https://github.com/lucko515/speech-recognition-neural-network/blob/master/vui\_notebook.ipynb

vui\_notebook.ipynb : Notebook completo. Autoexplicado.

README.md : Instrucciones de configuración.{Duda :Python 3.5? o Python3.6? }

requirements.txt : archivo con las librerias y versiones utilizadas.

flac\_to\_wav.sh : Shell script para convetir .flac ?? en .flac\_to\_wav

report.html : fichero htlm de presentación de los resultados.

Archivos destacados:

smaple\_models.py : Genera 7 tipos de redes neuronales. (KERAS)

Otras informaciones : (son suposiciones)

- Guarda el LibriSpeech dataset en la carpeta 'LibriSpeech'

- Mantiene la estructura de archivos original de 'LibriSpeech'

LibriSpeech /

/ dev-clean

/ test-clean

Crear los JSON files para train y validation datsets

data\_directory/group/speaker/[file\_id1.wav, file\_id2.wav, ...,speaker.trans.txt]

Where speaker.trans.txt has in each line, file\_id transcription

def main (data\_directory, output\_file)

IMPORTANTE :

A ver si entiendo la relación entre las 'transcripciones' del dataset y los 'labels' {letras A,B,C...}

de los outputs de salida de la red nueronal :

Suposición : Para poder calcular el 'error' (LOSS) debe de pasar el texto de la 'transcripción'

a 'letras sueltas' y comparandolas con las 'letras de los labels' poder determinar en cuanto se

ha equivocado, para volver a hacer el proceso.

data\_generator.py

"""

Defines a class that is used to featurize audio clips, and provide

them to the network for training or testing. El nombre de la clase es AudioGenerator

"""

class AudioGenerator():

from utils import calc\_feat\_dim, spectrogram\_from\_file, text\_to\_int\_sequence

# la función 'text\_to\_int\_sequence' tiene que estar realcionada

# esta dentro del metodo 'get\_batch'

def get\_batch(self, partition):

""" Obtain a batch of train, validation, or test data

en función de la variable partition

"""

# el metodo 'get\_batch' devuelve (inputs, outputs)

# inputs parece un diccionario con las claves:

'the\_input': valores de X

'the\_labels' : 'transcripcion del audio caracter a caracter'

# return the arrays

outputs = {'ctc': np.zeros([self.minibatch\_size])}

inputs = {'the\_input': X\_data,

'the\_labels': labels,

'input\_length': input\_length,

'label\_length': label\_length

}

return (inputs, outputs)

# dentro de este metodo (en la linea 95 ) hace lo siguente

# calculate labels & label\_length

label = np.array(text\_to\_int\_sequence(texts[cur\_index+i]))

labels[i, :len(label)] = label

label\_length[i] = len(label)

# mete en 'label' un ndarray resultante de aplicar 'text\_to\_int\_sequence' a 'texts'

texts = self.train\_texts

¿DONDE SE USA LA CLASE 'AudioGenerator'?

Se usa en el STEP3 de jupyter\_notebook denro de la funcion

def get\_predictions(index, partition, input\_to\_softmax, model\_path):

""" Print a model's decoded predictions

Params:

index (int): The example you would like to visualize

partition (str): One of 'train' or 'validation'

input\_to\_softmax (Model): The acoustic model

model\_path (str): Path to saved acoustic model's weights

"""

# load the train and test data

data\_gen = AudioGenerator(spectrogram=False)

data\_gen.load\_train\_data()

data\_gen.load\_validation\_data()

Una vez instanciado data\_gen se utiliza hasta nuev veces con varios métodos.

¿VAMOS AL ARCHIVO 'utils.py' para ver las funcion 'text\_to\_int\_sequence'

utils.py

"""

Defines various functions for processing the data.

"""

from char\_map import char\_map, index\_map

def text\_to\_int\_sequence(text):

""" Convert text to an integer sequence """

int\_sequence = []

for c in text:

if c == ' ':

ch = char\_map['<SPACE>']

else:

ch = char\_map[c]

int\_sequence.append(ch)

return int\_sequence

def int\_sequence\_to\_text(int\_sequence):

""" Convert an integer sequence to text """

text = []

for c in int\_sequence:

ch = index\_map[c]

text.append(ch)

return text

char\_map.py

"""

Defines two dictionaries for converting

between text and integer sequences.

"""

char\_map\_str = """

' 0

<SPACE> 1

a 2

b 3

c 4

d 5

e 6

f 7

g 8

h 9

i 10

j 11

k 12

l 13

m 14

n 15

o 16

p 17

q 18

r 19

s 20

t 21

u 22

v 23

w 24

x 25

y 26

z 27

"""

# the "blank" character is mapped to 28

char\_map = {}

index\_map = {}

for line in char\_map\_str.strip().split('\n'):

ch, index = line.split()

char\_map[ch] = int(index)

index\_map[int(index)+1] = ch

index\_map[2] = ' '

¿DONDE USO LA CLAVE DEL DICCIONARIO 'the\_labels'?