Python の token 取得の例

API 側を Python に書き直すことができたので、コード②で Spleeter 側を実装する。

```
separator = Separator('spleeter:2stems', 'wav')
separator.separate_to_file('./input/audio.wav',
'./audio_separated')
```

コード②：Spleeter による音声分離

コード②にて Spleeter 側に 2stems(vocal + other)の形で出力させることができたが、楽曲をそのまま使っているので、API 上で出力されたファイルを翻訳機に読み込ますことができなかった。そのため、Spleeter にて分けた音声ファイルのうち、音が無いまたはそれに等しい部分(≒声がない)を削除することを考えた。そこで、subprocess を利用して ffmpeg を使用することで音声の無音部分を削除できると考え、以下のコード③を考えた。

```
def cut_silent(movie, dB):
    os.chdir("../input")
    output = subprocess.run(["ffmpeg", "-i",  movie,
"-af", "silencedetect=noise={}dB:d=0.3".format(dB),
"-f", "null", "-"], stdout=subprocess.PIPE,
stderr=subprocess.PIPE)
    ss = str(output)
    lines = ss.replace("¥¥r", "")
    lines = lines.split('¥¥n')
    time_list = []
    for line in lines:
        if "silencedetect" in line:
         words = line.split(" ")
        for i in range(len(words)):
            if "silence_start" in words[i]:
                time_list.append(float(words[i+1]))
            if "silence_end" in words[i]:
                time_list.append(float(words[i + 1]))
    silence_section_list =
list(zip(*[iter(time_list)]*2))
    movie_name = movie.split(".")
    if str(silence_section_list[0][0]) != "0.0":
        split_file1 = "../output/" + movie_name[0] +
"_0" + ".mp4"
        subprocess.run(["ffmpeg", "-ss", str(0), "-i",
movie, "-t", str(silence_section_list[1][0]),
split_file1], stdout=subprocess.PIPE,
stderr=subprocess.PIPE)
```

コード③：無音部分の削除

無音部分を作成するにあたって、映像から一定の dB(デシベル)を下回る部分を削除するプログラムを作成した。これにより、無音部分を除いた、Vocal が存在する部分だけを API に渡すことが可能となった。この状態にて翻訳自体は完成した。

さらに、token のリサイクルを行うために、token 部分を変数化し、変数値を読み取り、過去に生成した token が使用できるかの判別を行うようにした。他のオーディオファイルの扱いにも同様の処理を行った。

他の工夫点として token 系列や payload を別ファイルにて一括管理した。

４．考察

翻訳機の実用性に関して、実際に歌詞に対して翻訳を行ってみたが、Spleeter を使用して”無理やり”声だけを抜き出しているので、翻訳精度は下がっていた。また、歌詞独特の言い回し等を翻訳すると、もともとの歌詞の意味が消えてしまうものもあり、実用性にはまだ至らないと考えた。

５．今後の展望

今後追加していきたい機能として、翻訳前の言語の特定、Spleeter の精度向上を目指したい。また、CUI は Android,IOS での実行は難しいので、今後 apk 形式も検討したいと思う。また、 後半で組んだものはまだ CUI でしか動作ができないので、今後 GUI に実装していこうと思う。

６．参考文献

[1] requests
(https://github.com/psf/requests)
[2] Spleeter
(https://github.com/deezer/spleeter)
[3] 音楽素材分離エンジン「Spleeter」
(https://note.com/field_mouse/n/nf2de7ed3cb09)
[4] 動画の無音部分を自動でカットする
(https://nantekottai.com/2020/06/14/video-cut-silence/)

７．謝辞

今回の研究を行うにあたり、NICT、fairydevices の研究者の皆様には多大なご指導を賜りました。深くお礼申し上げます。