import tensorflow as tf

from tensorflow import keras

import numpy as np

print(tf.\_\_version\_\_)

imdb = keras.datasets.imdb

(train\_data, train\_labels), (test\_data, test\_labels) =imdb.load\_data(num\_words=10000)

print("Training entries: {}, labels: {}".format(len(train\_data), len(train\_labels)))

print(train\_data[0])

print(train\_labels[0])

print(len(train\_data[0]), len(train\_data[1]))

# A dictionary mapping words to an integer index

word\_index = imdb.get\_word\_index()

# The first indices are reserved

word\_index = {k:(v+3) for k,v in word\_index.items()}

word\_index["<PAD>"] = 0

word\_index["<START>"] = 1

word\_index["<UNK>"] = 2

word\_index["<UNUSED>"] = 3

reverse\_word\_index = dict([(value, key) for (key, value) in word\_index.items()])

def decode\_review(text):

return ' '.join([reverse\_word\_index.get(i, '?') for i in text])

train\_data = keras.preprocessing.sequence.pad\_sequences(train\_data,

value=word\_index["<PAD>"],

padding='post',

maxlen=256)

test\_data = keras.preprocessing.sequence.pad\_sequences(test\_data,

value=word\_index["<PAD>"],

padding='post',

maxlen=256)

print(train\_data[0])

# input shape is the vocabulary count used for the movie reviews (10,000 words)

vocab\_size = 10000

model = keras.Sequential()

model.add(keras.layers.Embedding(vocab\_size, 16))

model.add(keras.layers.GlobalAveragePooling1D())

model.add(keras.layers.Dense(16, activation=tf.nn.relu))

model.add(keras.layers.Dense(1, activation=tf.nn.sigmoid))

model.summary()

model.compile(optimizer=tf.train.AdamOptimizer(),

loss='binary\_crossentropy',

metrics=['accuracy'])

x\_val = train\_data[:10000]

partial\_x\_train = train\_data[10000:]

y\_val = train\_labels[:10000]

partial\_y\_train = train\_labels[10000:]

history = model.fit(partial\_x\_train,

partial\_y\_train,

epochs=40,

batch\_size=512,

validation\_data=(x\_val, y\_val),

verbose=1)

results = model.evaluate(test\_data, test\_labels)

print(results)

history\_dict = history.history

history\_dict.keys()

dict\_keys(['loss', 'val\_loss', 'val\_acc', 'acc'])

import matplotlib.pyplot as plt

acc = history.history['acc']

val\_acc = history.history['val\_acc']

loss = history.history['loss']

val\_loss = history.history['val\_loss']

epochs = range(1, len(acc) + 1)

# "bo" is for "blue dot"

plt.plot(epochs, loss, 'bo', label='Training loss')

# b is for "solid blue line"

plt.plot(epochs, val\_loss, 'b', label='Validation loss')

plt.title('Training and validation loss')

plt.xlabel('Epochs')

plt.ylabel('Loss')

plt.legend()

plt.show()

plt.clf()

acc\_values = history\_dict['acc']

val\_acc\_values = history\_dict['val\_acc']

plt.plot(epochs, acc, 'bo', label='Training acc')

plt.plot(epochs, val\_acc, 'b', label='Validation acc')

plt.title('Training and validation accuracy')

plt.xlabel('Epochs')

plt.ylabel('Accuracy')

plt.legend()

plt.show()