كلية الحوسبة والمعلوماتية |
College of Computing and Informatics
الجامعة السعودية الإلكترونية
SAUDI ELECTRONIC UNIVERSITY
2011-1432

Second Semester – 2021/2022

Course Code	DS650
Course Name	Predictive Analytics
Assignment type	Critical Thinking
Module	05
Total Points	105 Points

Student ID	G200007615
Student Name	Abdulaziz Alqumayzi
CRN	21601

Solutions:

Critical Thinking Assignment 2

Develop a Program to Identify Fake News

Introduction

In this exercise, we will create a program to detect fake news in published articles, as well as present a python programming code and program results. The train dataset from Kaggle was used: https://www.kaggle.com/c/fake-news/data?select=train.csv

Python Programming Code

```
#!/usr/bin/env python
# coding: utf-8

# In[51]:


#!pip install transformers
#!pip install nltk
#!pip install sklearn
#!pip install seaborn
#nltk.download('stopwords')
#nltk.download('punkt')
#nltk.download('wordnet')
#nltk.download('omw-1.4')


# In[50]:


import seaborn as sns

#Libraries for text preprocessing
import re
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem.porter import PorterStemmer
from nltk.stem import WordNetLemmatizer
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from sklearn.model_selection import train_test_split

#Importing Tensorflow for model creation
import tensorflow as tf

#Libraries that check the accuracy of the model over test set
from sklearn.metrics import confusion_matrix, accuracy_score,
precision_score, recall_score
```

```
# In[34]:


df = pd.read_csv('train.csv')


# In[35]:


df.info()


# In[36]:


# Combining the title of the news with it's text

df["text"] = df["text"]+" "+df["title"]


# In[37]:


# Drop the unnecessary columns, title(already added to the text column) and
author

df.drop(["title", "author"], axis = 1, inplace = True)


# In[15]:


#Defining functions to clean text data

def remove_punctuation(text):  #Removing any kind of punctuation present in
the data
    return re.sub(r'[^\w\s]','', str(text))

def remove_urls(text):  #Removing urls from the data
    return re.sub(r"http\S+", " ", str(text))

def remove_stopwords(text):  #Removing stopwords(eg. this, that, am, be
etc)
    stop = stopwords.words("english")
    final_text = []
    for i in str(text).split():
        if i.strip() not in stop:
            final_text.append(i.strip())
    return " ".join(final_text)

def tokenize(text):
    tokens = re.split('\W+',text) #W+ means that either a word character
(A-Z) or a dash(-) can go there.
    return tokens

def tokenize_words(text):  #Converting all the text to lower case
    return word_tokenize(text.lower())


def stemming(text):  #Converting the words into their stem form
    porter_stemmer = PorterStemmer()
```

```
    return porter_stemmer.stem(str(text))

def lemmatization(text):  #Applying Lemaatization i.e., converting words
into their lemma
    wordnet_lemmatizer = WordNetLemmatizer()
    return wordnet_lemmatizer.lemmatize(str(text))


# In[39]:


def clean_text(text):
    text = remove_punctuation(text)
    text = remove_urls(text)
    text = remove_stopwords(text)
    text = tokenize_words(text)
    text = stemming(text)
    return lemmatization(text)

df['text'] = df['text'].apply(lambda x: clean_text(x))


# In[40]:


# Splitting the dataset into train and test set

x_train, x_test, y_train, y_test = train_test_split(df["text"],
df["label"], test_size = 0.25, random_state = 11)


# In[41]:


#Tokenize the words into vectors because we can only give numerical data as
input to the model

max_vocab = 25000
tokenizer = Tokenizer(num_words = max_vocab)
tokenizer.fit_on_texts(x_train)

x_train = tokenizer.texts_to_sequences(x_train)
x_test = tokenizer.texts_to_sequences(x_test)


# In[42]:


#Padding is applied so that we get the same length of input for each
article

x_train = pad_sequences(x_train, padding = "post", maxlen = 256)
x_test = pad_sequences(x_test, padding = "post", maxlen = 256)


# In[43]:


# Creating the RNN model

model = tf.keras.Sequential([
```

```
    tf.keras.layers.Embedding(max_vocab, 128),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64, return_sequences
= True)),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(16)),
    tf.keras.layers.Dense(64, activation = "relu"),
    tf.keras.layers.Dropout(0.4),
    tf.keras.layers.Dense(1)
])

model.summary()


# In[44]:


# Training the model

model.compile(loss = tf.keras.losses.BinaryCrossentropy(from_logits =
True),
              optimizer = tf.keras.optimizers.Adam(1e-4),
              metrics = ["accuracy"])

model.fit(x_train, y_train, epochs = 4, validation_split = 0.2, batch_size
= 32, shuffle = True)


# In[45]:


# Evaluating the test set

model.evaluate(x_test, y_test)


# In[46]:


y_pred = model.predict(x_test)

binary_prediction = []

for i in y_pred:
    if i>=0.5:
        binary_prediction.append(1)
    else:
        binary_prediction.append(0)


# In[47]:


# Checking model accuracy

print('Accuracy on testing set:', accuracy_score(binary_prediction,
y_test))
print('Precision on testing set:', precision_score(binary_prediction,
y_test))
print('Recall on testing set:', recall_score(binary_prediction, y_test))


# In[53]:
```

```
# Creating a heatmap to visualize the confusion matrix

matrix = confusion_matrix(binary_prediction, y_test, normalize='all')
plt.figure(figsize=(5, 5))
ax= plt.subplot()
sns.heatmap(matrix, annot=True, ax = ax)

# labels, title and ticks
ax.set_xlabel('Predicted Labels', size=20)
ax.set_ylabel('True Labels', size=20)
ax.set_title('Confusion Matrix', size=20)
ax.xaxis.set_ticklabels([0,1], size=15)
ax.yaxis.set_ticklabels([0,1], size=15);
```

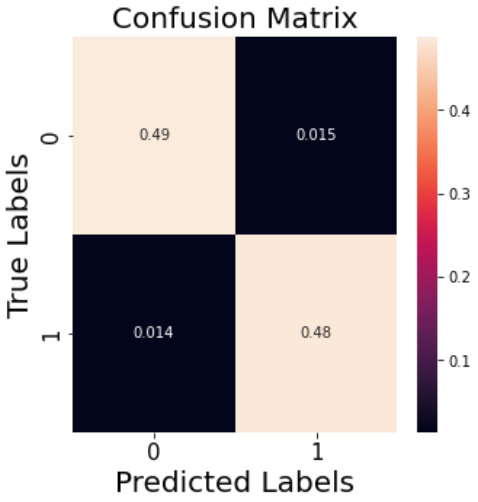
The model results:

```
Accuracy on testing set: 0.9709615384615384
Precision on testing set: 0.9694980694980695
Recall on testing set: 0.9721254355400697
```

The following figure 1 shows heatmap confusion matrix:

Figure 1

confusion matrix



References

Liu, Y. (2020). *Python machine learning by example Build Intelligent Systems using python, tensorflow 2, pytorch, and scikit-learn*. Packt.

SINGH, A. K. A. N. K. S. H. A. (2022, February 10). *Fake_news_detection*. Kaggle. Retrieved March 5, 2022, from https://www.kaggle.com/akankshasingh2001/fake-news-detection