ASSIGNMENT 4

|  |  |
| --- | --- |
| Assignment Date | 19 September 2022 |
| Student Name | AJITH M |
| Student Roll Number | 715319106001 |
| Maximum Marks | 2 Marks |

**ARTIFICIAL INTELLIGENCE**

* HISTORICALLY, Individual parts have been a cause of disunity & resulted in countries being exploited by foreign nations. The decline of  Mughals resulted in Individual kings fighting each other which led to the British Empire. The spirit of the ‘whole’ led to Independence.
* On the LEGAL front post-Independence, the constituent Assembly put together 395 Articles &  8 Schedules. Then began the clash of the parts -Fundamental Rights  v/s Directive Principles. Finally, the Keshavanand Bharti case focused on the ‘whole’  and gave birth to the ‘Basic Structure Doctrine’ which represents the essence of the constitution in India.
* On the CULTURAL Front, India resembles a melting pot of diverse cultures. From ancient times, it has incorporated the Greek, Persian, Turkish, Mughal & European cultures. It stands for the principle of ‘UNITY IN DIVERSITY. This has been possible because Individuals identity themselves as ‘INDIAN’ first.
* On the POLITICAL front, the recent repeal of the farm laws again proves the power of the whole. If Individual farmers had tried to protest, the farm laws might never have been repealed. The ‘Sanyukt Kisan Morcha’ formed by the farmers acted as a ‘united pressure group’ & brought the change which they intended.
* On the ECONOMIC front, India ranks 6th in terms of nominal GDP. Post COVID-19, the Indian Economy had contracted by 8%. This could have been worse without the 20% growth registered by the agricultural sector during the COVID period. This is an important lesson for policymakers to focus on the development of the economy as a ‘whole’ with a balanced focus on its parts.
* Finally, in SCIENCE & TECHNOLOGY  front, the Industrial Revolution 4.0 has begun on the ‘foundation of the Internet’. The Internet is one the best example of the power of the ‘whole’. It has given hope for universal access to technology thereby enabling an egalitarian society.

|  |
| --- |
| # coding: utf-8 |
|  |  |
|  | # In[1]: |
|  |  |
|  |  |
|  | import gensim, logging |
|  |  |
|  |  |
|  | # In[2]: |
|  |  |
|  |  |
|  | logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=logging.INFO) |
|  |  |
|  |  |
|  | # In[3]: |
|  |  |
|  |  |
|  | from gensim.models.doc2vec import LabeledSentence |
|  | from gensim.models import Doc2Vec |
|  |  |
|  |  |
|  | # In[4]: |
|  |  |
|  |  |
|  | with open("sentiment labelled sentences/yelp\_labelled.txt") as f: |
|  | for item\_no, line in enumerate(f): |
|  | print(item\_no, line) |
|  |  |
|  |  |
|  | # In[19]: |
|  |  |
|  |  |
|  | sentences = [] |
|  | sentiments = [] |
|  | with open("sentiment labelled sentences/yelp\_labelled.txt") as f: |
|  | for item\_no, line in enumerate(f): |
|  | line\_split = line.strip().split('\t') |
|  | sentences.append((line\_split[0], "yelp\_%d" % item\_no)) |
|  | sentiments.append(int(line\_split[1])) |
|  |  |
|  |  |
|  | # In[21]: |
|  |  |
|  |  |
|  | len(sentences), sentences |
|  |  |
|  |  |
|  | # In[37]: |
|  |  |
|  |  |
|  | import re |
|  | sentences = [] |
|  | sentiments = [] |
|  | for fname in ["yelp", "amazon\_cells", "imdb"]: |
|  | with open("sentiment labelled sentences/%s\_labelled.txt" % fname) as f: |
|  | for item\_no, line in enumerate(f): |
|  | line\_split = line.strip().split('\t') |
|  | sent = line\_split[0].lower() |
|  | sent = re.sub(r'\'', '', sent) |
|  | sent = re.sub(r'\W', ' ', sent) |
|  | sent = re.sub(r'\s+', ' ', sent).strip() |
|  | sentences.append(LabeledSentence(sent.split(), ["%s\_%d" % (fname, item\_no)])) |
|  | sentiments.append(int(line\_split[1])) |
|  |  |
|  |  |
|  | # In[38]: |
|  |  |
|  |  |
|  | sentences |
|  |  |
|  |  |
|  | # In[43]: |
|  |  |
|  |  |
|  | import random |
|  | class PermuteSentences(object): |
|  | def \_\_iter\_\_(self): |
|  | shuffled = list(sentences) |
|  | random.shuffle(shuffled) |
|  | for sent in shuffled: |
|  | yield sent |
|  | permuter = PermuteSentences() |
|  |  |
|  | model = Doc2Vec(permuter, min\_count=1) |
|  |  |
|  |  |
|  | # In[44]: |
|  |  |
|  |  |
|  | model.most\_similar('tasty') |

**import** numpy **as** np

**from** keras.models **import** Sequential, load\_model

**from** keras.layers **import** Dropout, Flatten, Conv2D, MaxPooling2D, Dense, Activation

**from** keras.utils **import** np\_utils

**from** keras.preprocessing.image **import** ImageDataGenerator

**from** keras.callbacks **import** TensorBoard

**import** itertools

Using TensorFlow backend.

In [2]:

*# all images will be converted to this size*

ROWS **=** 256

COLS **=** 256

CHANNELS **=** 3

In [3]:

train\_image\_generator **=** ImageDataGenerator(horizontal\_flip**=True**, rescale**=**1.**/**255, rotation\_range**=**45)

test\_image\_generator **=** ImageDataGenerator(horizontal\_flip**=False**, rescale**=**1.**/**255, rotation\_range**=**0)

train\_generator **=** train\_image\_generator**.**flow\_from\_directory('train', target\_size**=**(ROWS, COLS), class\_mode**=**'categorical')

test\_generator **=** test\_image\_generator**.**flow\_from\_directory('test', target\_size**=**(ROWS, COLS), class\_mode**=**'categorical')

Found 5994 images belonging to 200 classes.

Found 5794 images belonging to 200 classes.

In [12]:

train\_generator**.**reset()

test\_generator**.**reset()

model **=** Sequential()

model**.**add(Conv2D(64, (3,3), input\_shape**=**(ROWS, COLS, CHANNELS)))

model**.**add(Activation('relu'))

model**.**add(Conv2D(64, (3,3)))

model**.**add(Activation('relu'))

model**.**add(MaxPooling2D(pool\_size**=**(4,4)))

model**.**add(Conv2D(64, (3,3)))

model**.**add(Activation('relu'))

model**.**add(Conv2D(64, (3,3)))

model**.**add(Activation('relu'))

model**.**add(MaxPooling2D(pool\_size**=**(4,4)))

model**.**add(Flatten())

model**.**add(Dropout(0.5))

model**.**add(Dense(400))

model**.**add(Activation('relu'))

model**.**add(Dropout(0.5))

model**.**add(Dense(200))

model**.**add(Activation('softmax'))

model**.**compile(loss**=**'categorical\_crossentropy', optimizer**=**'adamax', metrics**=**['accuracy'])

model**.**summary()

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv2d\_19 (Conv2D) (None, 254, 254, 64) 1792

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

activation\_28 (Activation) (None, 254, 254, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_20 (Conv2D) (None, 252, 252, 64) 36928

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

activation\_29 (Activation) (None, 252, 252, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_10 (MaxPooling (None, 63, 63, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_21 (Conv2D) (None, 61, 61, 64) 36928

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

activation\_30 (Activation) (None, 61, 61, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_22 (Conv2D) (None, 59, 59, 64) 36928

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

activation\_31 (Activation) (None, 59, 59, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_11 (MaxPooling (None, 14, 14, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten\_5 (Flatten) (None, 12544) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_8 (Dropout) (None, 12544) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_9 (Dense) (None, 400) 5018000

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

activation\_32 (Activation) (None, 400) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_9 (Dropout) (None, 400) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_10 (Dense) (None, 200) 80200

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

activation\_33 (Activation) (None, 200) 0

=================================================================

Total params: 5,210,776

Trainable params: 5,210,776

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

In [15]:

tensorboard **=** TensorBoard(log\_dir**=**'./logs/custom')

model**.**fit\_generator(train\_generator, steps\_per\_epoch**=**512, epochs**=**10, callbacks**=**[tensorboard], verbose**=**2)

Epoch 1/10

- 434s - loss: 4.4682 - acc: 0.0687

Epoch 2/10

- 440s - loss: 4.1851 - acc: 0.0919

Epoch 3/10

- 443s - loss: 3.9278 - acc: 0.1270

Epoch 4/10

- 428s - loss: 3.6948 - acc: 0.1615

Epoch 5/10

- 437s - loss: 3.4944 - acc: 0.1935

Epoch 6/10

- 439s - loss: 3.3103 - acc: 0.2196

Epoch 7/10

- 438s - loss: 3.1253 - acc: 0.2492

Epoch 8/10

- 443s - loss: 2.9927 - acc: 0.2757

Epoch 9/10

- 431s - loss: 2.8474 - acc: 0.2998

Epoch 10/10

- 430s - loss: 2.7354 - acc: 0.3271

Out[15]:

In [16]:

print(model**.**evaluate\_generator(test\_generator, steps**=**1000))

[3.3455331521880121, 0.22266875981161696]

In [ ]: