**COMPARISON OF STT FPT.AI VS VIETTEL.AI**

*Luc Duong Hai,*

*Software Engineering FPT, FPT University*

*Hanoi, VietNam*

*Email:*[*haildhe141223@fpt.edu.vn,*](mailto:haildhe141223@fpt.edu.vn,)

*Nguyen Long Phuong*

*Software Engineering FPT, FPT University*

*Hanoi, VietNam*

*Email:*[*phuongnlhe141219@fpt.edu.vn*](mailto:phuongnlhe141219@fpt.edu.vn)

*Ha Duc Hanh*

*Software Engineering FPT, FPT University*

*Hanoi, VietNam*

*Email:*[*hanhhdhe141144@fpt.edu.vn,*](mailto:hanhhdhe141144@fpt.edu.vn,)

*Nguyen Huu Trung*

*Software Engineering FPT, FPT University*

*Hanoi, VietNam*

*Email:*[*trungnhhe141239@fpt.edu.vn*](mailto:trungnhhe141239@fpt.edu.vn)

*Abstract—Currently, Artificial Intelligence (AI) is very much concerned and applied to many different sectors, industries and life, especially natural language processing. In this article, our team will show how to process an audio file into a continuous audio clip by cutting out unnecessary silence, and pauses. After that use fpt.ai to convert audio to text. Thereby creating a program to help improve the quality of sound files up to 80% and music files up to 20%..*

*We used Fpt.ai API to detect audio and music after that compared with Viettelgroup.ai. The conclusion was found which one better included simple audio and music files.*

*Hopefully this article will help you have more ideas for developing natural language processing projects.*

Keywords—librosa, voice, silence, split, fpt.ai, viettelgroup.ai

# Introduction

In this project we will create a program that can download audio. Detect play, stop and analyzing the lyrics of audio file and music using commands on the linux operating system and python. After using our program, it will detect extractly silence, reduce almost noise, and also it can download song by original link.

Ability audio processing in Fpt.ai better than Viettelgroup.ai. In the other hand, Viettelgroup.ai has music processing better than Fpt.ai.

# Methodology and testing

## **Requirements**

In order to make a multiple continuous-speaking files without any silences, we need to prepare the following:

* Open  Google Colaboratory [1].
* Import necessary libraries includes librosa [2], IPython, noisereduce, ffmpeg, scipy, AudioSegment [3] …
* Using Fpt.ai [4] to detect a **.mp3** file then store the results to data texts .
* Comparison between (Fpt.ai) and (Viettelgroup.ai [5])

## **Main Idea**

Fig. 1. Flow chart

* Firstly, use urllib.request to download audio.
* Secondly, using InaSpeechSegmenter [6] to identify type of audio voice, and detect silences.
* Thirdly, reduce audio noise by using reducenoise [7] library.
* Fourthly, using API to send requests to fpt.ai and convert audio to texts.
* Finally, combine texts into one lyrics.txt.

## **Detail Process / algorithm**

**Step 1: Connect Colab with Google Drive**

from google.colab import drive

drive.mount('/content/drive')

**Step 2: Install necessary libraries**

#Install necessary libraries

**!pip install librosa**

**!pip install ffmpeg-python**

**!pip install noisereduce**

!apt -qq install -y sox

**!pip install pydub**

!pip install sox

!sudo apt-get install sox libsox-fmt-mp3

!apt-get install libsox-fmt-all sox libchromaprint-dev

**!pip install inaSpeechSegmenter**

**Step 3: Import libraries**

**import ffmpeg**

import urllib.request

**import librosa**

import matplotlib.pyplot as plt

import librosa.display

import IPython.display as ipd

**import noisereduce as nr**

import requests

**from inaSpeechSegmenter import Segmenter**

**from pydub import AudioSegment**

1. **The Functions of Libraries**

|  |  |
| --- | --- |
| **Libraries** | **Functions** |
| noisereduce | Noise reduction in python using spectral gating (speech, bioacoustics, time-domain signals) |
| librosa | A python package for music and audio analysis, create music information retrieval systems. |
| inaSpeechSegmenter | Split the audio signal into homogeneous zones of speech, music, and noise. Then detects speaker gender. |
| pydub | A library to manipulate audio data with a simple high-level interface. |
| ffmpeg | A very fast video and audio converter that can also grab from a live audio/video source. It can also convert between arbitrary sample rates and resize video on the fly with a high-quality polyphase filter. |

**Step 4: Download audio and show original audio file**

urllib.request.urlretrieve("https://media1.vocaroo.com/mp3/1o453JLtmsSa", "/content/drive/My Drive/Audio Folder/audio.mp3")

**#Create audio wave:**

x, sr = librosa.load('/content/drive/My Drive/Audio Folder/audio.mp3')

%matplotlib inline

plt.figure(figsize=(12, 4))

librosa.display.waveplot(x, sr=sr)

plt.title('Audio Wave')

plt.ylabel('Amplitude')

plt.xlabel('Time (Sec)')

ipd.Audio('/content/drive/My Drive/Audio Folder/audio.mp3')

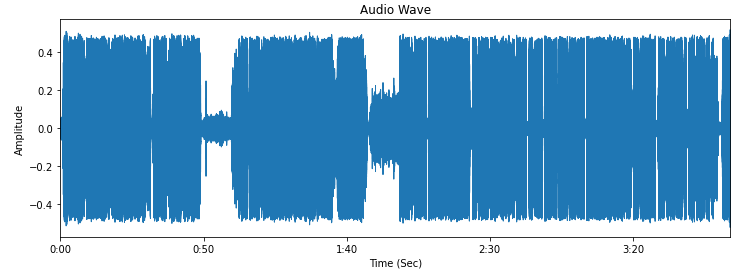


Fig. 2. Original audio

**Step 5: Using InaSpeechSegment to identified type of audio file and detect silences from**

**#Using InaSpeechSegmenter**:

media = '/content/drive/My Drive/Audio Folder/audio.mp3'

seg = Segmenter()

segmentation = seg(media)

print(segmentation)

**#DETECT SILENCE:**

sound\_file = AudioSegment.from\_mp3('/content/drive/My Drive/Audio Folder/audio.mp3')

# milliseconds in the sound track

# **ranges** define the start and the end times of each audio cut

ranges = [(0,50000),(60000,93000),(99780,107000),(118000,133800),(134000,163000),(163500,173500),(174000,218700),(219000,230000)]

count = 0

for x, y in ranges:

    new\_file = sound\_file[x : y]

    #output file

    new\_file.export("/content/drive/My Drive/Audio Folder/Audio/audio" + "" + str(count) +".mp3", format="mp3")

    count+=1

**Step 6: Reduce noise by Reduce noise libraries**

**#REDUCE NOISE:**

'''REDUCE NOISE ONE BY ONE

Because once audio files had diffirent noisies part so we have to detected one by one to get the good one'''

**#Load audio file:**

audio, sr = librosa.load('/content/drive/My Drive/Audio Folder/Audio/audio0.mp3')

**#Noise reduction audio0**

#noise\_part define the audio-frequency between 0 to 10Db

noisy\_part = audio[0:10000]

reduced\_noise = nr.reduce\_noise(audio\_clip=audio, noise\_clip=noisy\_part, verbose=False)

**#Check audio wave after reduce:**

librosa.display.waveplot(reduced\_noise, sr=sr)

plt.title('Reduced noise audio')

plt.ylabel('Amplitude')

plt.xlabel('Time (Sec)')

plt.show()

'''------------------------------------

OUTPUT GENERATOR:

    receives a destination path, file name, audio matrix, and sample rate,

    generates a wav file based on input

------------------------------------'''

def output\_file(destination ,filename, y, sr, ext=""):

    destination = destination + filename[:-4] + ext + '.mp3'

    librosa.output.write\_wav(destination, y, sr)

**#Generating output file**

output\_file(path\_name , 'audio0.mp3', reduced\_noise, sr, '\_reduced\_noise')

ipd.Audio(reduced\_noise, rate=sr)

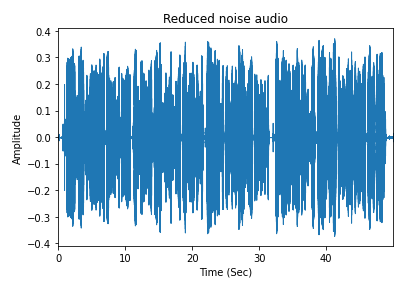


Fig. 3. Reduced Noise Audio

**Step 7: Sent request to Fpt.ai to convert audio to texts.**

#API FPT:

'''For somehow FPT.AI API (free API) will get limit rate to use then we have to using more codelines

'''--Once audio we will try once API

Anyway, if it's got limited. Don't worry. We prepared, below are some API keys we prepared, try it.

'''-- nYvfiwnSbtuKy0OK37W4aTVYaVzM3q3Q: Key 1

  zMMA0wOD4S0dlHNtIoULGickGb0ojvt1: Key 2

  GZhmL9nML2pJ8dncoXQ8IMLXvPnDQsHN

  aDStwoJ17CqKwfF1QZ3gxA4aCo0exMnk

  iedpov8BdCqIFWxXNr4LeAIIaAjcsfEo

  EdyezNlk7VhZEuiz3N7qGHyKZ5KtRg8Y---'''

# if not work.. Connect FPT.AI:

# Sign in an account then make your own API.

#Convert File 0

import requests

url = 'https://api.fpt.ai/hmi/asr/general'

payload = open('/content/drive/My Drive/Audio Folder/Audio Reduce/audio0\_reduced\_noise.mp3', 'rb').read()

headers = {

    'api-key': 'OhFLAJ7gpoPXIXQESHw3FowlrBRbjGio'

}

response = requests.post(url=url, data=payload, headers=headers)

print(response.json())

#for some how you have to try again if it not work

obj = response.json()

obj2 = obj['hypotheses']

listToStr = ' '.join([str(elem) for elem in obj2])

text = listToStr.split(':')

print(text[2])

**Step 8: Combine texts to one lyrics file, format .txt**

#Combine all data texts above to one lyrics.txt

import io, json

with io.open('/content/drive/My Drive/Audio Folder/lyrics.txt', 'w', encoding='utf-8') as f:

  f.write(json.dumps(text[2], ensure\_ascii=False))

 f.write(json.dumps(text1[2], ensure\_ascii=False))

 f.write(json.dumps(text2[2], ensure\_ascii=False))

 f.write(json.dumps(text3[2], ensure\_ascii=False))

 f.write(json.dumps(text4[2], ensure\_ascii=False))

 f.write(json.dumps(text5[2], ensure\_ascii=False))

 f.write(json.dumps(text6[2], ensure\_ascii=False))

 f.write(json.dumps(text7[2], ensure\_ascii=False))

# TESTING RESULT

It’s can be applied to many systems such as voice recognition, voice control and lyrics synchronization.

Result of sample output lyrics.txt:

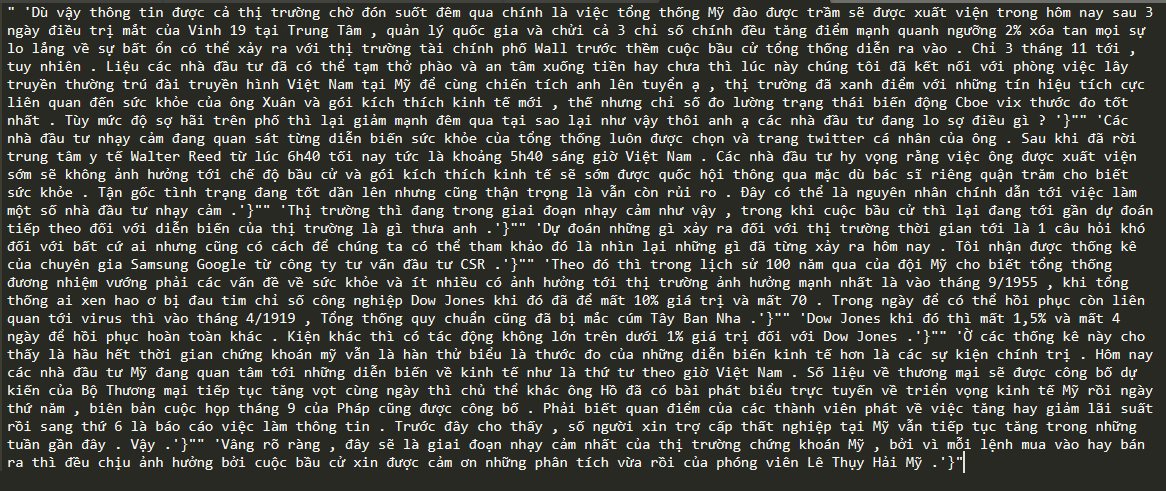


Fig. 4. Sample output lyrics

**Fig. 5. Sample output lyrics after translated by NH. Trung**

* **Quiet environment**: The program displays excellent sound with a bit noise and silience.
  + Ability processing in this environment is up to 80% with a simple audio file (not music file).
  + An audio file with more than 900 words after using Fpt.ai to detected, it has 827 words. And also we tried another one, audio with 560 words and once we used Fpt.ai to detected, it has 536 words (95,71%).
  + The accuracy of the first file is 795/827 words subequal 96% but it’s worked with simple audio, not music files.
  + The accuracy of the second file is 509/536 words subequal 94,96% also with simple audio.
  + The first audio file we trimmed to 7 files, the first trimmed file with 16 seconds and 63 words, after using Fpt.ai detected, we got 65 words (60/63 95,23%), and also Viettelgroup. ai detected too, we got 67 words (56/63 88,89%).

Fig. 6. Original audio cut lyrics

Fig. 7. Fpt.ai audio detect

Fig. 8. ViettelGroup audio detect

* + A music file has 273 words when detected stops and pauses. It trimmed to 7 files (20-40s) each. A trimmed file with 15s and 25 words. Fpt.ai detects 11 words (Accuracy 0%) and Viettel.ai detects 13 words (Accuracy 16%).

Fig. 9. Original music cut lyrics

Fig. 10. Viettelgroup.ai music detect

Fig. 11. Fpt.ai music detect

* **Noise stable environment**: The program displays fair sound without silience, sometimes the output contains tiny noises but it is trivial. Ability processing in noise stable environment is about 90%.
* **Noise unstable environment**: The program displays sound without silience but it still has noise because in this environment, noise level changes through time so the program can not get sample noise exactly in audio file to reduce noise . Ability processing in this environment is low, about 70%.
* **Summary**: This program performs well in quiet and noise stable environment. In noise unstable environment, program still works but it’s result contains unnessesary noise.

# Conclusion

In this article, we want to mention the audio processing function to create a complete file by removing unnecessary silence and noise in the audio file. Currently, this audio processing program is very interested and focused because it is applied to many areas of life such as: Speech recognition as know as (Speech recognition and then converting them into corresponding text), speech synthesis as know as (From an automatically synthesized text into speech), or text summary which is (From a long texts summarizes into just a shorter text as desired but still contains the most essential content ). In short, the above application is a very useful tool and my team hopes this program can be further improved and developed in the future.

# References

[1] E. Bisong and E. Bisong, “Google Colaboratory,” in *Building Machine Learning and Deep Learning Models on Google Cloud Platform*, 2019, pp. 59–64.

[2] and O. N. McFee, Brian, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, “librosa: Audio and music signal analysis in python,” *https://librosa.org/doc/latest/install.html*, 2013. https://groups.google.com/g/librosa.

[3] M. Strange, “AudioSegment,” 2018. https://github.com/MaxStrange/AudioSegment.

[4] T. D. Chung, M. Drieberg, M. F. Bin Hassan, and A. Khalyasmaa, “End-to-end Conversion Speed Analysis of an FPT.AI-based Text-to-Speech Application,” in *LifeTech 2020 - 2020 IEEE 2nd Global Conference on Life Sciences and Technologies*, 2020, pp. 136–139, doi: 10.1109/LifeTech48969.2020.1570620448.

[5] V. C. Center, “Speech Recognition,” 2018. https://viettelgroup.ai/.

[6] David.Doukhan., “Speech Segmentation Program,” 2018. https://github.com/ina-foss/inaSpeechSegmenter.

[7] T. Sainburg, “Noise reduction in python using spectral gating,” 2018. https://github.com/timsainb/noisereduce.