University of Missouri-Kansas City
School of Computing and Engineering

ICP : ICP-03

Course Name: Big Data Analytics and Applications
Course ID: 5542 COMP-SCI
Semester /Session : Spring 2022

Student ID :	**Student Name**
16334245	Mustavi Islam
16321217	Keenan Flynn

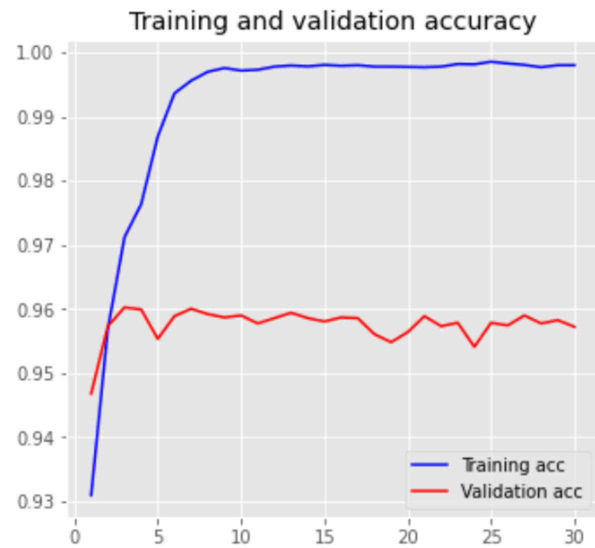
Description of the Problem :
ICP3 was a continuation of ICP2. In ICP2 we were asked to create a machine learning model to solve a sentiment analysis problem. In ICP3 we have been asked to create a deep learning model to solve that same problem. This deep learning model differs from the machine learning model in that we will only do data-preprocessing and not do feature engineering. This theoretically means less work, but it requires knowledge of neural networks and deep learning approaches

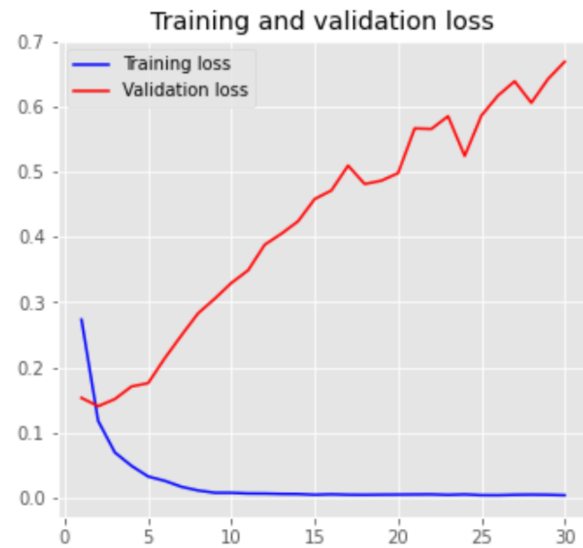
Description of Solution :
To solve this problem, a similar approach was taken as in ICP2, but a different classification model is used.

1. The data was imported into a Pandas Dataframe.
    a. Data is split into feature (tweet column) and labels (flag column)
2. String manipulation and cleaning was done to the ‘tweet’ column.
    a. ‘@user’ was removed
    b. All punctuation was removed using Regular Expressions
    c. Numbers were removed
3. NLTK library was used to do natural language processing.
    a. The sentences were tokenized using the WordTokenizer()
    b. Stop words were removed.
    c. The words were lemmatized using the WordNetLemmatizer()
    d. Part of Speech tags were concatenated to each word using nltk.pos_tag()
4. The Dataframe was prepared for deep learning.
    a. Data was split into 30% testing data and 70% training data.
    b. The sentences were tokenized into integer vectors.
        i. Each word is assigned a number based on that word's frequency.
    c. Padding was added to these vectors to get them into a uniform dimension.
5. Layers were added to a Keras Sequential() object.
    a. The Sequential() object uses defined layers to learn about the data.
    b. An Embedding() layer is added. This layer tells the model that it needs to learn a new embedding task through successive tasks.
    c. A GlobalMaxPool1D() layer emphasizes the important features.
    d. Dense() layers are the neural network with typical weights and biases.
    e. Dropout() layers reduce the chance of overfitting.
6. The Model was run
    a. The model is compiled using binary cross entropy as the loss parameter, accuracy as the metric, and adam as the optimizer.
    b. The model was fit using 6 epochs and a batch size of 40.
    c. Loss was found to be 3.5% and accuracy found to be 96%..

Challenges:
This ICP presented interesting challenges. Setting up the model was not difficult, as several resources were found online. When we had initially run the model, we were getting an accuracy of ~7%. This is obviously very bad. We did research and troubleshooting to determine why we were facing this issue. Eventually we started to modify the hyperparameters. After changing the Dense layer nodes, we found that our accuracy was much better, getting close to 95%. This was very exciting to see and worth the work and research that we did.

Another challenge was deciding how many epochs to use when we fit our model. We want to minimize loss and maximize accuracy, which we visualized in the below chart.





Initially we used 30 epochs. This chart tells us that we drastically overfit the model and that the correct number of epochs to use is actually around 6.

Learning Outcomes :

1. We learned how to add layers to a Sequential() neural network.
2. We learned how to split the data into training and testing sets.
3. We learned how to transform text data into data that can be entered into a Keras neural network.
4. We learned how to modify hyperparameters to get the best performance.
5. We learned how to troubleshoot a model to get the best results whether that be high accuracy or low loss.

Resources: https://realpython.com/python-keras-text-classification/ **->** We used multiple code snippets from this site

Video Link: https://youtu.be/SAUOk8e67uw

Screenshots :

+ Code

```
[41] import numpy as np
     import pandas as pd
     import nltk
     import matplotlib.pyplot as plt
     nltk.download("popular")
```

We just loaded our dataset

```
pd.set_option('display.max_colwidth', None)
Data = pd.read_csv('https://raw.githubusercontent.com/dD2405/Twitter_Sentiment_Analysis/master/train.csv')
```

We showed our dataset where we will work with the tweets and label

```
[43] Data.head(20)
     #len(Data)
```

	id	label	tweet
0	1	0	@user when a father is dysfunctional and is so selfish he drags his kids into his dysfunction. #run
1	2	0	@user @user thanks for #lyft credit i can't use cause they don't offer wheelchair vans in pdx. #disapointed #getthanked
2	3	0	bihday your majesty
3	4	0	#model i love u take with u all the time in urð□□±!!! ð□□□ð□□□ð□□□ð□□□ð□□¦ð□□¦ð□□¦

```
Data.isnull().sum()
```

```
id       0
label    0
tweet    0
dtype: int64
```

```
[45] import re
     import string
```

We declared tweet as our features with which we will process

```
[46] features=Data.iloc[:,2].values

     features
```

```
array([' @user when a father is dysfunctional and is so selfish he drags his kids into his dysfunction.   #run',
       "@user @user thanks for #lyft credit i can't use cause they don't offer wheelchair vans in pdx.    #disapointed #getthanked",
       '  bihday your majesty', ...,
       'listening to sad songs on a monday morning otw to work is sad  ',
       '@user #sikh #temple vandalised in in #calgary, #wso condemns  act  ',
       'thank you @user for you follow  '], dtype=object)
```

Before work with data, We are deleting the unnecessary words,punctuations,white spaces and etc .Our aim is to clean the datas.

```
[47] for i in range(len(features)):
       features[i]=features[i].replace("@user","")
       features[i]=features[i].translate(str.maketrans("","",string.punctuation))
       features[i]=''.join(i for i in features[i] if not i.isdigit())
       features[i]=re.sub(r'\s+',' ',features[i],flags=re.I)
       features[i]=re.sub(r'[!@#$%^&*()_+|\}{;:/><.}]','',features[i],flags=re.I)
       features[i]=re.sub(r'\s+[a-zA-Z]\s+', ' ',features[i])
       features=Data.iloc[:,2].values
```

After cleaning our datas,We showed them

```
Data.iloc[:,2].values

array([' when father is dysfunctional and is so selfish he drags his kids into his dysfunction run',
       ' thanks for lyft credit cant use cause they dont offer wheelchair vans in pdx disapointed getthanked',
       ' bihday your majesty', ...,
       'listening to sad songs on monday morning otw to work is sad ',
       ' sikh temple vandalised in in calgary wso condemns act ',
       'thank you for you follow '], dtype=object)
```

We declared label as our desired output

+ Code + Text

```
[51] labels=Data.iloc[:,1].values

     labels

array([0, 0, 0, ..., 0, 1, 0])
```

Using NLTK, we tokenize the tweets into a list of words

```
[52] import nltk
     from nltk import word_tokenize

     Data['tweet']=Data['tweet'].apply( lambda x : word_tokenize(x) )
```

We removed the stop words and non alpha words

```
[54] from nltk.corpus import stopwords
     stopwords=stopwords.words("english")
     Data['tweet']=Data['tweet'].apply( lambda x : [word for word in x if word not in stopwords and word.isalpha() ])

     Data
```

	id	label	tweet
0	1	0	[father, dysfunctional, selfish, drags, kids, dysfunction, run]
1	2	0	[thanks, lyft, credit, cant, use, cause, dont, offer, wheelchair, vans, pdx, disapointed, getthanked]
2	3	0	[bihday, majesty]
3	4	0	[model, love, take, time]

We stemmed the words to bring them in the basic form

```
[55] from nltk import WordNetLemmatizer
     lemma = WordNetLemmatizer()
     Data['tweet']=Data['tweet'].apply(lambda x : [lemma.lemmatize(word) for word in x])
     Data
```

	id	label	tweet
0	1	0	[father, dysfunctional, selfish, drag, kid, dysfunction, run]
1	2	0	[thanks, lyft, credit, cant, use, cause, dont, offer, wheelchair, van, pdx, disapointed, getthanked]
2	3	0	[bihday, majesty]
3	4	0	[model, love, take, time]
4	5	0	[factsguide, society, motivation]
...	...	...	...
31957	31958	0	[ate, isz]

Add POS tags to each word

```
[56] Data['tweet']=Data['tweet'].apply(lambda x : [nltk.pos_tag(x)])
     Data['tweet']=Data['tweet'].apply(lambda x : [str(word_pair[0])+'_'+str(word_pair[1]) for word_pair in x[0]])
```

Modify dataframe structure to prepare data for DL

```
[57] Data['tweet']=Data['tweet'].apply(lambda x : [' '.join(x)])
     Data['tweet']=Data['tweet'].str[0]
     Data
```

	id	label	tweet
0	1	0	father_RBR dysfunctional_JJ selfish_JJ drag_NN kid_NN dysfunction_NN run_VB
1	2	0	thanks_NNS lyft_VBP credit_NN cant_NN use_NN cause_NN dont_NN offer_VBP wheelchair_NN van_NN pdx_NN disapointed_VBD getthanked_VBD
2	3	0	bihday_NN majesty_NN

We split the dataset. We use 70% of data for training and 30% for testing

```
import sklearn
from sklearn.model_selection import train_test_split

features=Data.iloc[:,2].values
X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=0.30, random_state=42)
```

```
from keras.preprocessing.text import Tokenizer
token=Tokenizer(num_words=10000)
token.fit_on_texts(X_train)    #was fit_on_sequences but was returning empty lists when calling texts_to_sequences
```

Embed the words

```
[62] X_train=token.texts_to_sequences(X_train)
     X_test=token.texts_to_sequences(X_test)
     maxlen=200
```

```
[63] X_train
```

```
  7,
  929,
  9,
  253,
  4,
  1476,
  2,
  172,
  10,
  35,
```

```python
from keras.preprocessing.sequence import pad_sequences

X_train=pad_sequences(X_train,padding='post',maxlen=maxlen)
X_test=pad_sequences(X_test,padding='post',maxlen=maxlen)
```

```python
X_test.shape
```

```
(9589, 200)
```

```python
[27] X_train.shape
```

```
(22373, 200)
```

```python
[28] features.shape
```

```
(31962,)
```

```python
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, BatchNormalization, Activation, GlobalMaxPool1D
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras.constraints import maxnorm
from keras.utils import np_utils
from keras.layers.pooling import GlobalMaxPooling1D
from keras.layers.embeddings import Embedding
```

```python
[67] model=Sequential()
```

Here We used embedding layer at first Then we used flattening t Then we used two dense layer with 12 and 7 unit respectively We used

```python
vocab_size=len(token.word_index)+1
model.add(Embedding(input_dim=vocab_size,output_dim=50,input_length=maxlen, trainable=True))
model.add(GlobalMaxPool1D())
#model.add(Flatten())


model.add(Dense(12,activation='relu'))
model.add(Dropout(0.2))

model.add(Dense(7,activation='relu'))
model.add(Dropout(0.2))

model.add(Dense(1,activation='sigmoid'))

model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])

model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, 200, 50)           1438250

 global_max_pooling1d_1 (Glo  (None, 50)               0
 balMaxPooling1D)

 dense_3 (Dense)             (None, 12)                612

 dropout_2 (Dropout)         (None, 12)                0

 dense_4 (Dense)             (None, 7)                 91

 dropout_3 (Dropout)         (None, 7)                 0

 dense_5 (Dense)             (None, 1)                 8

=================================================================
Total params: 1,438,961
Trainable params: 1,438,961
Non-trainable params: 0
_________________________________________________________________
```

We now need to fit the model

```
[69] history = model.fit(X_train,y_train,batch_size=40,epochs=6,verbose=True,validation_data=(X_test,y_test))

Epoch 1/6
560/560 [==============================] - 12s 21ms/step - loss: 0.3348 - accuracy: 0.9170 - val_loss: 0.1857 - val_accuracy: 0.9287
Epoch 2/6
560/560 [==============================] - 12s 21ms/step - loss: 0.1594 - accuracy: 0.9490 - val_loss: 0.1293 - val_accuracy: 0.9564
Epoch 3/6
560/560 [==============================] - 12s 21ms/step - loss: 0.0974 - accuracy: 0.9756 - val_loss: 0.1358 - val_accuracy: 0.9595
Epoch 4/6
560/560 [==============================] - 12s 21ms/step - loss: 0.0681 - accuracy: 0.9848 - val_loss: 0.1399 - val_accuracy: 0.9587
Epoch 5/6
560/560 [==============================] - 12s 21ms/step - loss: 0.0481 - accuracy: 0.9907 - val_loss: 0.1562 - val_accuracy: 0.9594
Epoch 6/6
560/560 [==============================] - 12s 21ms/step - loss: 0.0357 - accuracy: 0.9941 - val_loss: 0.1867 - val_accuracy: 0.9589
```

```
loss, accuracy = model.evaluate(X_train, y_train, verbose=False)
print("Training Accuracy: {:.4f}".format(accuracy))
loss, accuracy = model.evaluate(X_test, y_test, verbose=False)
print("Testing Accuracy:  {:.4f}".format(accuracy))
```

```
Training Accuracy: 0.9982
Testing Accuracy:  0.9589
```

```
plot_history(history)
```