Deep Learning
Deep Neural Network

Yoon Joong Kim,
Hanbat National University

음성합성

개념

Deep Learning 음성합성

Tacotron 2

Yoon Joong Kim

Department of Computer Engineering, Hanbat National University

yjkim@hanbat.ac.kr

1. 음성 합성(音聲合成, speech synthesis)

- 음성 합성(音聲合成, speech synthesis)
  - 말뭉치(Text)에 대하여 말소리의 음파(speech wave)를 기계가 자동으로 만들어 내는 기술, TTS(=Text-to-Speech)
  - 초기에,
    - 한 사람의 말소리를 녹음하여 일정한 음성 단위로 분할한 다음, 부호를 붙여 합성기에 입력하였다가 지시에 따라 필요한 음성 단위만을 다시 합쳐 말소리를 인위로 만들어내는 기술
    - 연결합성, 단위선택합성, Diphone 합성, 도메인별 합성, 포만트합성, 조음합성, HMM 기반합성, 사인파합성

https://en.wikipedia.org/wiki/Speech_synthesis

2. Deep learning 기반 음성합성

- WaveNet : 2016. 9. DeepMind
  - A deep generative model of raw audio waveforms
  - 딥러닝 기반모델은 음향학적 특징으로터 원시파형을 모델링 할 수 있다.
    - 음향학적 특징: 멜스케일 스펙트로그램 또는 스펙트로그램 뿐만 아니라 잘 처리된 언어학적 특징
- Chr2wav : 2017 Mila(연구소)
  - end-to-end model to produce raw waveform in an end-to-end method
    - 2017 년 초 Mila (연구소) 는 end-to-end 방식으로 원시 파형을 생성하는 모델 인 char2wav를 제안 했습니다 .
- Tacotron, VoiceLoop : 2017,Google 과 Facebook
  - Tacotron2
    - Google 은 WaveNet 보코더와 수정 된 Tacotron 아키텍처를 결합하여 종단 간 음성 합성을 수행하는 Tacotron2를 제안
    - Tacotron2는 사람의 목소리에 접근하는 고품질 음성을 생성 할 수 있다.

https://en.wikipedia.org/wiki/Speech_synthesis

3. Tacotron 2: Generating Human-like Speech from Text

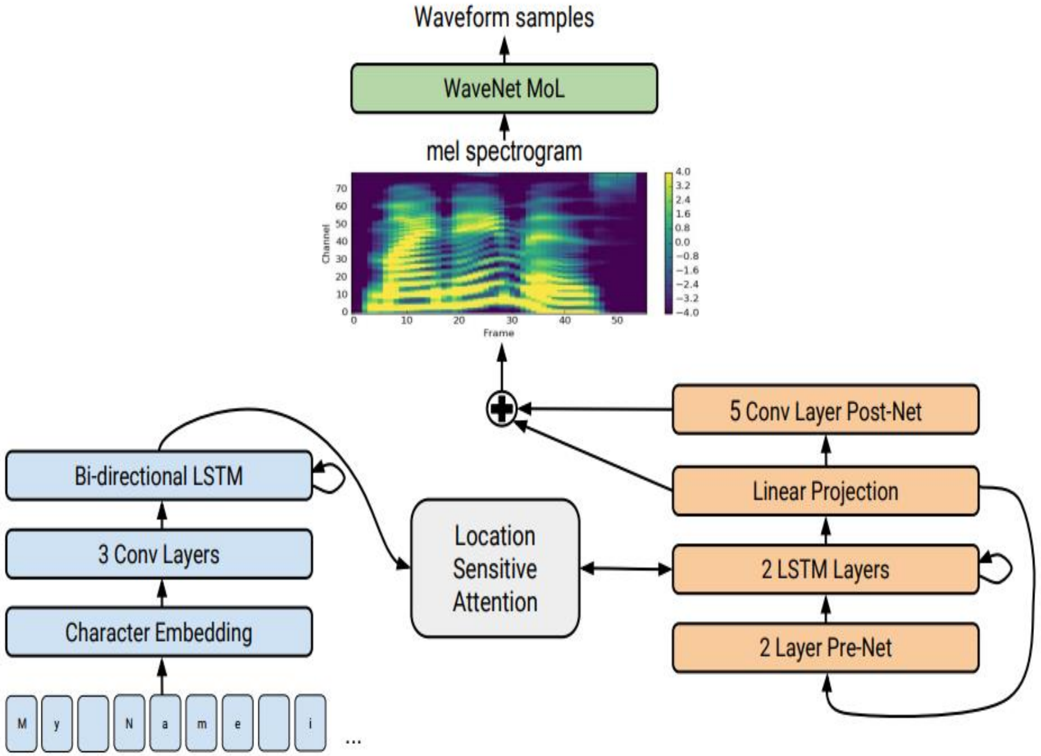
- Tacotron 2: Generating Human-like Speech from Text
    - Tuesday, December 19, 2017
    - Posted by Jonathan Shen and Ruoming Pang, Software Engineers, on behalf of the Google Brain and Machine Perception Teams
  - Tacotron + WaveNet,
  - “Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions”[link]
    - sequence-to-sequence model optimized for TTS to map a sequence of letters to a sequence of features that encode the audio.
    - an 80-dimensional audio spectrogram with frames computed every 12.5 milliseconds, capture not only pronunciation of words, but also various subtleties of human speech, including volume, speed and intonation. Finally these features are converted to a 24 kHz waveform using a WaveNet-like architecture.

https://ai.googleblog.com/2017/12/tacotron-2-generating-human-like-speech.html

3. Tacotron 2(cont.)

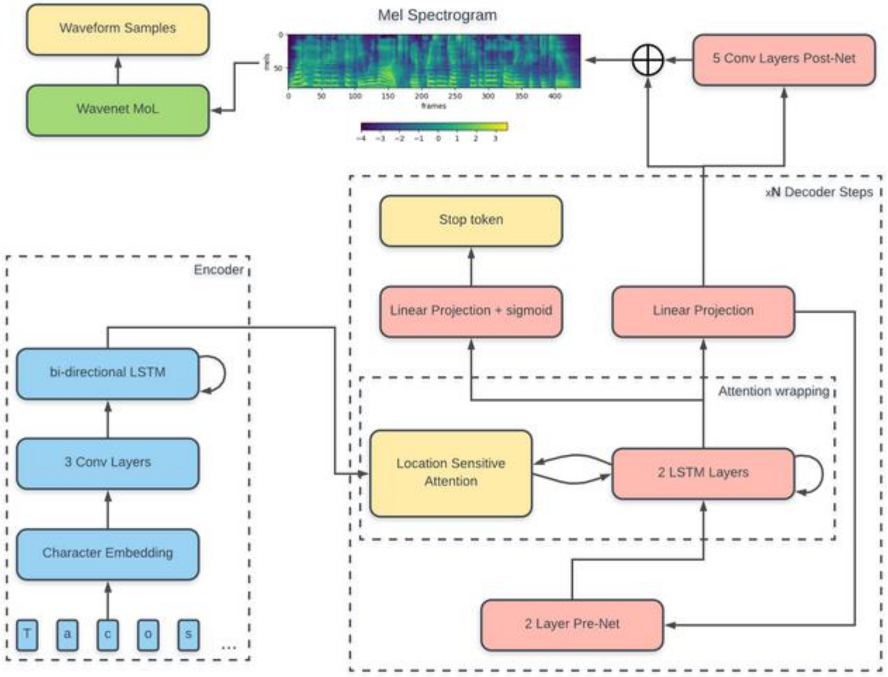
- You can listen to some of the Tacotron 2 audio samples that demonstrate the results of our state-of-the-art TTS system. In an evaluation where we asked human listeners to rate the naturalness of the generated speech, we obtained a score that was comparable to that of professional recordings.

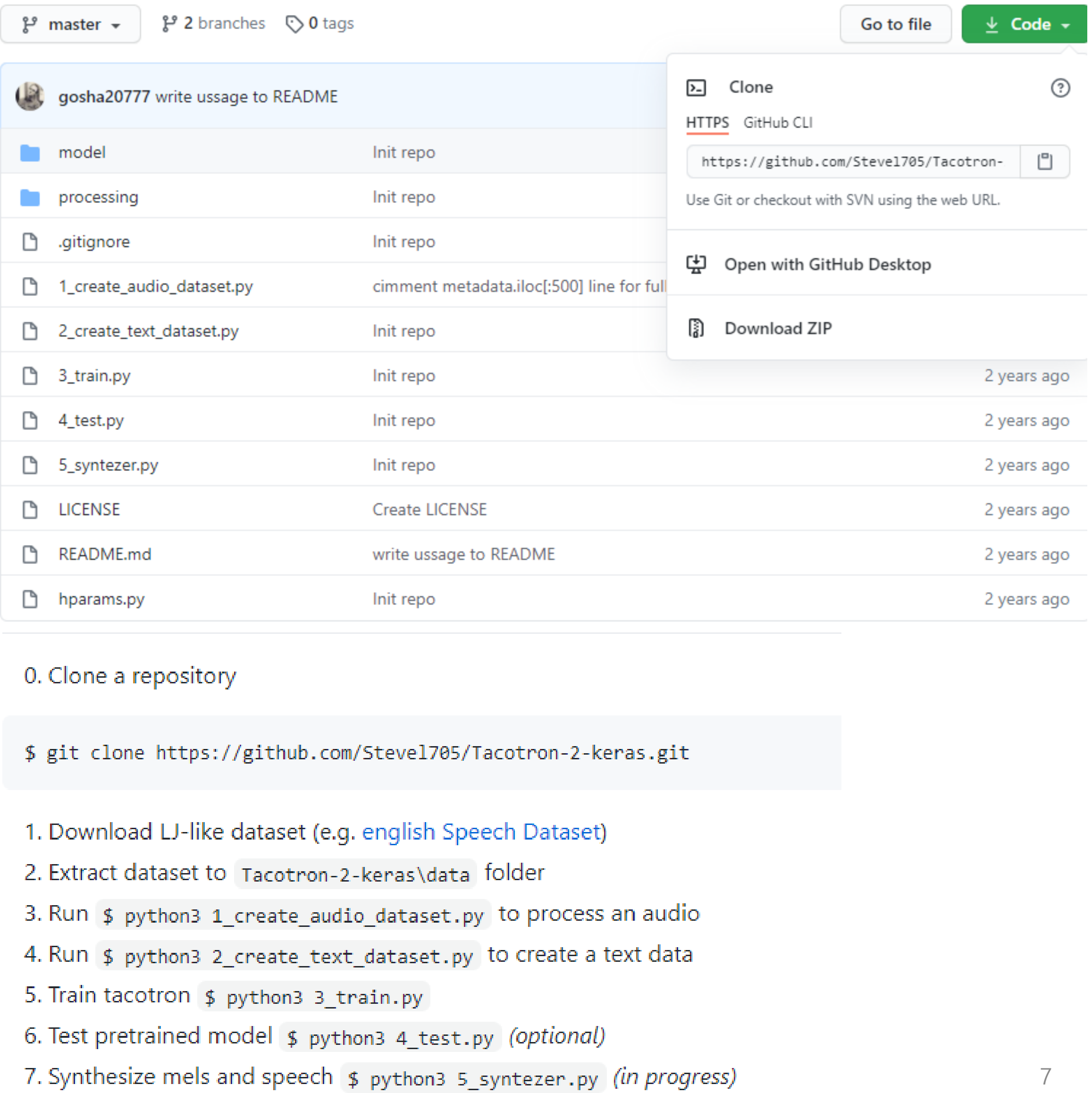
  While our samples sound great, there are still some difficult problems to be tackled. For example, our system has difficulties pronouncing complex words (such as "decorum" and "merlot"), and in extreme cases it can even randomly generate strange noises. Also, our system cannot yet generate audio in realtime. Furthermore, we cannot yet control the generated speech, such as directing it to sound happy or sad. Each of these is an interesting research problem on its own.



A detailed look at Tacotron 2's model architecture. The lower half of the image describes the sequence-to-sequence model that maps a sequence of letters to a spectrogram. For technical details, please refer to the paper.

4. Tacotron-2-keras (Without Wavenet vocoder)





gosha20777 write usage to README

https://github.com/Stevel705/Tacotron-

Use Git or checkout with SVN using the web URL.

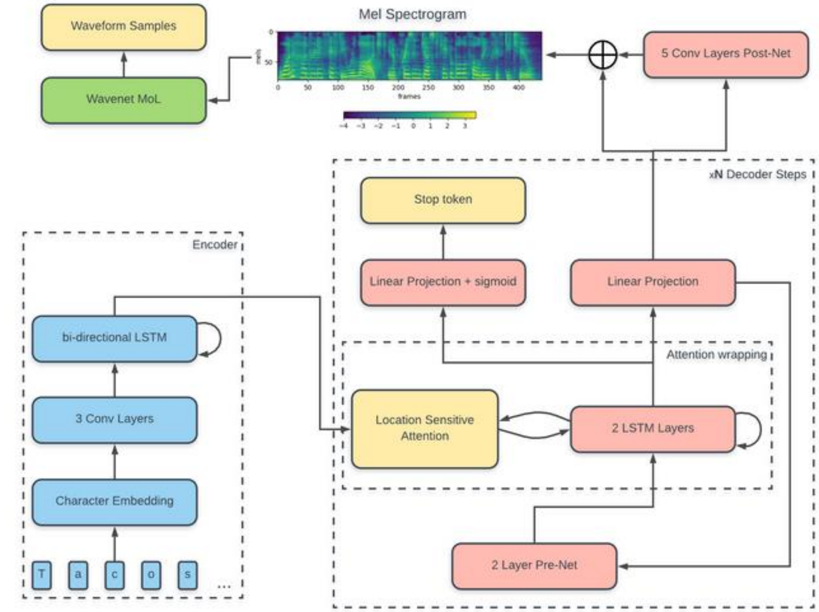
model	Init repo	
processing	Init repo	
.gitignore	Init repo	
1_create_audio_dataset.py	ciment metadata.iloc[:500] line for full	
2_create_text_dataset.py	Init repo	
3_train.py	Init repo	2 years ago
4_test.py	Init repo	2 years ago
5_syntezer.py	Init repo	2 years ago
LICENSE	Create LICENSE	2 years ago
README.md	write usage to README	2 years ago
hparams.py	Init repo	2 years ago

0. Clone a repository

```
$ git clone https://github.com/Stevel705/Tacotron-2-keras.git
```

1. Download LJ-like dataset (e.g. english Speech Dataset)
2. Extract dataset to `Tacotron-2-keras\data` folder
3. Run `$ python3 1_create_audio_dataset.py` to process an audio
4. Run `$ python3 2_create_text_dataset.py` to create a text data
5. Train tacotron `$ python3 3_train.py`
6. Test pretrained model `$ python3 4_test.py` *(optional)*
7. Synthesize mels and speech `$ python3 5_syntezer.py` *(in progress)*

https://github.com/Stevel705/Tacotron-2-keras



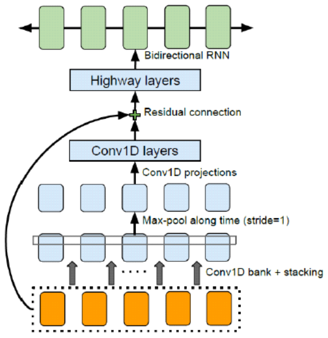
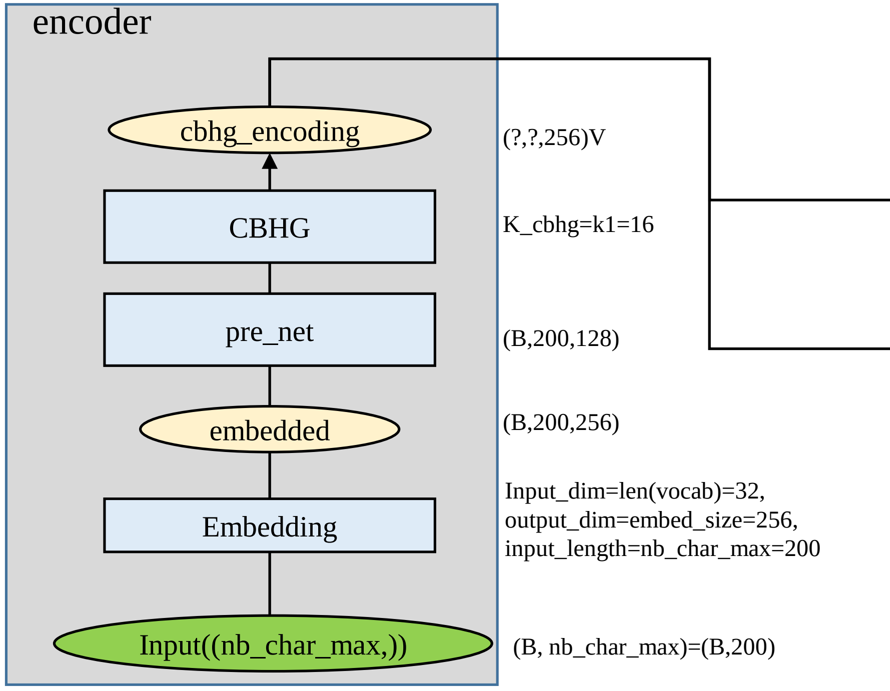


그림 2. CBHG 구조



encoder

cbhg_encoding (?,?,256)V

CBHG K_cbhg=k1=16

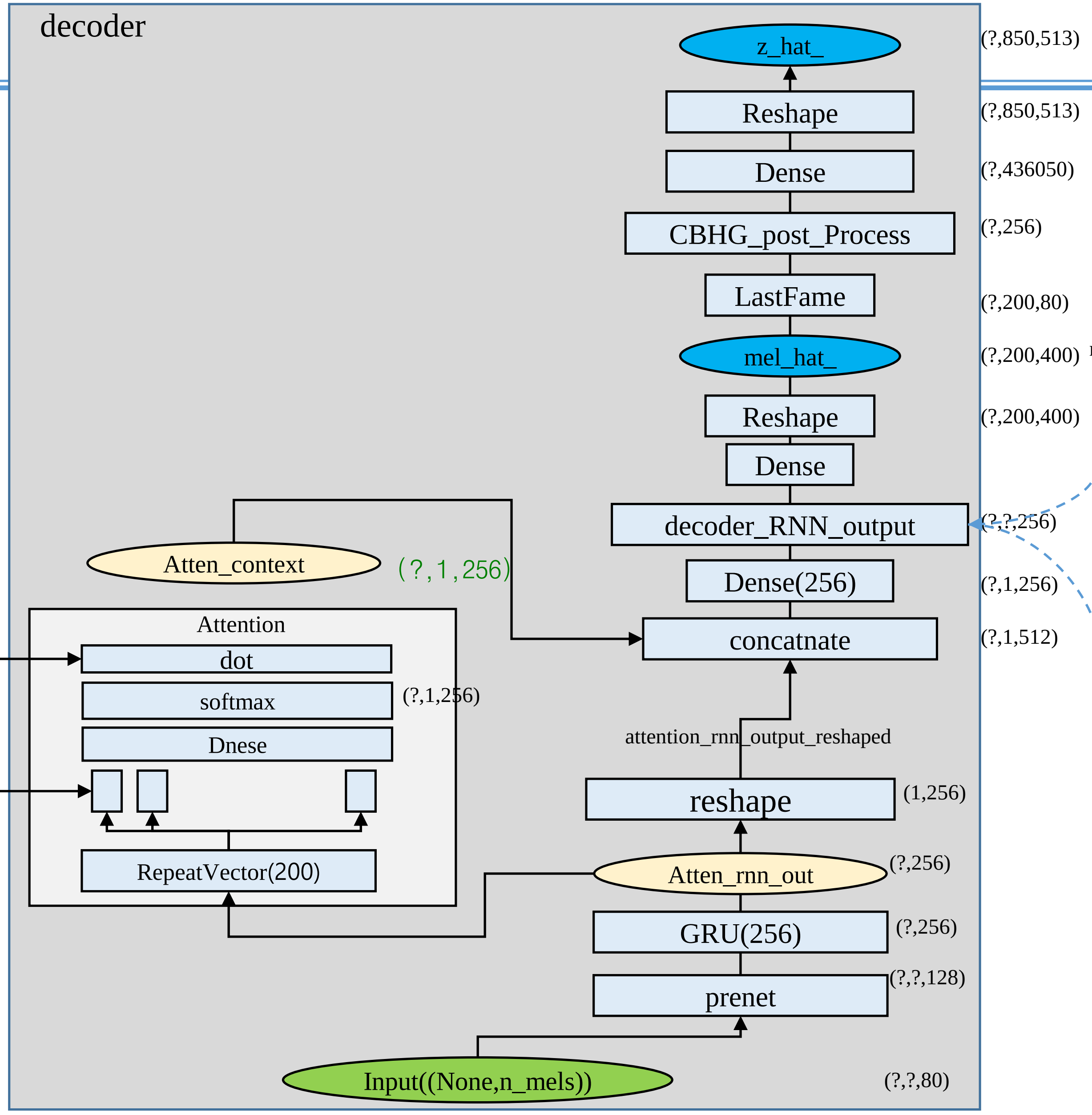
pre_net (B,200,128)

embedded (B,200,256)

Embedding Input_dim=len(vocab)=32,
output_dim=embed_size=256,
input_length=nb_char_max=200

Input((nb_char_max,)) (B, nb_char_max)=(B,200)

encoder input



decoder

z_hat_ (?,850,513)

Reshape (?,850,513)

Dense (?,436050)

CBHG_post_Process (?,256)

LastFame (?,200,80)

mel_hat_ (?,200,400) mel_spectro_training

Reshape (?,200,400)

Dense

decoder_RNN_output (?,?,256)

Atten_context (?,1,256)

Dense(256) (?,1,256)

concatnate (?,1,512)

Attention

dot

softmax (?,1,256)

Dnese

RepeatVector(200)

attention_rnn_output_reshaped

reshape (1,256)

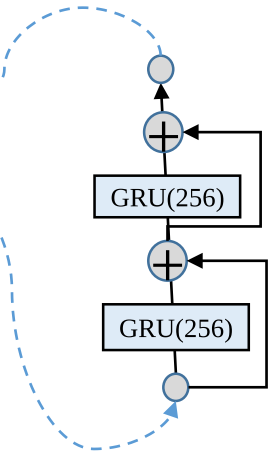
Atten_rnn_out (?,256)

GRU(256) (?,256)

prenet (?,?,128)

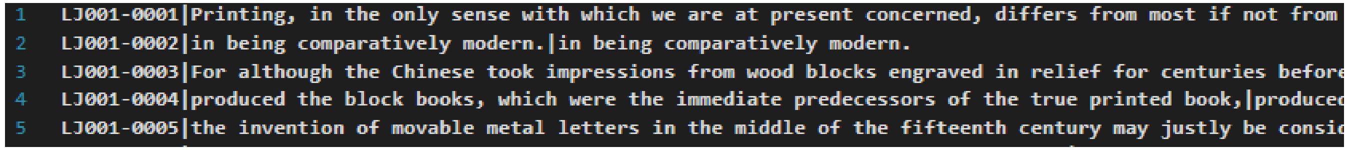
Input((None,n_mels)) (?,?,80)

(None,n_mels) decoder_input_training

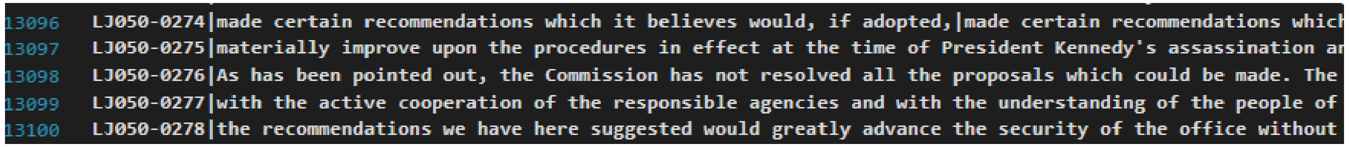


4.1 create_audio_dataset.py

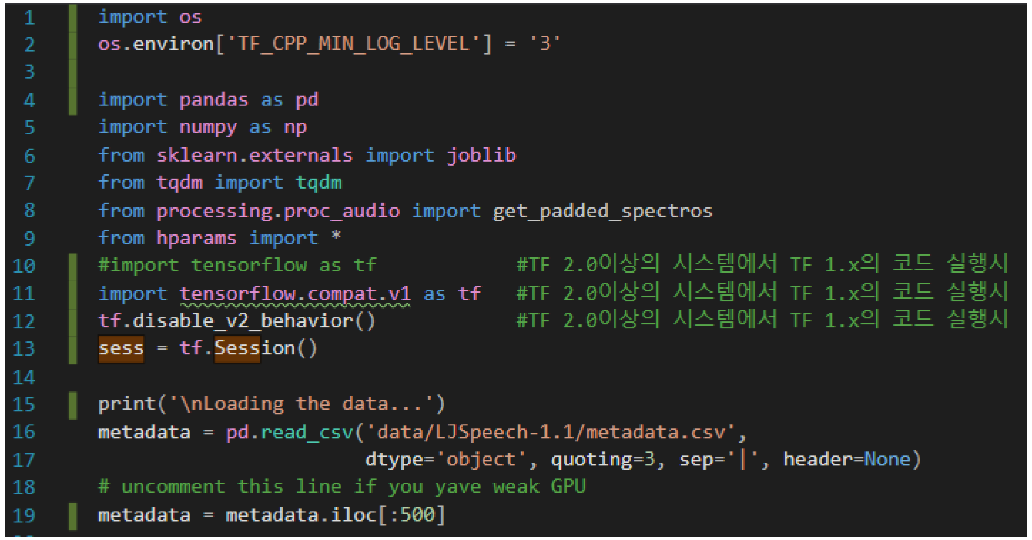
- Data/LJSpeech-1.1/metadata.csv
  - wav file => decoder_input, mel_spectro, spectro



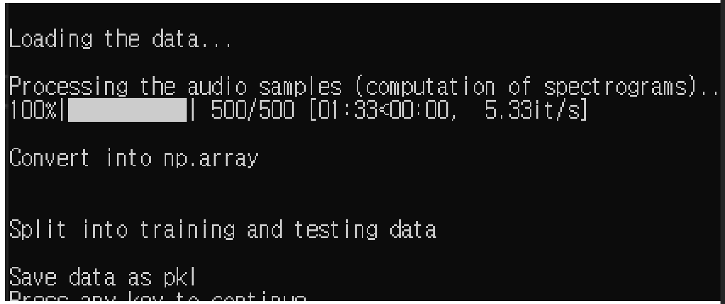
```
LJ001-0001|Printing, in the only sense with which we are at present concerned, differs from most if not from
LJ001-0002|in being comparatively modern.|in being comparatively modern.
LJ001-0003|For although the Chinese took impressions from wood blocks engraved in relief for centuries before
LJ001-0004|produced the block books, which were the immediate predecessors of the true printed book,|produced
LJ001-0005|the invention of movable metal letters in the middle of the fifteenth century may justly be consid
```



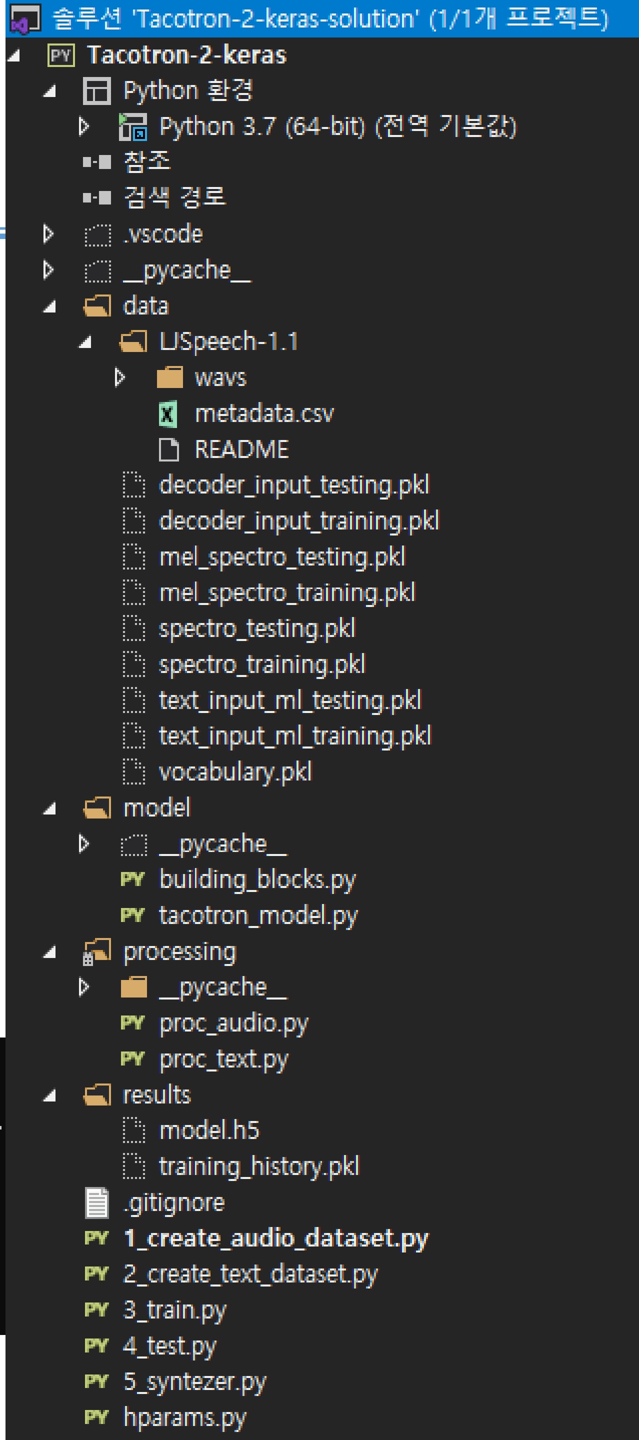
```
13096  LJ050-0274|made certain recommendations which it believes would, if adopted,|made certain recommendations which
13097  LJ050-0275|materially improve upon the procedures in effect at the time of President Kennedy's assassination ar
13098  LJ050-0276|As has been pointed out, the Commission has not resolved all the proposals which could be made. The
13099  LJ050-0277|with the active cooperation of the responsible agencies and with the understanding of the people of
13100  LJ050-0278|the recommendations we have here suggested would greatly advance the security of the office without
```



```python
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'

import pandas as pd
import numpy as np
from sklearn.externals import joblib
from tqdm import tqdm
from processing.proc_audio import get_padded_spectros
from hparams import *
#import tensorflow as tf          #TF 2.0이상의 시스템에서 TF 1.x의 코드 실행시
import tensorflow.compat.v1 as tf   #TF 2.0이상의 시스템에서 TF 1.x의 코드 실행시
tf.disable_v2_behavior()            #TF 2.0이상의 시스템에서 TF 1.x의 코드 실행시
sess = tf.Session()

print('\nLoading the data...')
metadata = pd.read_csv('data/LJSpeech-1.1/metadata.csv',
                       dtype='object', quoting=3, sep='|', header=None)
# uncomment this line if you yave weak GPU
metadata = metadata.iloc[:500]
```



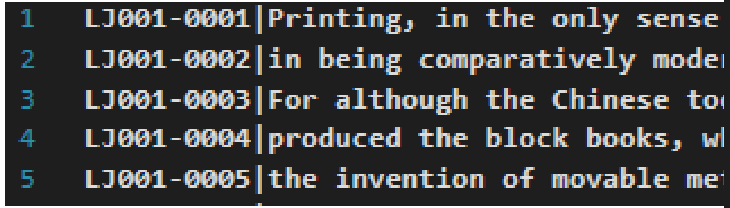
```
Loading the data...

Processing the audio samples (computation of spectrograms)..
100%|██████████| 500/500 [01:33<00:00,  5.33it/s]

Convert into np.array


Split into training and testing data

Save data as pkl
Press any key to continue
```



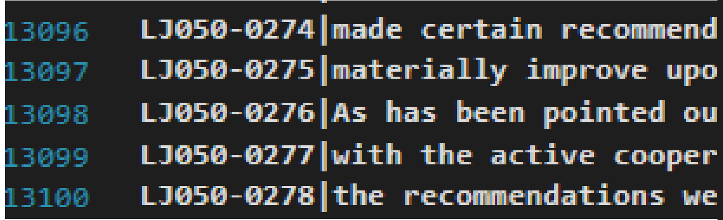
```
솔루션 'Tacotron-2-keras-solution' (1/1개 프로젝트)
  Tacotron-2-keras
    Python 환경
      Python 3.7 (64-bit) (전역 기본값)
    참조
    검색 경로
    .vscode
    __pycache__
    data
      LJSpeech-1.1
        wavs
        metadata.csv
        README
      decoder_input_testing.pkl
      decoder_input_training.pkl
      mel_spectro_testing.pkl
      mel_spectro_training.pkl
      spectro_testing.pkl
      spectro_training.pkl
      text_input_ml_testing.pkl
      text_input_ml_training.pkl
      vocabulary.pkl
    model
      __pycache__
      building_blocks.py
      tacotron_model.py
    processing
      __pycache__
      proc_audio.py
      proc_text.py
    results
      model.h5
      training_history.pkl
    .gitignore
    1_create_audio_dataset.py
    2_create_text_dataset.py
    3_train.py
    4_test.py
    5_syntezer.py
    hparams.py
```

4.1 create_au

- Data/LJSpeech-1.1/me
  - wav file => decoder_input, mel_spectro, spectro

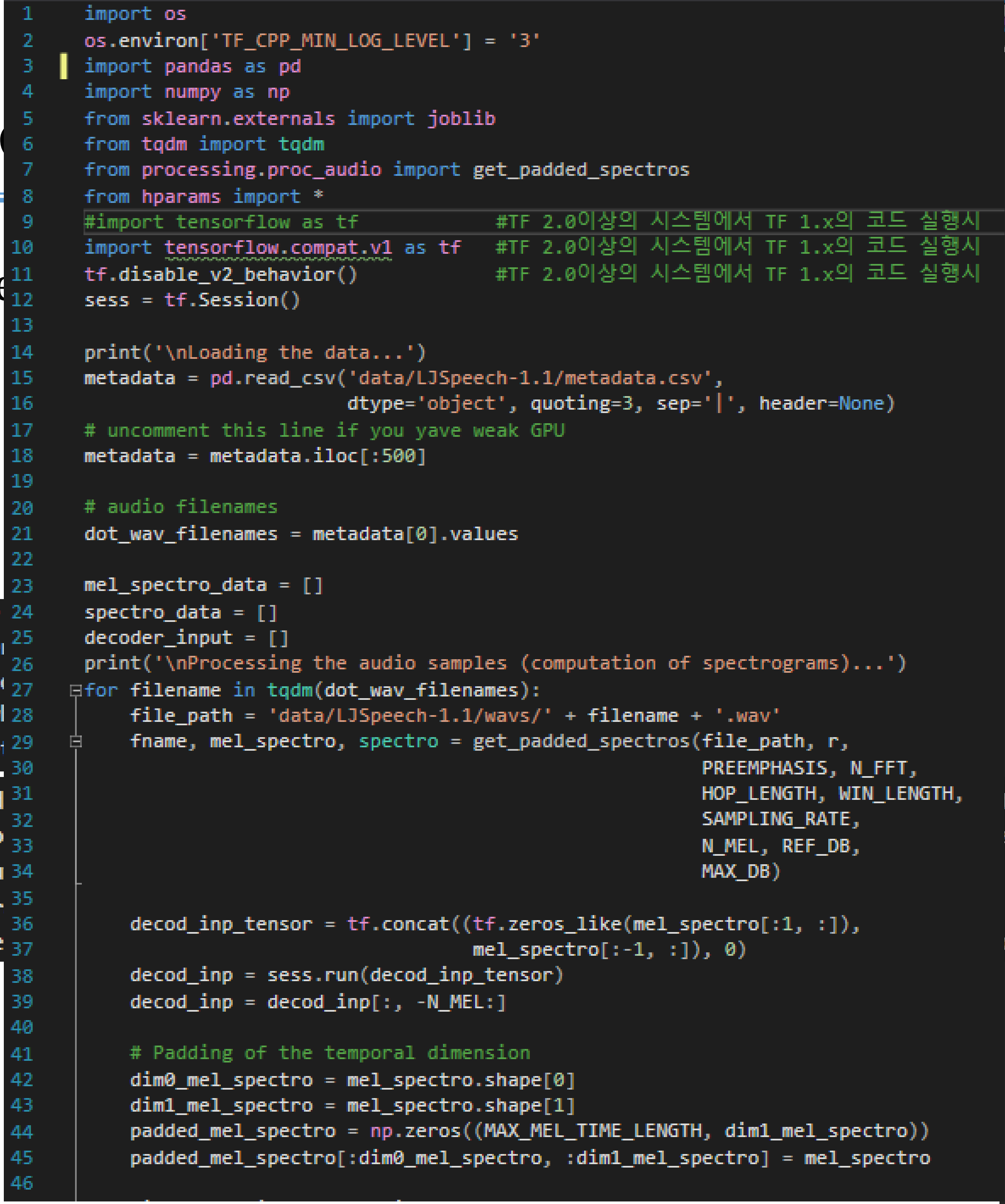


```
LJ001-0001|Printing, in the only sense
LJ001-0002|in being comparatively mode
LJ001-0003|For although the Chinese to
LJ001-0004|produced the block books, wl
LJ001-0005|the invention of movable me
```

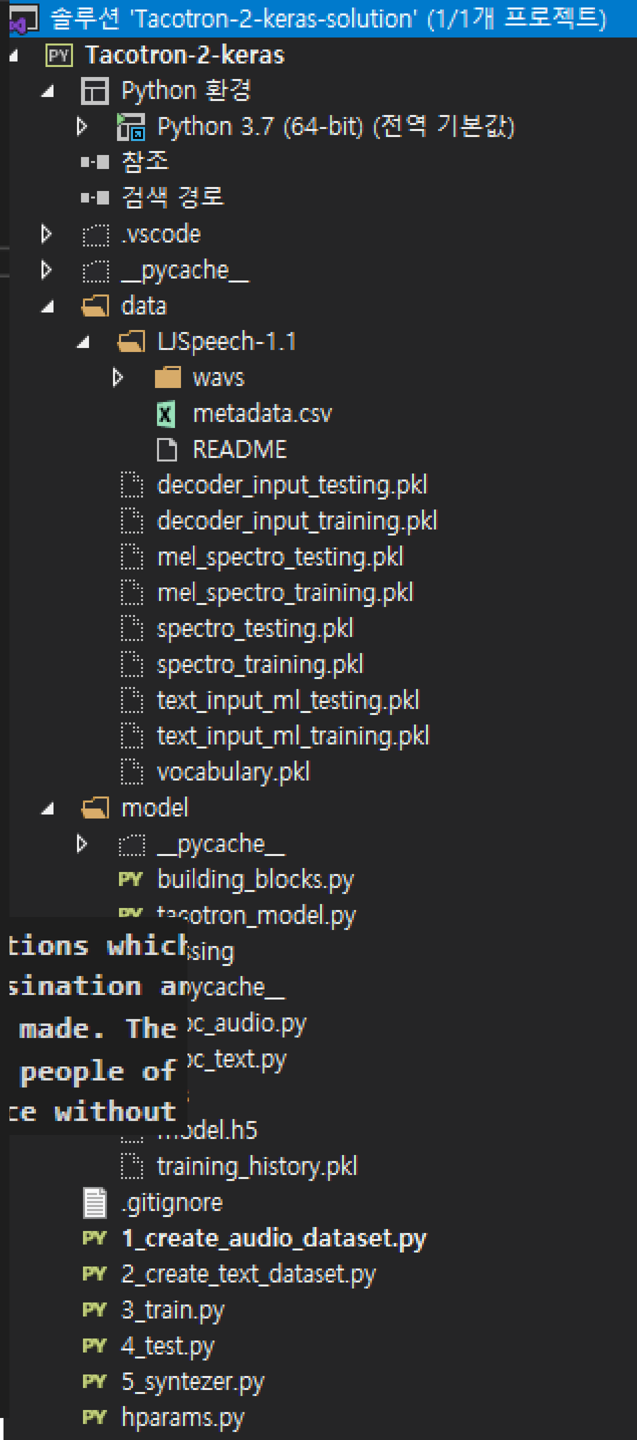


```
13096   LJ050-0274|made certain recommend
13097   LJ050-0275|materially improve upo
13098   LJ050-0276|As has been pointed ou
13099   LJ050-0277|with the active cooper
13100   LJ050-0278|the recommendations we
```

```
tions which
sination an
 made. The
 people of
ce without
```



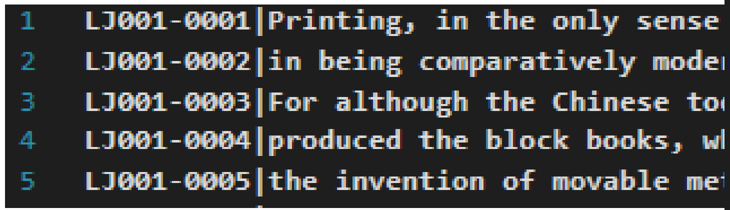
```python
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
import pandas as pd
import numpy as np
from sklearn.externals import joblib
from tqdm import tqdm
from processing.proc_audio import get_padded_spectros
from hparams import *
#import tensorflow as tf           #TF 2.0이상의 시스템에서 TF 1.x의 코드 실행시
import tensorflow.compat.v1 as tf  #TF 2.0이상의 시스템에서 TF 1.x의 코드 실행시
tf.disable_v2_behavior()           #TF 2.0이상의 시스템에서 TF 1.x의 코드 실행시
sess = tf.Session()

print('\nLoading the data...')
metadata = pd.read_csv('data/LJSpeech-1.1/metadata.csv',
                       dtype='object', quoting=3, sep='|', header=None)
# uncomment this line if you yave weak GPU
metadata = metadata.iloc[:500]

# audio filenames
dot_wav_filenames = metadata[0].values

mel_spectro_data = []
spectro_data = []
decoder_input = []
print('\nProcessing the audio samples (computation of spectrograms)...')
for filename in tqdm(dot_wav_filenames):
    file_path = 'data/LJSpeech-1.1/wavs/' + filename + '.wav'
    fname, mel_spectro, spectro = get_padded_spectros(file_path, r,
                                                      PREEMPHASIS, N_FFT,
                                                      HOP_LENGTH, WIN_LENGTH,
                                                      SAMPLING_RATE,
                                                      N_MEL, REF_DB,
                                                      MAX_DB)

    decod_inp_tensor = tf.concat((tf.zeros_like(mel_spectro[:1, :]),
                                  mel_spectro[:-1, :]), 0)
    decod_inp = sess.run(decod_inp_tensor)
    decod_inp = decod_inp[:, -N_MEL:]

    # Padding of the temporal dimension
    dim0_mel_spectro = mel_spectro.shape[0]
    dim1_mel_spectro = mel_spectro.shape[1]
    padded_mel_spectro = np.zeros((MAX_MEL_TIME_LENGTH, dim1_mel_spectro))
    padded_mel_spectro[:dim0_mel_spectro, :dim1_mel_spectro] = mel_spectro
```



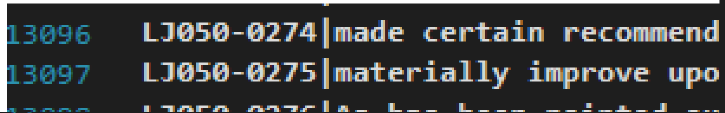
```
솔루션 'Tacotron-2-keras-solution' (1/1개 프로젝트)
  Tacotron-2-keras
    Python 환경
      Python 3.7 (64-bit) (전역 기본값)
    참조
    검색 경로
    .vscode
    __pycache__
    data
      LJSpeech-1.1
        wavs
        metadata.csv
        README
      decoder_input_testing.pkl
      decoder_input_training.pkl
      mel_spectro_testing.pkl
      mel_spectro_training.pkl
      spectro_testing.pkl
      spectro_training.pkl
      text_input_ml_testing.pkl
      text_input_ml_training.pkl
      vocabulary.pkl
    model
      __pycache__
      building_blocks.py
      tacotron_model.py
    sing
      ycache__
      c_audio.py
      c_text.py
      odel.h5
      training_history.pkl
    .gitignore
    1_create_audio_dataset.py
    2_create_text_dataset.py
    3_train.py
    4_test.py
    5_syntezer.py
    hparams.py
```

4.1 create_au

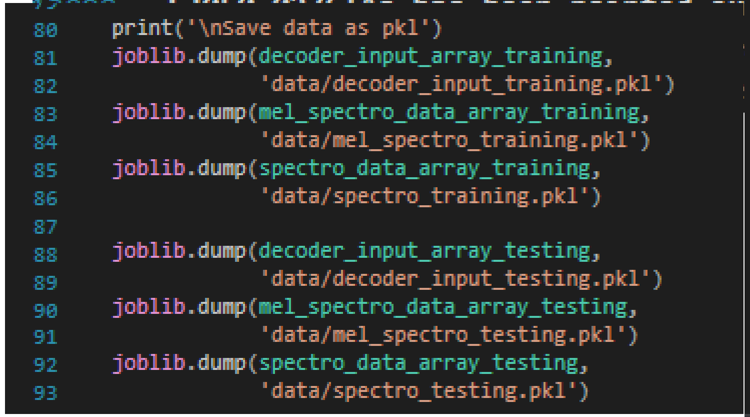
- Data/LJSpeech-1.1/me
  - wav file => decoder_input, mel_spectro, spectro



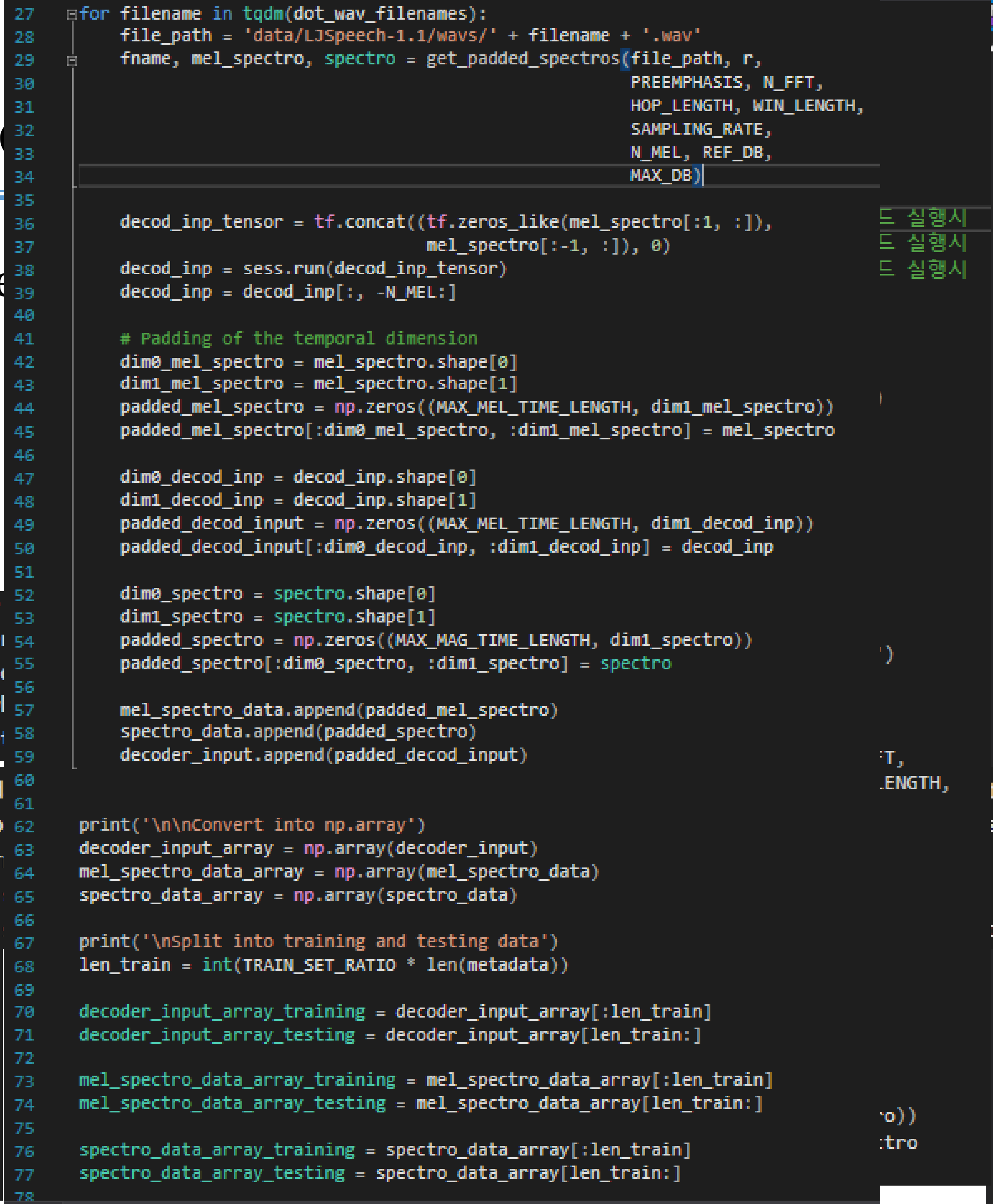
```
LJ001-0001|Printing, in the only sense
LJ001-0002|in being comparatively mode
LJ001-0003|For although the Chinese to
LJ001-0004|produced the block books, wl
LJ001-0005|the invention of movable me
```



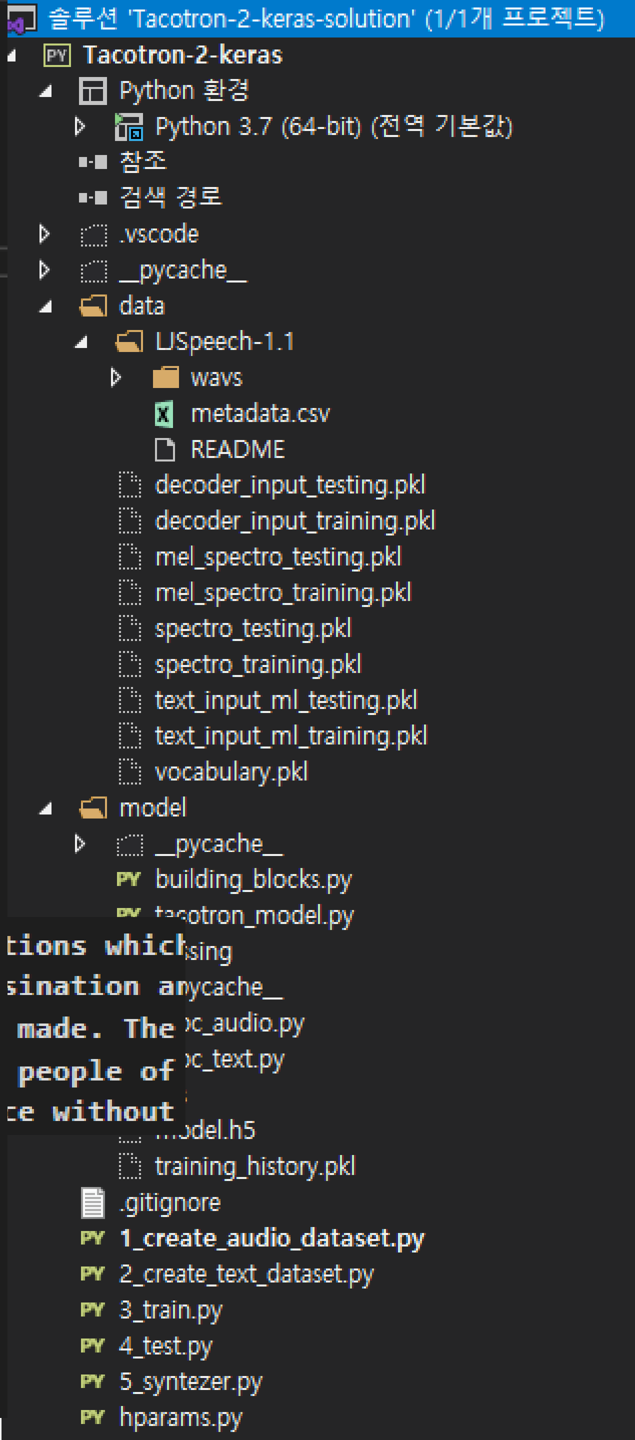
```
13096    LJ050-0274|made certain recommend
13097    LJ050-0275|materially improve upo
```



```
    print('\nSave data as pkl')
    joblib.dump(decoder_input_array_training,
                'data/decoder_input_training.pkl')
    joblib.dump(mel_spectro_data_array_training,
                'data/mel_spectro_training.pkl')
    joblib.dump(spectro_data_array_training,
                'data/spectro_training.pkl')

    joblib.dump(decoder_input_array_testing,
                'data/decoder_input_testing.pkl')
    joblib.dump(mel_spectro_data_array_testing,
                'data/mel_spectro_testing.pkl')
    joblib.dump(spectro_data_array_testing,
                'data/spectro_testing.pkl')
```

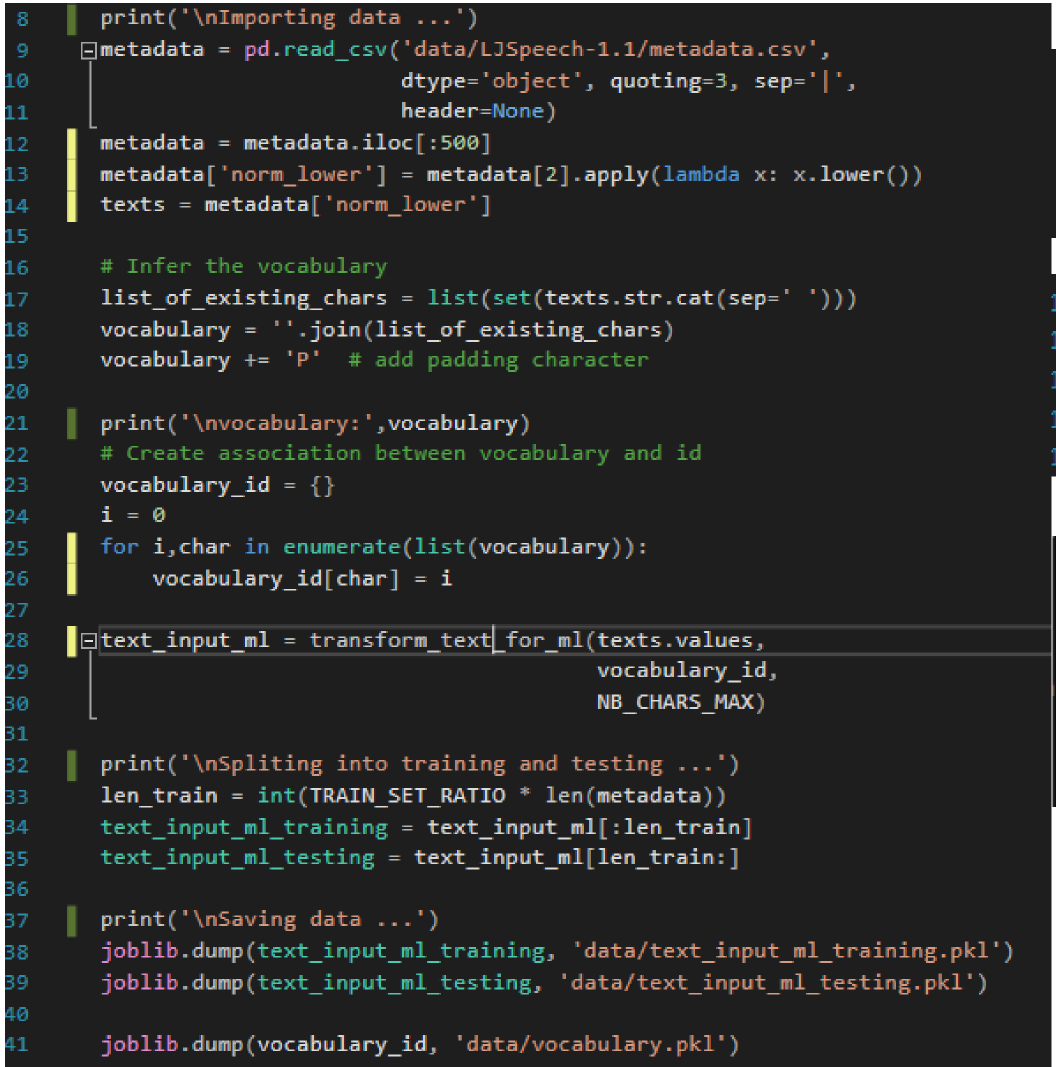


```
for filename in tqdm(dot_wav_filenames):
    file_path = 'data/LJSpeech-1.1/wavs/' + filename + '.wav'
    fname, mel_spectro, spectro = get_padded_spectros(file_path, r,
                                                       PREEMPHASIS, N_FFT,
                                                       HOP_LENGTH, WIN_LENGTH,
                                                       SAMPLING_RATE,
                                                       N_MEL, REF_DB,
                                                       MAX_DB)

    decod_inp_tensor = tf.concat((tf.zeros_like(mel_spectro[:1, :]),
                                  mel_spectro[:-1, :]), 0)
    decod_inp = sess.run(decod_inp_tensor)
    decod_inp = decod_inp[:, -N_MEL:]

    # Padding of the temporal dimension
    dim0_mel_spectro = mel_spectro.shape[0]
    dim1_mel_spectro = mel_spectro.shape[1]
    padded_mel_spectro = np.zeros((MAX_MEL_TIME_LENGTH, dim1_mel_spectro))
    padded_mel_spectro[:dim0_mel_spectro, :dim1_mel_spectro] = mel_spectro

    dim0_decod_inp = decod_inp.shape[0]
    dim1_decod_inp = decod_inp.shape[1]
    padded_decod_input = np.zeros((MAX_MEL_TIME_LENGTH, dim1_decod_inp))
    padded_decod_input[:dim0_decod_inp, :dim1_decod_inp] = decod_inp

    dim0_spectro = spectro.shape[0]
    dim1_spectro = spectro.shape[1]
    padded_spectro = np.zeros((MAX_MAG_TIME_LENGTH, dim1_spectro))
    padded_spectro[:dim0_spectro, :dim1_spectro] = spectro

    mel_spectro_data.append(padded_mel_spectro)
    spectro_data.append(padded_spectro)
    decoder_input.append(padded_decod_input)


print('\n\nConvert into np.array')
decoder_input_array = np.array(decoder_input)
mel_spectro_data_array = np.array(mel_spectro_data)
spectro_data_array = np.array(spectro_data)

print('\nSplit into training and testing data')
len_train = int(TRAIN_SET_RATIO * len(metadata))

decoder_input_array_training = decoder_input_array[:len_train]
decoder_input_array_testing = decoder_input_array[len_train:]

mel_spectro_data_array_training = mel_spectro_data_array[:len_train]
mel_spectro_data_array_testing = mel_spectro_data_array[len_train:]

spectro_data_array_training = spectro_data_array[:len_train]
spectro_data_array_testing = spectro_data_array[len_train:]
```

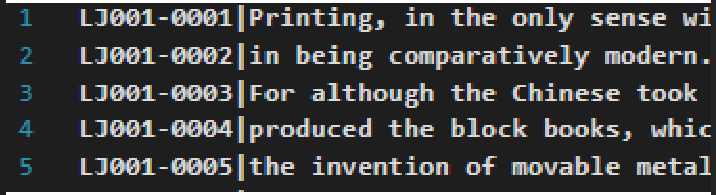


4.2 create_text_dataset.py

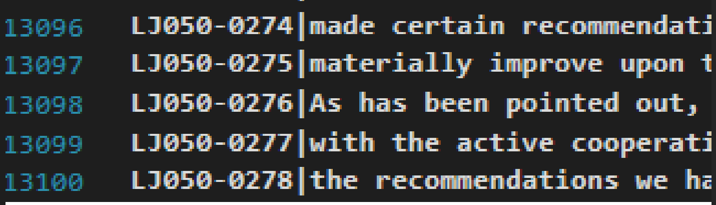
- Data/LJSpeech-1.1/metadata.csv



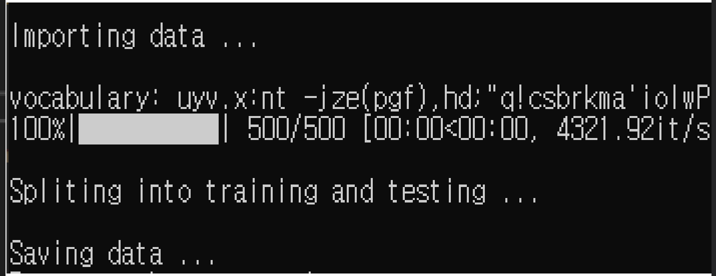
```python
print('\nImporting data ...')
metadata = pd.read_csv('data/LJSpeech-1.1/metadata.csv',
                       dtype='object', quoting=3, sep='|',
                       header=None)
metadata = metadata.iloc[:500]
metadata['norm_lower'] = metadata[2].apply(lambda x: x.lower())
texts = metadata['norm_lower']

# Infer the vocabulary
list_of_existing_chars = list(set(texts.str.cat(sep=' ')))
vocabulary = ''.join(list_of_existing_chars)
vocabulary += 'P'  # add padding character

print('\nvocabulary:',vocabulary)
# Create association between vocabulary and id
vocabulary_id = {}
i = 0
for i,char in enumerate(list(vocabulary)):
    vocabulary_id[char] = i

text_input_ml = transform_text_for_ml(texts.values,
                                      vocabulary_id,
                                      NB_CHARS_MAX)

print('\nSpliting into training and testing ...')
len_train = int(TRAIN_SET_RATIO * len(metadata))
text_input_ml_training = text_input_ml[:len_train]
text_input_ml_testing = text_input_ml[len_train:]

print('\nSaving data ...')
joblib.dump(text_input_ml_training, 'data/text_input_ml_training.pkl')
joblib.dump(text_input_ml_testing, 'data/text_input_ml_testing.pkl')

joblib.dump(vocabulary_id, 'data/vocabulary.pkl')
```



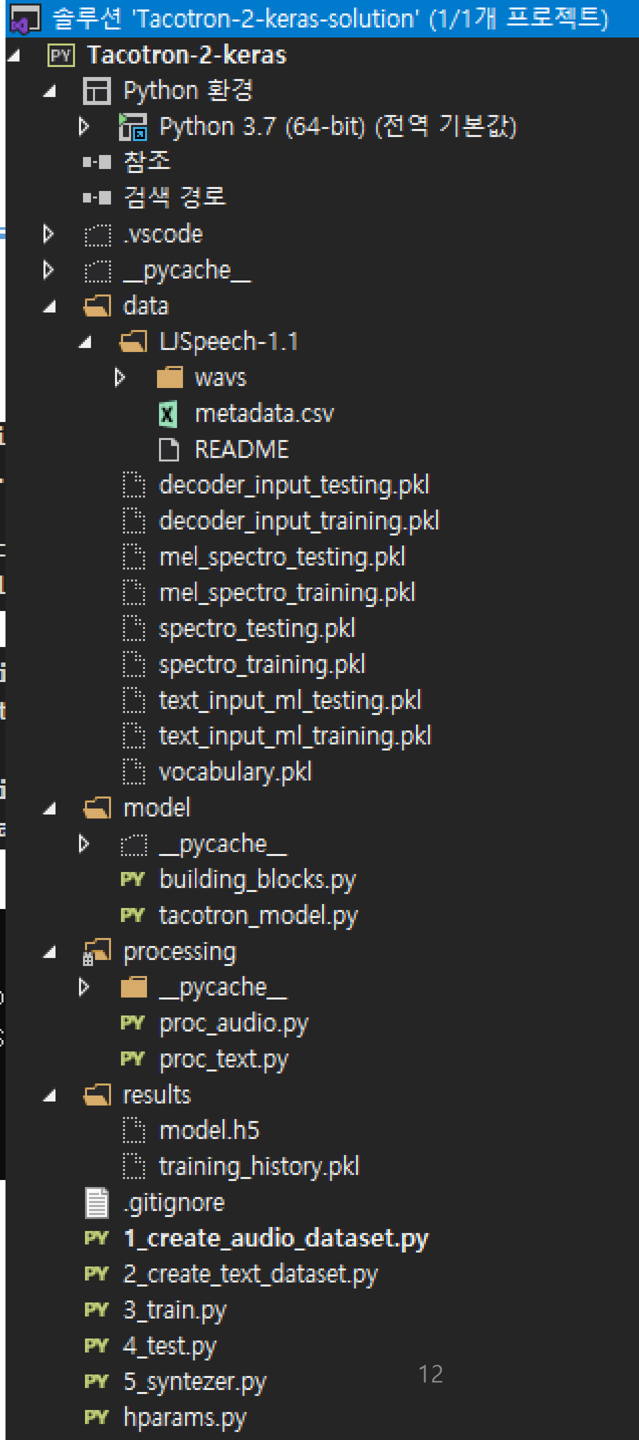
```
LJ001-0001|Printing, in the only sense wi
LJ001-0002|in being comparatively modern.
LJ001-0003|For although the Chinese took
LJ001-0004|produced the block books, whic
LJ001-0005|the invention of movable metal
```



```
LJ050-0274|made certain recommendati
LJ050-0275|materially improve upon t
LJ050-0276|As has been pointed out,
LJ050-0277|with the active cooperati
LJ050-0278|the recommendations we ha
```

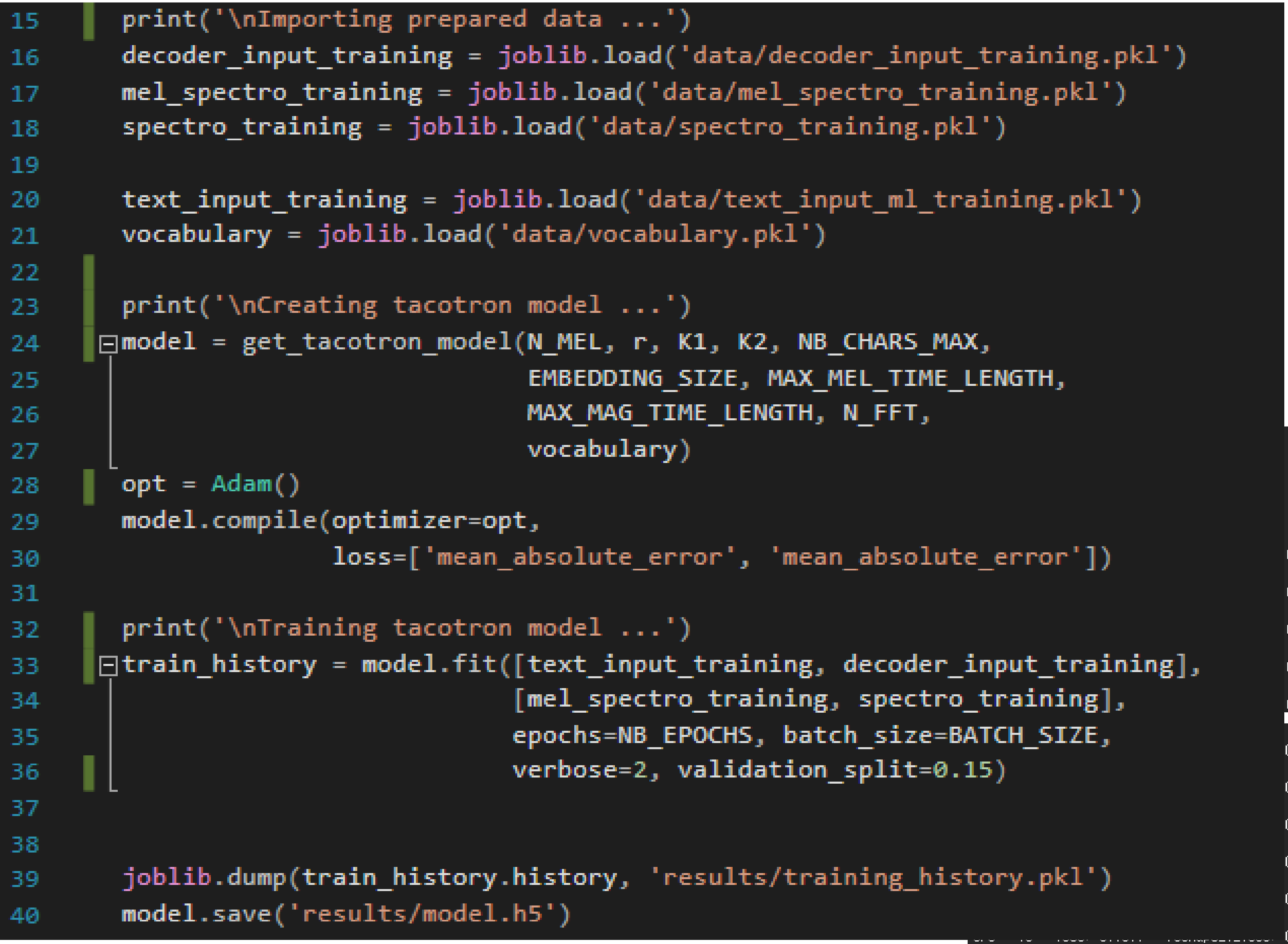


```
Importing data ...

vocabulary: uyv.x:nt -jze(pgf),hd;"q!csbrkma'iolwP
100%|██████████| 500/500 [00:00<00:00, 4321.92it/s

Spliting into training and testing ...

Saving data ...
```

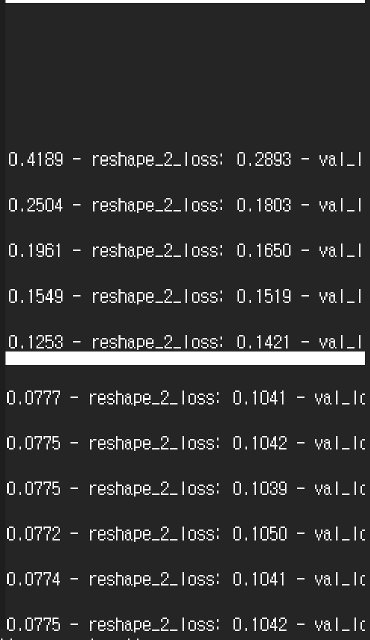


```
솔루션 'Tacotron-2-keras-solution' (1/1개 프로젝트)
◢ Tacotron-2-keras
  ◢ Python 환경
    ▷ Python 3.7 (64-bit) (전역 기본값)
    참조
    검색 경로
  ▷ .vscode
  ▷ __pycache__
  ◢ data
    ◢ LJSpeech-1.1
      ▷ wavs
        metadata.csv
        README
      decoder_input_testing.pkl
      decoder_input_training.pkl
      mel_spectro_testing.pkl
      mel_spectro_training.pkl
      spectro_testing.pkl
      spectro_training.pkl
      text_input_ml_testing.pkl
      text_input_ml_training.pkl
      vocabulary.pkl
  ◢ model
    ▷ __pycache__
      building_blocks.py
      tacotron_model.py
  ◢ processing
    ▷ __pycache__
      proc_audio.py
      proc_text.py
  ◢ results
      model.h5
      training_history.pkl
    .gitignore
    1_create_audio_dataset.py
    2_create_text_dataset.py
    3_train.py
    4_test.py
    5_syntezer.py
    hparams.py
```

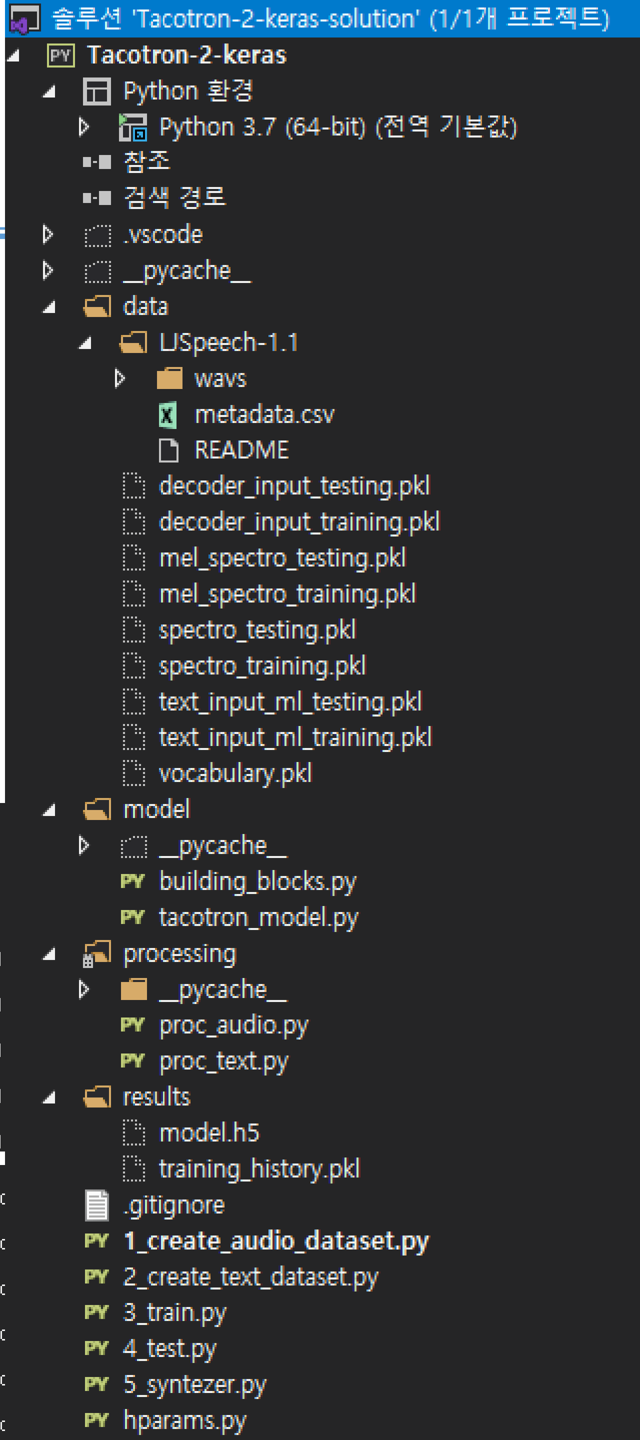
4.3 train.py



```python
print('\nImporting prepared data ...')
decoder_input_training = joblib.load('data/decoder_input_training.pkl')
mel_spectro_training = joblib.load('data/mel_spectro_training.pkl')
spectro_training = joblib.load('data/spectro_training.pkl')

text_input_training = joblib.load('data/text_input_ml_training.pkl')
vocabulary = joblib.load('data/vocabulary.pkl')

print('\nCreating tacotron model ...')
model = get_tacotron_model(N_MEL, r, K1, K2, NB_CHARS_MAX,
                           EMBEDDING_SIZE, MAX_MEL_TIME_LENGTH,
                           MAX_MAG_TIME_LENGTH, N_FFT,
                           vocabulary)
opt = Adam()
model.compile(optimizer=opt,
              loss=['mean_absolute_error', 'mean_absolute_error'])

print('\nTraining tacotron model ...')
train_history = model.fit([text_input_training, decoder_input_training],
                          [mel_spectro_training, spectro_training],
                          epochs=NB_EPOCHS, batch_size=BATCH_SIZE,
                          verbose=2, validation_split=0.15)


joblib.dump(train_history.history, 'results/training_history.pkl')
model.save('results/model.h5')
```

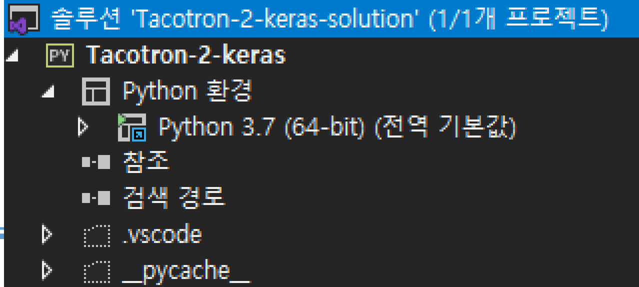


```
0.4189 - reshape_2_loss: 0.2893 - val_l

0.2504 - reshape_2_loss: 0.1803 - val_l

0.1961 - reshape_2_loss: 0.1650 - val_l

0.1549 - reshape_2_loss: 0.1519 - val_l

0.1253 - reshape_2_loss: 0.1421 - val_l

0.0777 - reshape_2_loss: 0.1041 - val_lc

0.0775 - reshape_2_loss: 0.1042 - val_lc

0.0775 - reshape_2_loss: 0.1039 - val_lc

0.0772 - reshape_2_loss: 0.1050 - val_lc

0.0774 - reshape_2_loss: 0.1041 - val_lc

0.0775 - reshape_2_loss: 0.1042 - val_lc
```

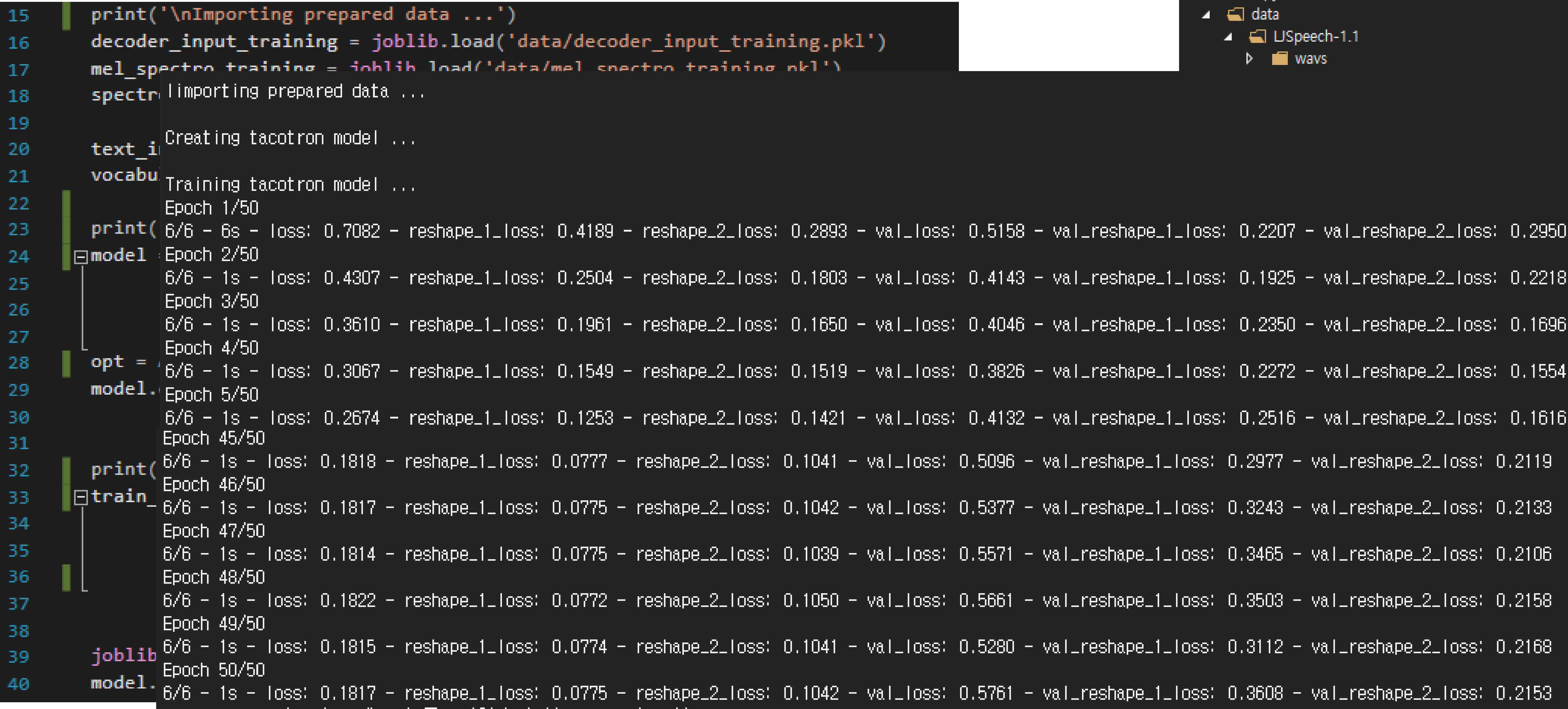


```
솔루션 'Tacotron-2-keras-solution' (1/1개 프로젝트)
  Tacotron-2-keras
    Python 환경
      Python 3.7 (64-bit) (전역 기본값)
    참조
    검색 경로
    .vscode
    __pycache__
    data
      LJSpeech-1.1
        wavs
        metadata.csv
        README
      decoder_input_testing.pkl
      decoder_input_training.pkl
      mel_spectro_testing.pkl
      mel_spectro_training.pkl
      spectro_testing.pkl
      spectro_training.pkl
      text_input_ml_testing.pkl
      text_input_ml_training.pkl
      vocabulary.pkl
    model
      __pycache__
      building_blocks.py
      tacotron_model.py
    processing
      __pycache__
      proc_audio.py
      proc_text.py
    results
      model.h5
      training_history.pkl
    .gitignore
    1_create_audio_dataset.py
    2_create_text_dataset.py
    3_train.py
    4_test.py
    5_syntezer.py
    hparams.py
```

4.3 train.py



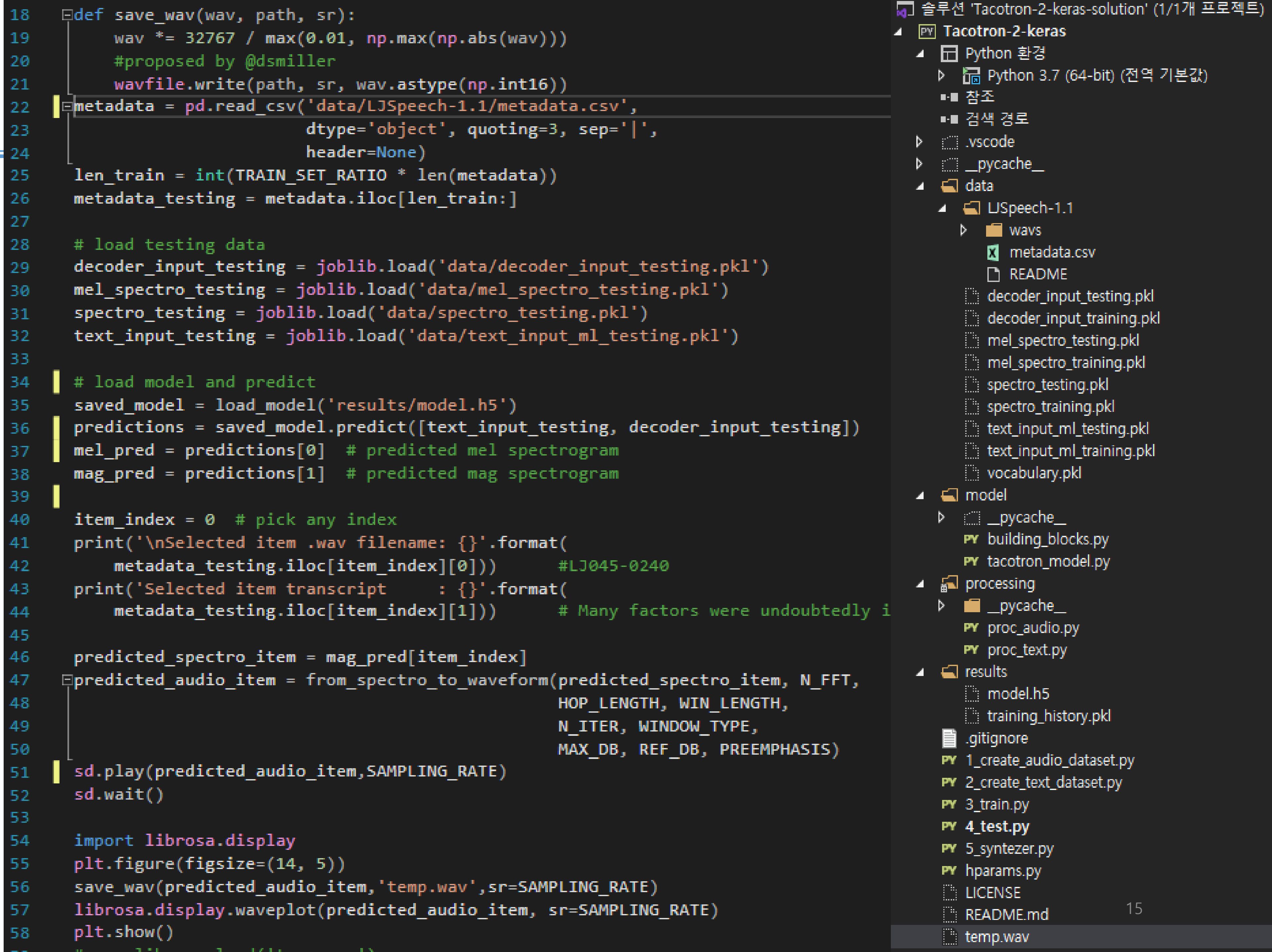
솔루션 'Tacotron-2-keras-solution' (1/1개 프로젝트)
Tacotron-2-keras
Python 환경
Python 3.7 (64-bit) (전역 기본값)
참조
검색 경로
.vscode
__pycache__
data
LJSpeech-1.1
wavs



```
print('\nImporting prepared data ...')
decoder_input_training = joblib.load('data/decoder_input_training.pkl')
mel_spectro_training = joblib.load('data/mel_spectro_training.pkl')
spectr

text_i
vocabu

print(
model

opt = 
model.


print(
train_



joblib
model.
```

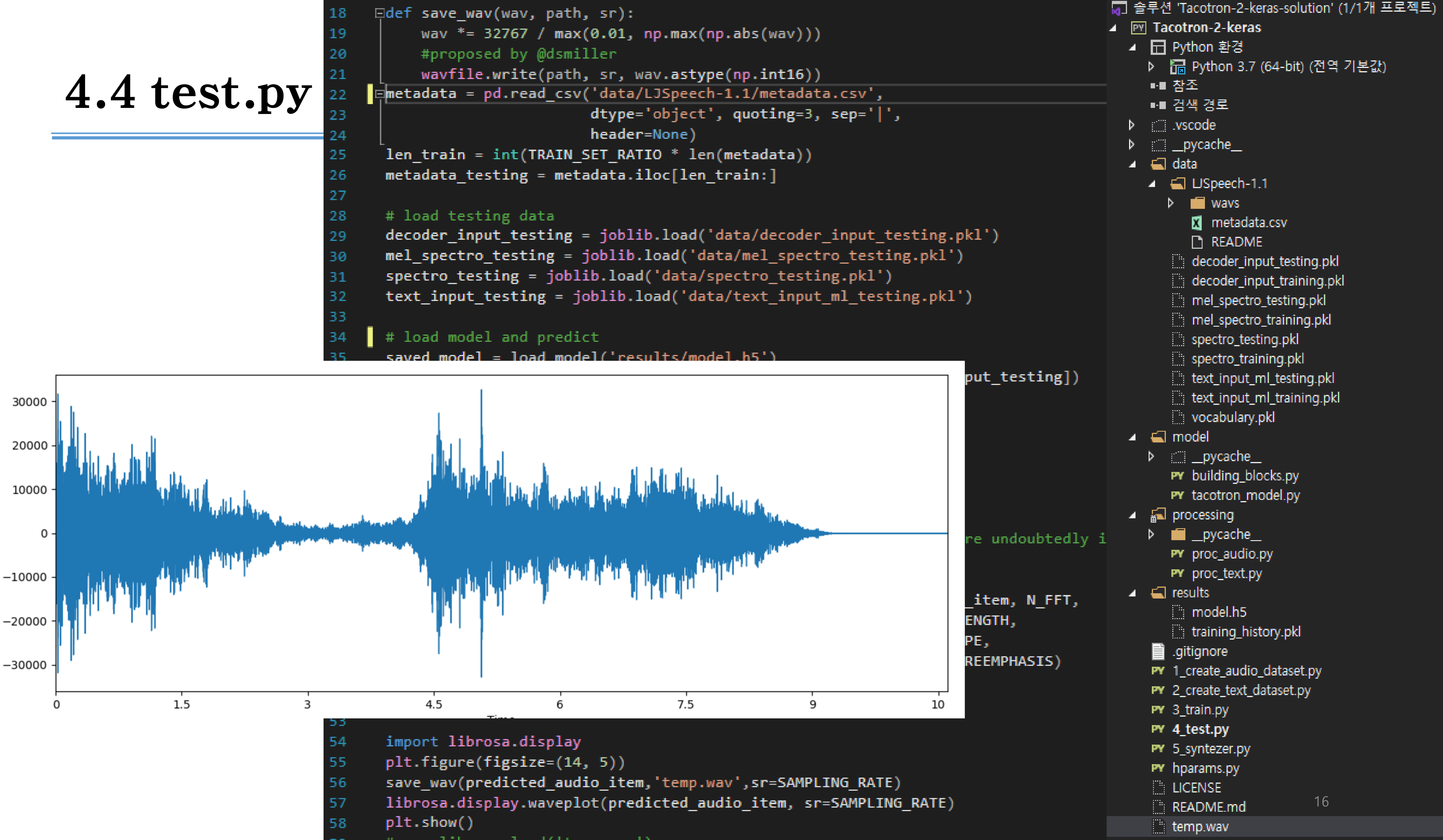
```
Importing prepared data ...

Creating tacotron model ...

Training tacotron model ...
Epoch 1/50
6/6 - 6s - loss: 0.7082 - reshape_1_loss: 0.4189 - reshape_2_loss: 0.2893 - val_loss: 0.5158 - val_reshape_1_loss: 0.2207 - val_reshape_2_loss: 0.2950
Epoch 2/50
6/6 - 1s - loss: 0.4307 - reshape_1_loss: 0.2504 - reshape_2_loss: 0.1803 - val_loss: 0.4143 - val_reshape_1_loss: 0.1925 - val_reshape_2_loss: 0.2218
Epoch 3/50
6/6 - 1s - loss: 0.3610 - reshape_1_loss: 0.1961 - reshape_2_loss: 0.1650 - val_loss: 0.4046 - val_reshape_1_loss: 0.2350 - val_reshape_2_loss: 0.1696
Epoch 4/50
6/6 - 1s - loss: 0.3067 - reshape_1_loss: 0.1549 - reshape_2_loss: 0.1519 - val_loss: 0.3826 - val_reshape_1_loss: 0.2272 - val_reshape_2_loss: 0.1554
Epoch 5/50
6/6 - 1s - loss: 0.2674 - reshape_1_loss: 0.1253 - reshape_2_loss: 0.1421 - val_loss: 0.4132 - val_reshape_1_loss: 0.2516 - val_reshape_2_loss: 0.1616
Epoch 45/50
6/6 - 1s - loss: 0.1818 - reshape_1_loss: 0.0777 - reshape_2_loss: 0.1041 - val_loss: 0.5096 - val_reshape_1_loss: 0.2977 - val_reshape_2_loss: 0.2119
Epoch 46/50
6/6 - 1s - loss: 0.1817 - reshape_1_loss: 0.0775 - reshape_2_loss: 0.1042 - val_loss: 0.5377 - val_reshape_1_loss: 0.3243 - val_reshape_2_loss: 0.2133
Epoch 47/50
6/6 - 1s - loss: 0.1814 - reshape_1_loss: 0.0775 - reshape_2_loss: 0.1039 - val_loss: 0.5571 - val_reshape_1_loss: 0.3465 - val_reshape_2_loss: 0.2106
Epoch 48/50
6/6 - 1s - loss: 0.1822 - reshape_1_loss: 0.0772 - reshape_2_loss: 0.1050 - val_loss: 0.5661 - val_reshape_1_loss: 0.3503 - val_reshape_2_loss: 0.2158
Epoch 49/50
6/6 - 1s - loss: 0.1815 - reshape_1_loss: 0.0774 - reshape_2_loss: 0.1041 - val_loss: 0.5280 - val_reshape_1_loss: 0.3112 - val_reshape_2_loss: 0.2168
Epoch 50/50
6/6 - 1s - loss: 0.1817 - reshape_1_loss: 0.0775 - reshape_2_loss: 0.1042 - val_loss: 0.5761 - val_reshape_1_loss: 0.3608 - val_reshape_2_loss: 0.2153
```

4.4 test.py



```python
def save_wav(wav, path, sr):
    wav *= 32767 / max(0.01, np.max(np.abs(wav)))
    #proposed by @dsmiller
    wavfile.write(path, sr, wav.astype(np.int16))
metadata = pd.read_csv('data/LJSpeech-1.1/metadata.csv',
                       dtype='object', quoting=3, sep='|',
                       header=None)
len_train = int(TRAIN_SET_RATIO * len(metadata))
metadata_testing = metadata.iloc[len_train:]

# load testing data
decoder_input_testing = joblib.load('data/decoder_input_testing.pkl')
mel_spectro_testing = joblib.load('data/mel_spectro_testing.pkl')
spectro_testing = joblib.load('data/spectro_testing.pkl')
text_input_testing = joblib.load('data/text_input_ml_testing.pkl')

# load model and predict
saved_model = load_model('results/model.h5')
predictions = saved_model.predict([text_input_testing, decoder_input_testing])
mel_pred = predictions[0]  # predicted mel spectrogram
mag_pred = predictions[1]  # predicted mag spectrogram

item_index = 0  # pick any index
print('\nSelected item .wav filename: {}'.format(
    metadata_testing.iloc[item_index][0]))      #LJ045-0240
print('Selected item transcript     : {}'.format(
    metadata_testing.iloc[item_index][1]))      # Many factors were undoubtedly i

predicted_spectro_item = mag_pred[item_index]
predicted_audio_item = from_spectro_to_waveform(predicted_spectro_item, N_FFT,
                                                HOP_LENGTH, WIN_LENGTH,
                                                N_ITER, WINDOW_TYPE,
                                                MAX_DB, REF_DB, PREEMPHASIS)
sd.play(predicted_audio_item,SAMPLING_RATE)
sd.wait()

import librosa.display
plt.figure(figsize=(14, 5))
save_wav(predicted_audio_item,'temp.wav',sr=SAMPLING_RATE)
librosa.display.waveplot(predicted_audio_item, sr=SAMPLING_RATE)
plt.show()
```

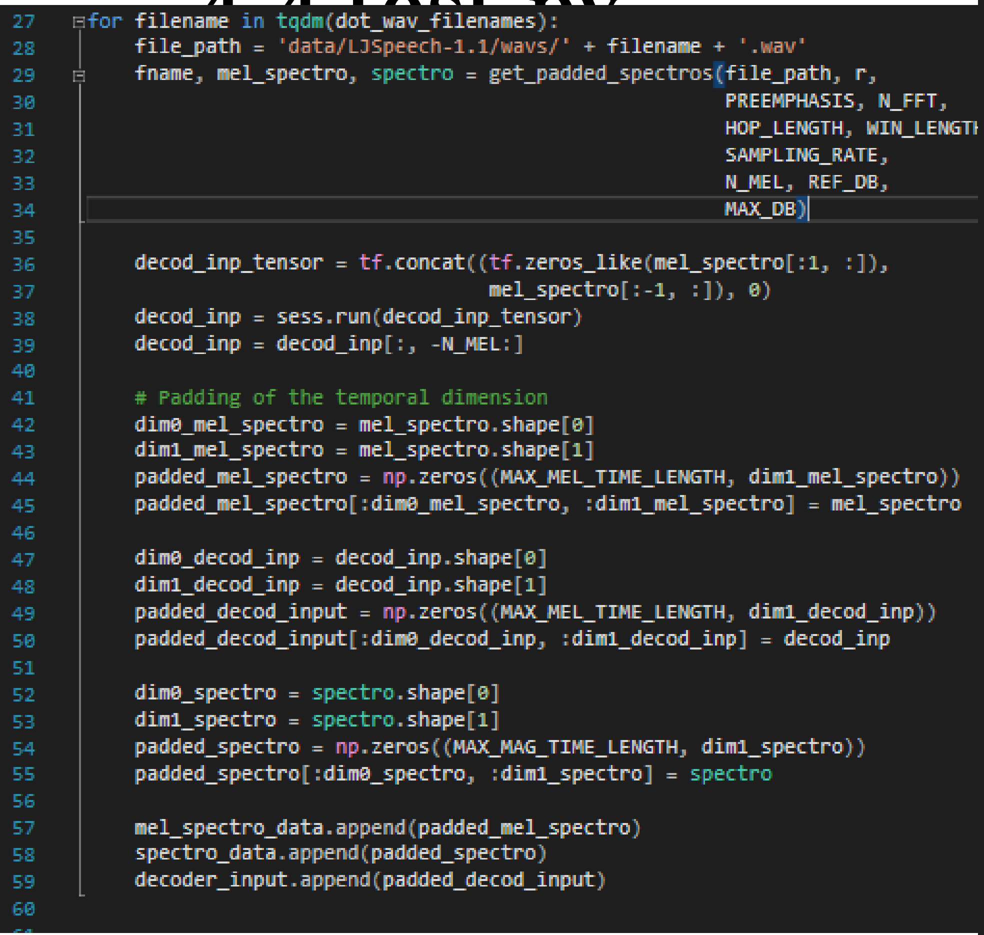
4.4 test.py



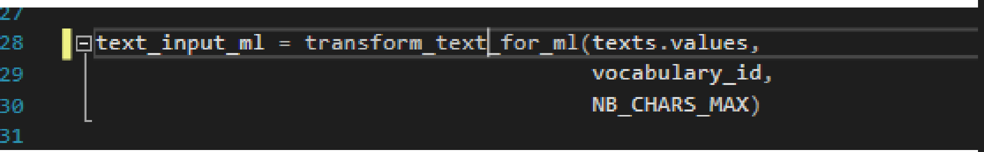
```
def save_wav(wav, path, sr):
    wav *= 32767 / max(0.01, np.max(np.abs(wav)))
    #proposed by @dsmiller
    wavfile.write(path, sr, wav.astype(np.int16))
metadata = pd.read_csv('data/LJSpeech-1.1/metadata.csv',
                       dtype='object', quoting=3, sep='|',
                       header=None)
len_train = int(TRAIN_SET_RATIO * len(metadata))
metadata_testing = metadata.iloc[len_train:]

# load testing data
decoder_input_testing = joblib.load('data/decoder_input_testing.pkl')
mel_spectro_testing = joblib.load('data/mel_spectro_testing.pkl')
spectro_testing = joblib.load('data/spectro_testing.pkl')
text_input_testing = joblib.load('data/text_input_ml_testing.pkl')

# load model and predict
saved_model = load_model('results/model.h5')
                                                            put_testing])

                                                            re undoubtedly i

                                                            _item, N_FFT,
                                                            ENGTH,
                                                            PE,
                                                            REEMPHASIS)

import librosa.display
plt.figure(figsize=(14, 5))
save_wav(predicted_audio_item,'temp.wav',sr=SAMPLING_RATE)
librosa.display.waveplot(predicted_audio_item, sr=SAMPLING_RATE)
plt.show()
```

솔루션 'Tacotron-2-keras-solution' (1/1개 프로젝트)
- Tacotron-2-keras
  - Python 환경
    - Python 3.7 (64-bit) (전역 기본값)
  - 참조
  - 검색 경로
  - .vscode
  - __pycache__
  - data
    - LJSpeech-1.1
      - wavs
      - metadata.csv
      - README
    - decoder_input_testing.pkl
    - decoder_input_training.pkl
    - mel_spectro_testing.pkl
    - mel_spectro_training.pkl
    - spectro_testing.pkl
    - spectro_training.pkl
    - text_input_ml_testing.pkl
    - text_input_ml_training.pkl
    - vocabulary.pkl
  - model
    - __pycache__
    - building_blocks.py
    - tacotron_model.py
  - processing
    - __pycache__
    - proc_audio.py
    - proc_text.py
  - results
    - model.h5
    - training_history.pkl
  - .gitignore
  - 1_create_audio_dataset.py
  - 2_create_text_dataset.py
  - 3_train.py
  - **4_test.py**
  - 5_syntezer.py
  - hparams.py
  - LICENSE
  - README.md
  - temp.wav

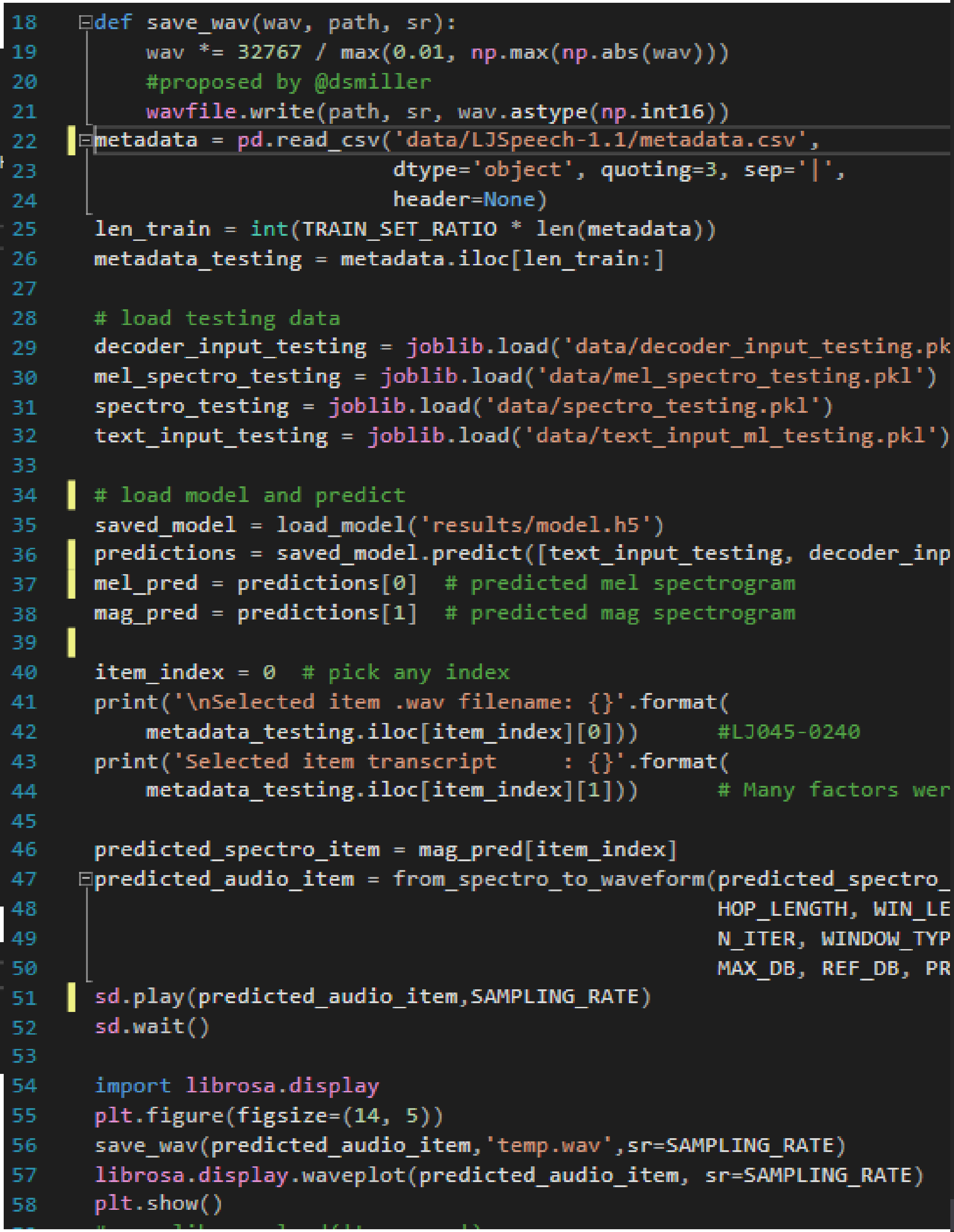
4-4 test.py



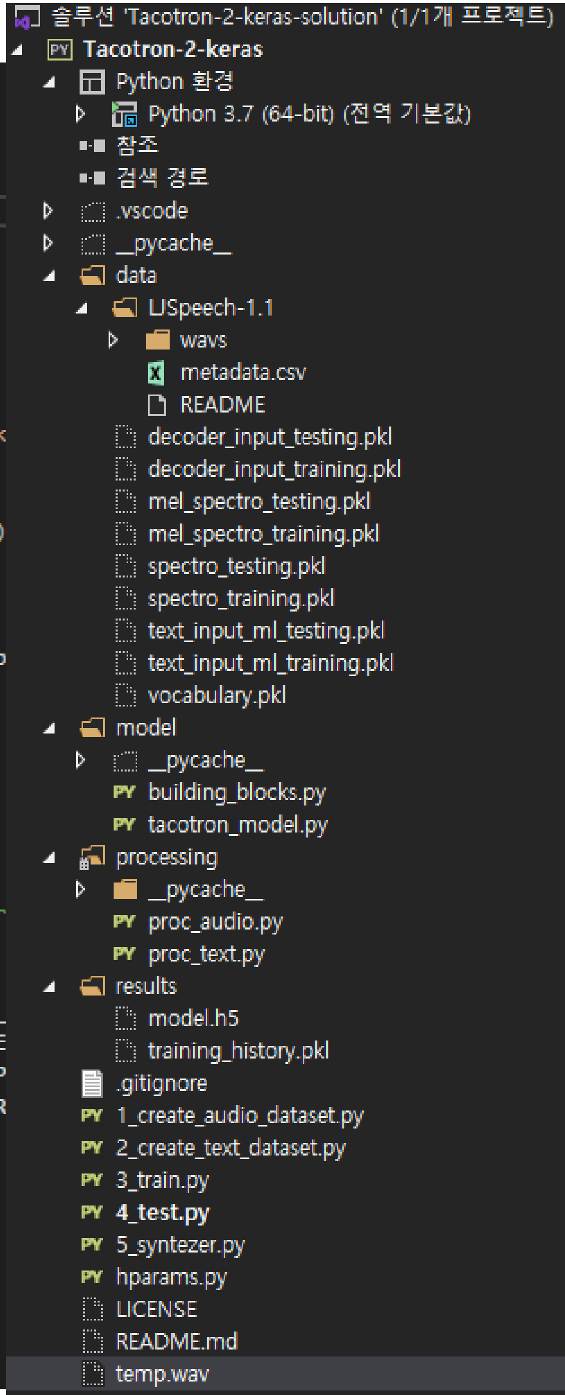
```python
for filename in tqdm(dot_wav_filenames):
    file_path = 'data/LJSpeech-1.1/wavs/' + filename + '.wav'
    fname, mel_spectro, spectro = get_padded_spectros(file_path, r,
                                                      PREEMPHASIS, N_FFT,
                                                      HOP_LENGTH, WIN_LENGTH
                                                      SAMPLING_RATE,
                                                      N_MEL, REF_DB,
                                                      MAX_DB)

    decod_inp_tensor = tf.concat((tf.zeros_like(mel_spectro[:1, :]),
                                  mel_spectro[:-1, :]), 0)
    decod_inp = sess.run(decod_inp_tensor)
    decod_inp = decod_inp[:, -N_MEL:]

    # Padding of the temporal dimension
    dim0_mel_spectro = mel_spectro.shape[0]
    dim1_mel_spectro = mel_spectro.shape[1]
    padded_mel_spectro = np.zeros((MAX_MEL_TIME_LENGTH, dim1_mel_spectro))
    padded_mel_spectro[:dim0_mel_spectro, :dim1_mel_spectro] = mel_spectro

    dim0_decod_inp = decod_inp.shape[0]
    dim1_decod_inp = decod_inp.shape[1]
    padded_decod_input = np.zeros((MAX_MEL_TIME_LENGTH, dim1_decod_inp))
    padded_decod_input[:dim0_decod_inp, :dim1_decod_inp] = decod_inp

    dim0_spectro = spectro.shape[0]
    dim1_spectro = spectro.shape[1]
    padded_spectro = np.zeros((MAX_MAG_TIME_LENGTH, dim1_spectro))
    padded_spectro[:dim0_spectro, :dim1_spectro] = spectro

    mel_spectro_data.append(padded_mel_spectro)
    spectro_data.append(padded_spectro)
    decoder_input.append(padded_decod_input)
```



```python
text_input_ml = transform_text_for_ml(texts.values,
                                      vocabulary_id,
                                      NB_CHARS_MAX)
```

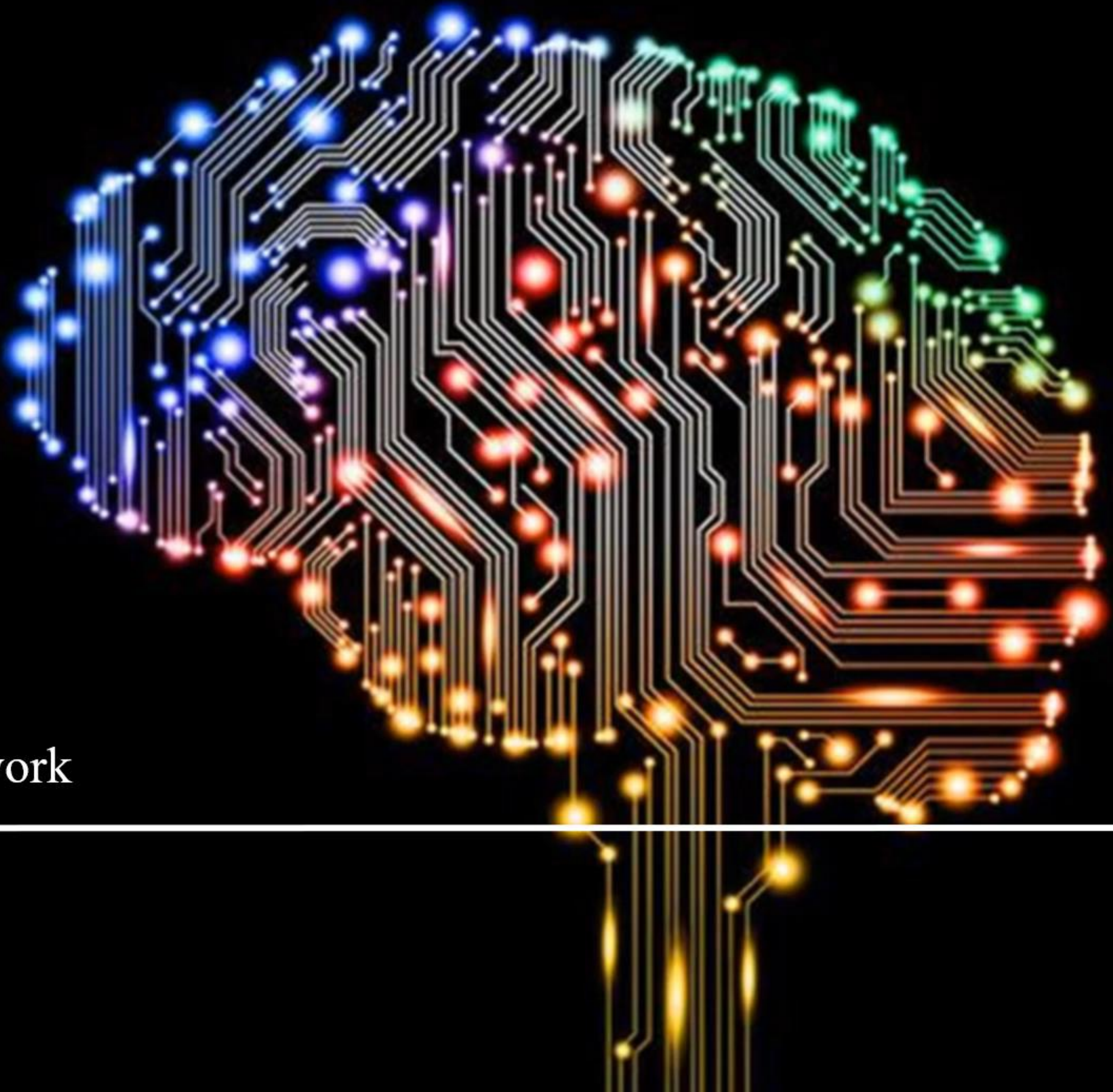


```python
def save_wav(wav, path, sr):
    wav *= 32767 / max(0.01, np.max(np.abs(wav)))
    #proposed by @dsmiller
    wavfile.write(path, sr, wav.astype(np.int16))
metadata = pd.read_csv('data/LJSpeech-1.1/metadata.csv',
                       dtype='object', quoting=3, sep='|',
                       header=None)
len_train = int(TRAIN_SET_RATIO * len(metadata))
metadata_testing = metadata.iloc[len_train:]

# load testing data
decoder_input_testing = joblib.load('data/decoder_input_testing.pk
mel_spectro_testing = joblib.load('data/mel_spectro_testing.pkl')
spectro_testing = joblib.load('data/spectro_testing.pkl')
text_input_testing = joblib.load('data/text_input_ml_testing.pkl')

# load model and predict
saved_model = load_model('results/model.h5')
predictions = saved_model.predict([text_input_testing, decoder_inp
mel_pred = predictions[0]  # predicted mel spectrogram
mag_pred = predictions[1]  # predicted mag spectrogram

item_index = 0  # pick any index
print('\nSelected item .wav filename: {}'.format(
    metadata_testing.iloc[item_index][0]))      #LJ045-0240
print('Selected item transcript     : {}'.format(
    metadata_testing.iloc[item_index][1]))      # Many factors wer

predicted_spectro_item = mag_pred[item_index]
predicted_audio_item = from_spectro_to_waveform(predicted_spectro_
                                                HOP_LENGTH, WIN_LE
                                                N_ITER, WINDOW_TYP
                                                MAX_DB, REF_DB, PR
sd.play(predicted_audio_item,SAMPLING_RATE)
sd.wait()

import librosa.display
plt.figure(figsize=(14, 5))
save_wav(predicted_audio_item,'temp.wav',sr=SAMPLING_RATE)
librosa.display.waveplot(predicted_audio_item, sr=SAMPLING_RATE)
plt.show()
```



4.4 test.py

- 각 Epoch별 학습 과정
  - 예측 음성
    - 001.wav
    - 102.wav
    - 207.wav
    - 400.wav
    - 810.wav



Deep Learning
Deep Neural Network

Yoon Joong Kim,
Hanbat National University