

HƯỚNG DẪN CHI TIẾT TRAIN TEXT TO SPEECH TỪ ĐẦU















* Lưu ý: Các thao tác được thực hiện trên ubuntu 20.04, có thể khác đôi chút trên hệ điều hành khác, bạn cần tinh chỉnh phù hợp.

B0: Chuẩn bị

- text để thu âm khoảng 50-80k câu, đánh số thứ tự cho các câu để phục vụ gán nhãn
- một người thu âm đọc các câu đó, Mic lọc nhiễu, phòng cách âm

B1: Gán nhãn

Sau khi thu âm thành các file sẽ có dạng xx-yy.m4a

	245002-24716.m4a	
	26811-26900.m4a	
	26571-26810.m4a	
	26201-26570.m4a	
	25916-26200.m4a	
	25707-25915.m4a	
	25562-25706.m4a	

Trong đó, xx là câu bắt đầu, yy là câu kết thúc, .m4a tùy vào thiết bị thu âm sẽ có các đuôi, cái này không ảnh hưởng nhiều lắm vì sau này chúng ta đều xử lý ở bước sau

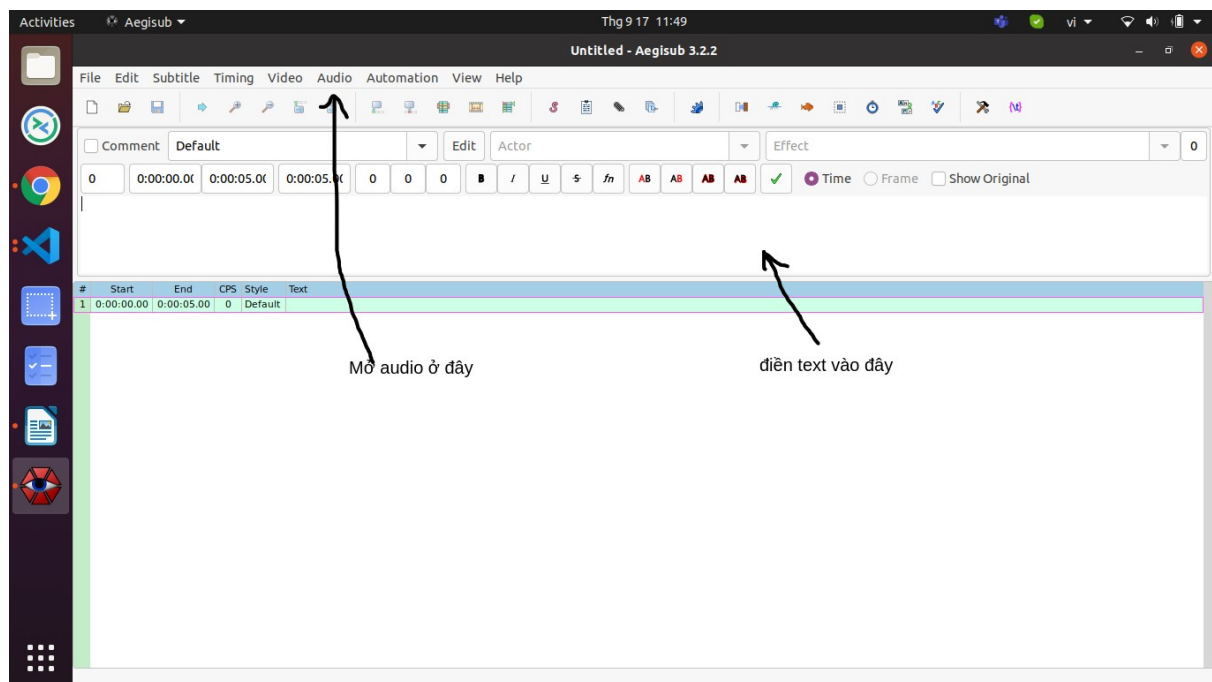
- Chuẩn bị phần mềm gán nhãn:

+ Aegisub (phần mềm để gán nhãn): <https://aegisub.en.uptodown.com/windows> (đây là bản cho windows, ubuntu có bản tương tự)

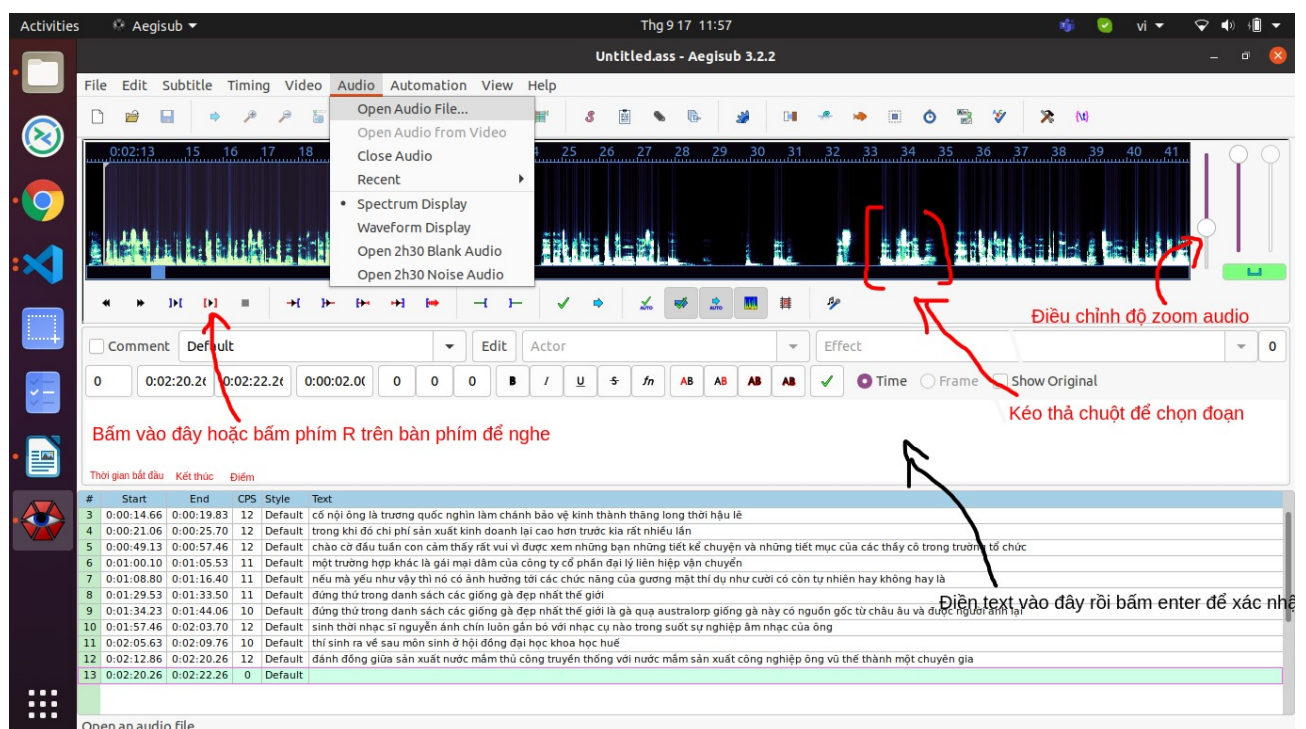
+ visual studio code: <https://code.visualstudio.com/Download> (Cái này không bắt buộc phải cài, mục đích là để mở file txt và gán nhãn cho tiện, bạn hoàn toàn có thể sử dụng phần mềm khác để mở file txt)

- Gán nhãn:

Khi mở giao diện phần mềm sẽ có dạng như này :



Để mở audio đã thu âm chọn: Audio → Open Audio File.. Sau đó tới thư mục chứa audio rồi open. Sau khi mở ta sẽ thấy giao diện giống hình dưới. Trong đó



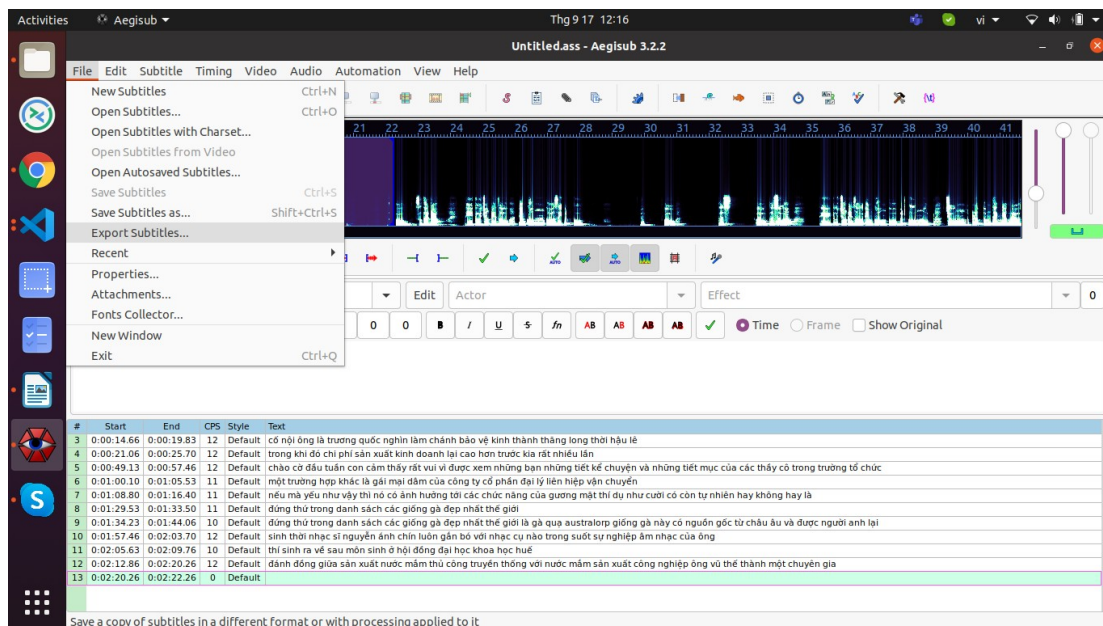
chú ý: cột CPS là điểm, thường bo càng đúng càng thấp, nhưng chỉ mang tính tương đối, thường điểm tốt < 15

* Lưu ý khi gán nhãn: Vì mục đích chọn ra đoạn audio có chất lượng âm thanh tốt và từ phải gán chính xác với câu đó (vì khi đọc người đọc có thể đọc khác đi với text đã chuẩn bị sẵn ví dụ như xuôi miệng hay text lủng củng) nên có vài gợi ý sau:

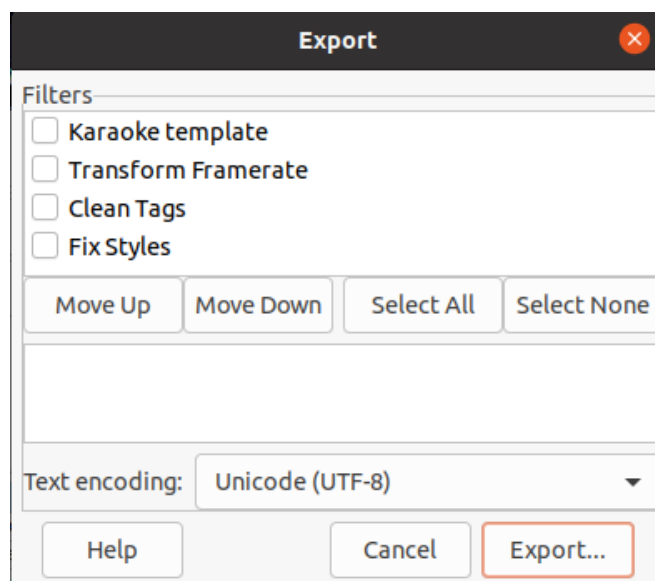
- Bỏ các đoạn có tiếng còi xe quá nhiều, tiếng ồn loạn xạ quá nhiều, các đoạn giọng ngắt nghỉ sai, nếu một câu ngắt sai ít thì có thể lấy, nếu ngắt sai nhiều quá cũng bỏ, có vài câu đọc cũng khác với text tương ứng thì sửa theo giọng đọc
- Đoạn nào audio bị dè chệ cũng bỏ

Còn file text ta mở bằng visual code bằng cách bấm chuột phải chọn Open with → visual studio code.

Sau khi gán nhãn xong một file, chọn File (ở góc trái trên màn hình) → Export subtitles...



Sau đó sẽ hiện lên cửa sổ như này (trên windows có thể khác đôi chút):



Để mặc định, chọn Export.. Sau đó lưu tên file giống như tên audio nhưng thay đuôi .m4a (hoặc .mp3 tùy thiết bị thu) bằng đuôi .srt :

Ví dụ : tên file audio: 1-99.m4a → tên file sau khi gán nhãn: 1-99.srt

Sau khi export xong file đuôi .srt, bạn chọn File → New Subtitles để thực hiện gán nhãn trên đoạn audio mới.

Trong quá trình gán nhãn, có thể có gián đoạn như uống nước, nghỉ, bạn có thể lưu lại project Aegisub đang làm dở bằng cách bấm ctrl + S để lúc sau có thể vào làm tiếp.(Lưu ý project aegisub khác với file đuôi srt bạn xuất ra nhé)

B2: Lọc nhiễu:

Lọc nhiễu là vấn đề quan trọng trong chuẩn bị dữ liệu, hiện tại mình có tìm được 2 cách lọc nhiễu, bạn có cách hay hơn hoàn toàn có thể góp ý và bổ sung.

C1: Lọc tự động (Lọc tương đối)

- <https://github.com/facebookresearch/denoiser>

Facebook research có hỗ trợ tool để lọc tự động nhiễu với độ chính xác chấp nhận được. Bạn có thể train model cho riêng mình hoặc dùng pre-trained model của họ.

Link paper: <https://arxiv.org/abs/2006.12847> trong trường hợp muốn tìm hiểu kỹ

- Các bước cài đặt thì bạn đọc readme là hiểu.

- Trong trường hợp dùng pre-trained model, họ chỉ hỗ trợ tần số 16kHz và 22kHz vậy nên trước khi dùng ta cần chuẩn hóa audio bằng cách sử dụng ffmpeg (lúc install các thư viện của facebook denoiser cũng bao gồm ffmpeg rồi):

```
ffmpeg -i test.wav -ac 1 -ar 16000 noisy/test_mono.wav
```

trong đó -i là input file

-ac channel audio

-ar tần số lấy mẫu

test_mono.wav là tên file output

Đây là cho 1 file, còn nhiều file chỉ cần viết một đoạn code python nhỏ là xong!
(check file norm_audio.py)

- Tiếp theo cho vào lọc tự động:

+ Nếu đoạn audio dài ta cần thêm tham số streaming:

```
%cd '/content/drive/MyDrive/denoiser/'
```

```
!pip3 install denoiser
```

```
!python3 -m denoiser.enhance --noisy_dir=noisy/test --out_dir=cleaned/test_stream --  
device="cuda" --sample_rate 16000 --streaming
```

Giải thích tham số:

--noisy_dir : thư mục chứa audio nhiễu

--out_dir : thư mục chứa audio lọc nhiễu

--device : có GPU thì để cuda

--sample_rate : tần số lấy mẫu audio

--streaming: với các audio dài, còn nếu input là các audio ngắn < 15s thì có thể bỏ tham số này đi

C2: Lọc thủ công

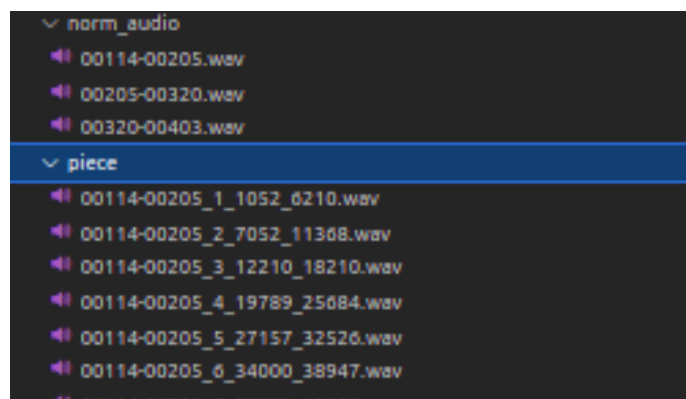
- Dù tool của facebook denoiser rất tuyệt nhưng ta chỉ không lọc được hoàn toàn, cần phải kiểm tra lại kỹ càng, lọc thủ công có thể dùng audacity hoặc các phần mềm tương đương để lọc (cái này đ/c Đức có thể làm tốt)

B3: Cắt audio thành các file nhỏ

Sau khi lọc nhiễu sạch sẽ, ta cắt audio bằng đoạn code python đơn giản sử dụng thêm thư viện pysrt và pydub.

(Check file cut_audio.py để thêm chi tiết)

Format một đoạn audio sau khi cắt có dạng:



Trong đó:

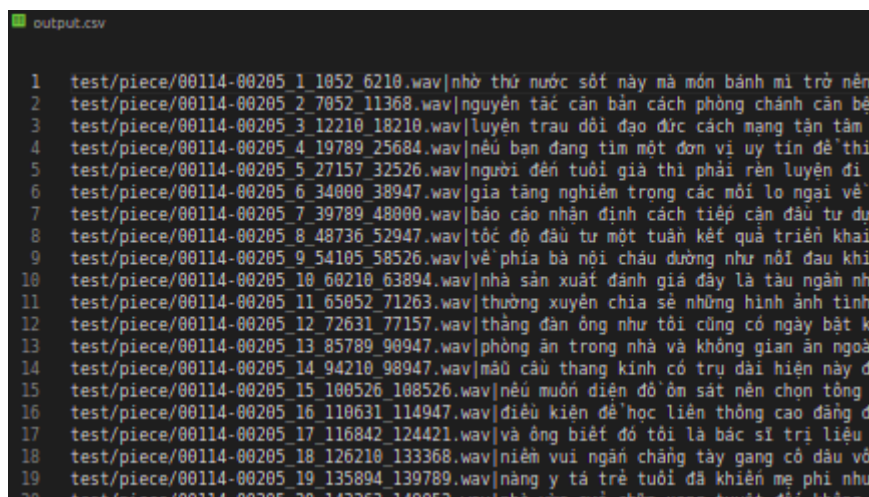
00114-00205: tên file gốc chưa cắt

1: số thứ tự

1052: thời gian bắt đầu trong file gốc, đơn vị (ms)

6210: thời gian kết thúc trong file gốc, đơn vị (ms)

Đồng thời cũng tạo ra file output.csv chứa câu với audio tương ứng vừa mới chia nhỏ ra



B4: Training Montreal Force Aligner để có được căn chỉnh chính xác từng từ một.

Tiếp theo ta cần sinh ra file .TextGrid để cho vào MFA training. Vậy, textGrid là gì ?

Nhìn chung file TextGrid sẽ có dạng như thế này, mỗi một từ sẽ có thời gian chính xác, ngày xưa người ta gán nhãn cho từng từ chứ không phải từng câu như bây giờ. Nhưng may mắn thay, giờ chúng ta có thể training cho từng từ một chính bằng Montreal Force Aligner (quá trình training hơi phức tạp. Nếu muốn tìm hiểu cụ thể, bạn có thể tìm tại đây:

<https://eleanorchodroff.com/tutorial/montreal-forced-aligner.html>

```
File type = "ooTextFile"
Object class = "TextGrid"

xmin = 0
xmax = 2.3
tiers? <exists>
size = 3
item []:
  item [1]:
    class = "IntervalTier"
    name = "Mary"
    xmin = 0
    xmax = 2.3
    intervals: size = 1
    intervals [1]:
      xmin = 0
      xmax = 2.3
      text = ""
  item [2]:
    class = "IntervalTier"
    name = "John"
    xmin = 0
    xmax = 2.3
    intervals: size = 1
    intervals [1]:
      xmin = 0
      xmax = 2.3
      text = ""
  item [3]:
    class = "TextTier"
    name = "bell"
    xmin = 0
    xmax = 2.3
    points: size = 0
```

Hình trên là output sau khi training, vậy input là gì ? Ta cần sử dụng:

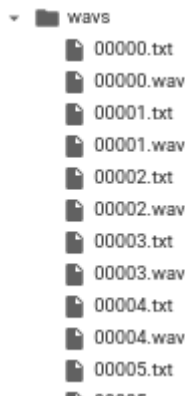
- file textGrid từ file text thu được
- chuẩn bị transcript (đã làm ở trên)
- một file từ điển
- acoustic model

- Tiếp đến ta cần tạo bộ từ điển. Chạy file get_word.py để lấy các từ trong câu, sau đó chạy file get_phonemes.py để tạo ra file lexicon.txt

- Khi đẩy lên drive ta để cây thư mục như thế này:

```
> . waves
  . install_mfa.sh
  . lexicon.txt
  . phonemes.txt
  . scripts.csv
  . words.txt
```

- Có 2 phần quan trọng:
- + file `lexicon.txt` vừa bên trên tạo được
- + thư mục `wavs` chứa audio file và text file tương ứng



- Bước tiếp theo là mang lên google colab để train MFA. Có một tips nhỏ là bạn zip file lại rồi đẩy lên google drive lưu trữ rồi dùng google colab unzip. Lý do là vì cơ chế sao lưu google gây hiện tượng trùng lặp hoặc mất file khi đẩy một số lượng lớn liên tục. Chạy file này để train:

<https://colab.research.google.com/gist/NTT123/c99b5a391af56e0cb8f7b190d3d7f0ee/infore-mfa-example.ipynb#scrollTo=uwlgRSd19lbK>

(Chú ý: notebook này họ cũng gen lexicon file như bước trên rồi, mình chỉ cần quan tâm đến cell cuối

```
!source /tmp/mfa/miniconda3/bin/activate aligner; mfa train --clean -C /content/wavs /content/lexicon.txt /content/InfoRe_Tg
```

thay đổi các file/folder phù hợp với google colab của mình:

`/content/wavs` : thư mục chứa corpus (gồm text + audio)

`/content/lexicon.txt` : file từ điển

`/content/InfoRe_Tg` : folder chứa textGrid sau khi train xong (mình thích để gì cũng được)

Sau khi training xong ta được kết quả như hình dưới:


```

1 File type = "ooTextFile"
2 Object class = "TextGrid"
3
4 xmin = 0
5 xmax = 5.2012
6 tiers? <exists>
7 size = 2
8 item []:
9   item [1]:
10     class = "IntervalTier"
11     name = "words"
12     xmin = 0
13     xmax = 5.2012
14     intervals: size = 23
15     intervals [1]:
16       xmin = 0
17       xmax = 0.33
18       text = ""
19     intervals [2]:
20       xmin = 0.33
21       xmax = 0.55
22       text = "hiện"
23     intervals [3]:
24       xmin = 0.55
25       xmax = 0.79
26       text = "nay"
27     intervals [4]:
28       xmin = 0.79
29       xmax = 0.95
30       text = "vị"
31     intervals [5]:
32       xmin = 0.95
33       xmax = 1.21
34       text = "tri"
35     intervals [6]:
36       xmin = 1.21
37       xmax = 1.36
38       text = "của"
39     intervals [7]:
40       xmin = 1.36
41       xmax = 1.59
42       text = "hàn"

```

```

item [2]:
  class = "IntervalTier"
  name = "phones"
  xmin = 0
  xmax = 5.2012
  intervals: size = 72
  intervals [1]:
    xmin = 0
    xmax = 0.33
    text = "sil"
  intervals [2]:
    xmin = 0.33
    xmax = 0.41
    text = "h"
  intervals [3]:
    xmin = 0.41
    xmax = 0.44
    text = "i"
  intervals [4]:
    xmin = 0.44
    xmax = 0.49
    text = "ê"
  intervals [5]:
    xmin = 0.49
    xmax = 0.55
    text = "n"
  intervals [6]:
    xmin = 0.55
    xmax = 0.63
    text = "n"
  intervals [7]:
    xmin = 0.63
    xmax = 0.71
    text = "a"
  intervals [8]:
    xmin = 0.71
    xmax = 0.79
    text = "y"

```

Như ta thấy MFA gán nhãn đến từng phoneme, rất tuyệt vời đúng ko :D

B5: Train duration model + acoustic model

Từ textGrid file thu được từ trên + audio, ta tiến hành train duration model:

<https://github.com/NTT123/vietTTS>

https://colab.research.google.com/drive/1oczrWOQOr1Y_qLdgis1twSINZlfPVXoY?usp=sharing#scrollTo=5sgn7TobiG2F

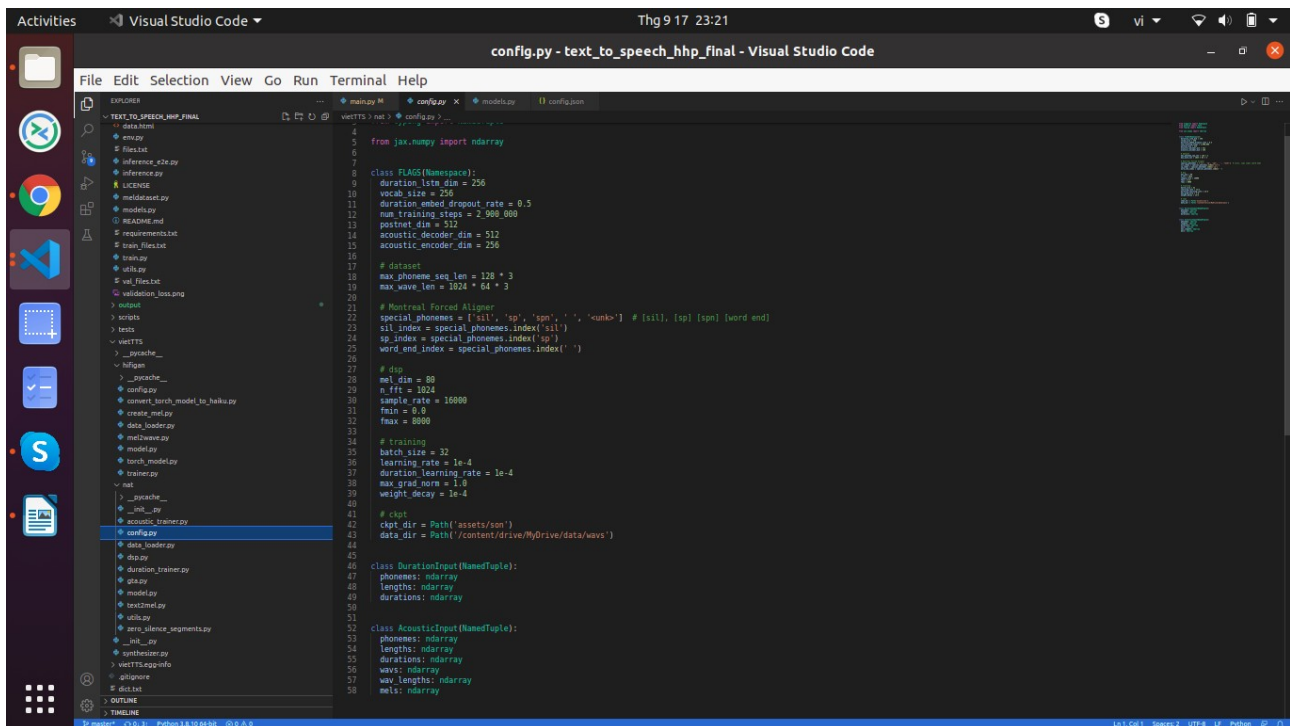
(file notebook chạy)

chạy cell sau để train :

!python3 -m vietTTS.nat.duration_trainer

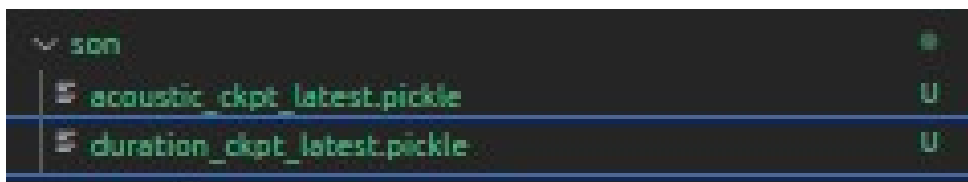
Mọi cấu hình ở trong file: vietTTS/nat/config.py

Tương tự sau khi train duration model, ta train acoustic model, cấu hình vẫn trong file config.py ở trên. Train acoustic model khá lâu thường ít nhất 1 tuần mới đạt chất lượng:



Hình: config.py (cái này mình gửi file nén cho anh Đoàn rồi)

Sau khi train xong duration + acoustic model, bạn đọc kỹ chính là Tacotron2 kết hợp MFA, sau bước này ta đã chuyển từ text → melSpectrogram. Ta cũng thu được 2 file weights trong thư mục: assets/son/acoustic_ckpt_latest.pickle và duration_ckpt_latest.pickle (Lưu ý đường dẫn có thể khác tùy vào lúc bạn cấu hình trên file config.py)



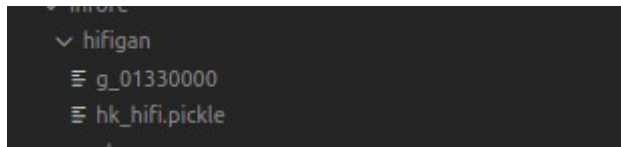
B6: train HiFiGAN vocoder:

HiFiGAN chuyển từ melSpectroGram → Audio. Link paper: <https://arxiv.org/abs/2010.05646>
repository git này cũng implement từ : <https://github.com/jik876/hifi-gan>

Train Vocoder:

Ta có thể train từ đầu hoặc fine tune tùy ý: Đọc readme của họ ghi rất rõ. Train vocoder cũng mất 7-10 ngày nữa, và check thường xuyên xem quá trình học như nào.

Sau khi train xong ta cũng thu được file weight:



g_01330000 chính là file weight. Với 1.300.000 steps training