1402-04-11

Persian Gulf University

Faculty of Intelligent Systems Engineering and Data Science

Natural Language Processing

Dr. Mohammad Bidoki

Homework #6

محمد برزگر

4010724001

آرین قره محمد زاده قشقایی

4017724001

ابتدا دیتا را از سایت Kaggle دانلود کرده و آن را در گوگل کولب آپلود میکنیم و به فرمت دیتافریم در pandas در می آوریم:

```
import pandas as pd
from google.colab import files

# Upload the CSV file
uploaded = files.upload()

# Get the first uploaded file name
filename = list(uploaded.keys())[0]

# Read the CSV file into a Pandas DataFrame
df = pd.read_csv(filename)

# Display the DataFrame
df.head()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

Saving Tweets.csv to Tweets.csv

	tweet_id	airline_sentiment	airline_sentiment_confidence	negativereason	negativereason_confidence	airline	airline_sentiment_gold	name	negativereason_gold	retw
0	570306133677760513	neutral	1.0000	NaN	NaN	Virgin America	NaN	cairdin	NaN	
1	570301130888122368	positive	0.3486	NaN	0.0000	Virgin America	NaN	jnardino	NaN	
2	570301083672813571	neutral	0.6837	NaN	NaN	Virgin	NaN	vvonnalynn	NaN	

سپس با استفاده از دستورات زیر، لیبل های neutral را حذف میکنیم. بعد از آن، لیبل های negative را به 1 و لیبل های positive را به 0 تبدیل می کنیم:

```
num_rows = df.shape[0]
num_rows
```

```
14640
```

```
# Delete any row that is Neutral
df = df[df['airline_sentiment'] != 'neutral']
num_rows = df.shape[0]
num_rows
```

```
11541
```

```
# converting negative to 1 and positive to 0
df.loc[df['airline_sentiment'] == 'negative', 'airline_sentiment'] = 1
df.loc[df['airline_sentiment'] == 'positive', 'airline_sentiment'] = 0
```

دیتافریم ما بعد از این مرحله به این شکل در می آید:

```
df
```

	tweet_id	airline_sentiment	airline_sentiment_confidence	negativereason	negativereason_confidence	airline	airline_sentiment_gold	name	negative
1	570301130888122368	0	0.3486	NaN	0.0000	Virgin America	NaN	jnardino	
3	570301031407624196	1	1.0000	Bad Flight	0.7033	Virgin America	NaN	jnardino	
4	570300817074462722	1	1.0000	Can't Tell	1.0000	Virgin America	NaN	jnardino	
5	570300767074181121	1	1.0000	Can't Tell	0.6842	Virgin America	NaN	jnardino	
6	570300616901320704	0	0.6745	NaN	0.0000	Virgin America	NaN	cjmcginnis	
...	...	...	...	...	...	...	...	...	
14633	569587705937600512	1	1.0000	Cancelled Flight	1.0000	American	NaN	RussellsWriting	

سپس با استفاده از regular expression، در متن توییت یوزر نیم ها که با @ شروع میشوند، هشتگ ها که با # شروع میشوند، لینک های url و Unicode اموجی ها را حذف می کنیم.

```
[ ] import re

    # Deleting usernames, hashtags, emojis, and URLs

    # Define a regex pattern to match words starting with '@' or '#', emojis, or URLs
    pattern = r'@\w+|#\w+|[\U0001F600-\U0001F64F\U0001F300-\U0001F5FF\U0001F680-\U0001F6FF\U0001F1E0-\U0001F1FF]+|http\S+'

    # Apply the regex pattern and replace the matched words with an empty string
    df['text'] = df['text'].str.replace(pattern, '', regex=True)
```

```
<ipython-input-6-938150e881e8>:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['text'] = df['text'].str.replace(pattern, '', regex=True)
```

سپس تمامی کاراکتر ها، بجز کاراکتر های زبان انگلیسی (a-z) را از توییت ها حذف میکنیم تا علامت های اضافی، اعداد و سایر کاراکتر ها حذف شوند.

```
[ ] # Deleting all characters except for English alphabet characters

    # Define a regex pattern to match non-alphabetic characters
    pattern = r'[^a-zA-Z\s]+'

    # Apply the regex pattern and replace non-alphabetic characters with an empty string
    df['text'] = df['text'].str.replace(pattern, '', regex=True)
```

سپس تمامی کاراکتر ها را به lowercase تبدیل میکنیم:

```
[ ]  # Making all characters lowercase

     df['text'] = df['text'].str.lower()
```

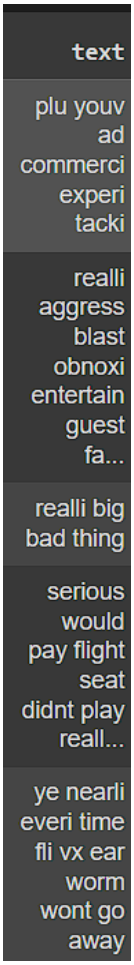
سپس stopword ها را از متن توییت ها پاک می کنیم. از کتابخانه nltk برای این کار استفاده می کنیم:

```
# Delete Stopwords

import nltk
from nltk.corpus import stopwords

nltk.download('stopwords')
stop_words = set(stopwords.words('english'))

def remove_stopwords(text):
    words = text.split()
    filtered_words = [word for word in words if word.lower() not in stop_words]
    return ' '.join(filtered_words)

df['text'] = df['text'].apply(remove_stopwords)
```

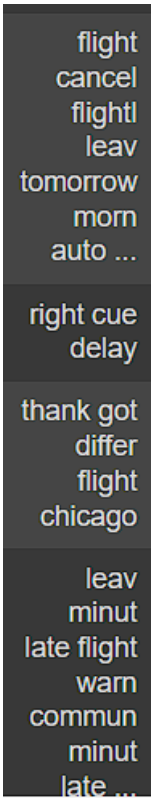
سپس با روش stemming ریشه کلمات را ریشه خود کلمه می کنیم که یکی از مراحل normalization است.

```
[ ]  from nltk.stem import PorterStemmer
     from nltk.tokenize import word_tokenize
     import nltk

     nltk.download('punkt')

     # Using stemming

     stemmer = PorterStemmer()

     def apply_stemming(text):
         words = word_tokenize(text)  # Tokenize the text into words
         stemmed_words = [stemmer.stem(word) for word in words]  # Apply stemming to each word
         return ' '.join(stemmed_words)  # Join the stemmed words back into a single string

     # Apply the apply_stemming function to the 'text' column
     df['text'] = df['text'].apply(apply_stemming)
```

متن توییت ها پس از مراحل پیش پردازش به چنین شکلی در می آیند:



text
plu youv ad commerci experi tacki
realli aggress blast obnoxi entertain guest fa...
realli big bad thing
serious would pay flight seat didnt play reall...
ye nearli everi time fli vx ear worm wont go away



flight cancel flightl leav tomorrow morn auto ...
right cue delay
thank got differ flight chicago
leav minut late flight warn commun minut late ...

سپس ستون text که شامل تمامی توییت ها هست را به یک لیست تبدیل می کنیم:

```
[ ]  # convert to list
     detail_list = df['text'].tolist()

     print(detail_list)
```

سپس با استفاده از padding می آییم تمامی ماتریس توییت ها را به یک سایز تبدیل می کنیم تا بتوانیم آن را برای ترین مدل استفاده کنیم. مقادیر 0 به سمت راست ماتریس ها اضافه شده است تا تمامی آن ها یک اندازه بشوند.

همچنین با استفاده از tokenizer زیر می آییم به هر کلمه در دیتاست یک عدد اختصاص می دهیم، که هر کلمه با عدد متناظر خود جایگزین می شود.

در خروجی قسمت زیر، هر لیست نشان دهنده متن یک توییت است و هر عدد نشان دهنده یک کلمه است:

```
[ ]  # Padding

     from keras.preprocessing.text import Tokenizer
     from tensorflow.keras.preprocessing.sequence import pad_sequences

     sentences = detail_list

     tokenizer = Tokenizer()
     tokenizer.fit_on_texts(sentences)
     sequences = tokenizer.texts_to_sequences(sentences)

     # Padding the sequences (post-padding)
     max_sequence_length = max(len(seq) for seq in sequences)
     padded_sequences = pad_sequences(sequences, maxlen=max_sequence_length, padding='post')

     print(padded_sequences)
```

```
[[ 429  374  501 ...    0    0    0]
 [  61 2235 1573 ...    0    0    0]
 [  61  331  118 ...    0    0    0]
 ...
 [   2   37  215 ...    0    0    0]
 [ 133   50   39 ...    0    0    0]
 [ 251   47    1 ...    0    0    0]]
```

سپس ستون لیبل ها را به لیست تبدیل می کنیم:

```
[ ]  labels = df['airline_sentiment'].tolist()
```

```
[ ]  df.iloc[0]['text']
```

```
'plu youv ad commerci experi tacki'
```

سپس با استفاده از تابع زیر، بزرگترین عددی که به یک کلمه اختصاص داده شده است را پیدا میکنیم که در واقع معادل است با تعداد کلمات unique در کل متون توییت ها:

```
[ ]  def find_max_number(list_of_lists):
         max_number = float('-inf')  # Initialize the maximum number with negative infinity

         # Iterate through each sublist
         for sublist in list_of_lists:
             # Iterate through each element in the sublist
             for number in sublist:
                 if number > max_number:
                     max_number = number

         return max_number


     max_number = find_max_number(padded_sequences)
     print(max_number)
```

```
7323
```

سپس مدل را به train set و test set تقسیم میکنیم که اینجا 80٪ برای ترین و 20٪ برای تست در نظر گرفتیم:

```
[ ]  from sklearn.model_selection import train_test_split

     # Splitting into train and test sets
     train_labels, test_labels, train_values, test_values = train_test_split(labels, padded_sequences, test_size=0.2, random_state=42)

     # Printing the train and test sets
     print("Train labels:", len(train_labels))
     print("Test labels:", len(test_labels))
     print("Train values:", len(train_values))
     print("Test values:", len(test_values))
```

```
Train labels: 9232
Test labels: 2309
Train values: 9232
Test values: 2309
```

سپس با استفاده از تنسورفلو و کراس، مدل درخواست شده در را میسازیم.

در اولین تلاش مدل اورفیت شد، سپس تصمیم گرفتیم dropout را 0.5 قرار بدهیم و یک پارامتر l2 regularization به لایه LSTM اضافه بکنیم، که با این روش مشکل رفع گردید و دقت خوبی را نتیجه گرفتیم.

همچنین از loss binary crossentropy و adam optimizer استفاده کردیم:

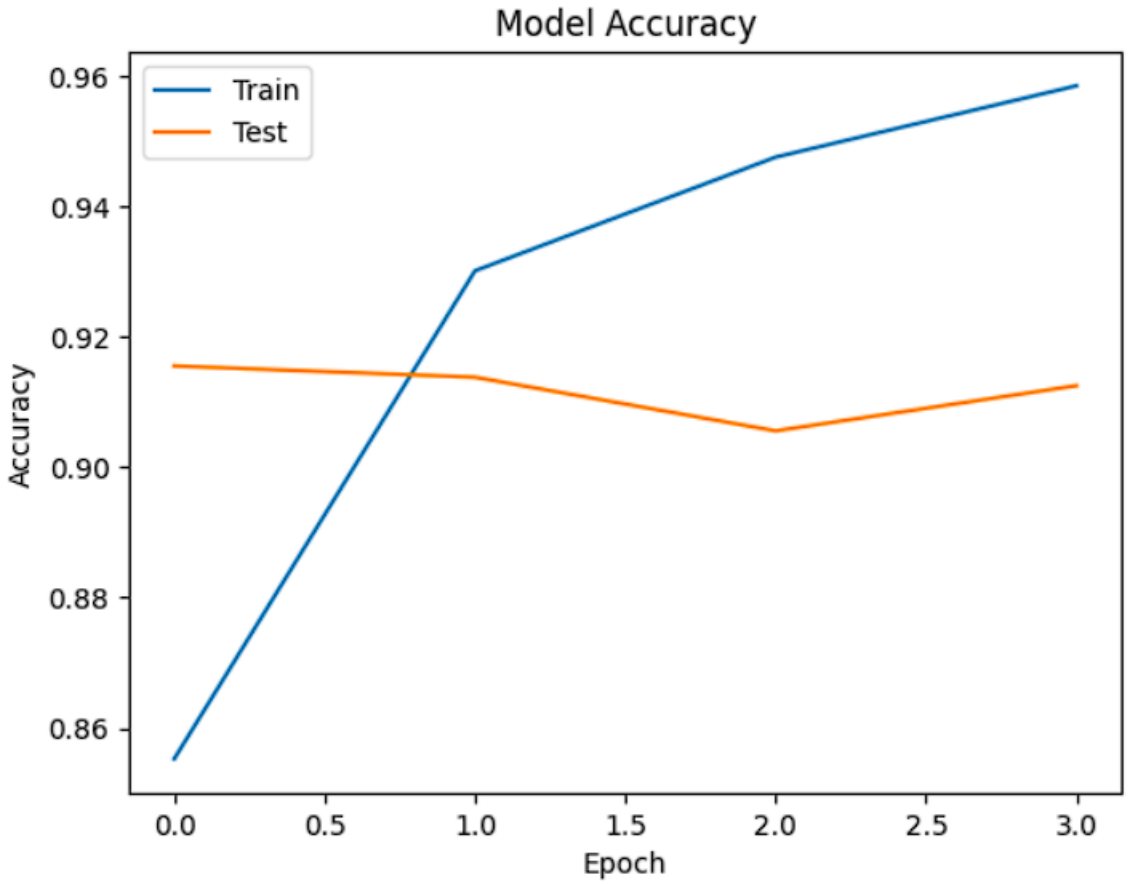
```
[ ] from tensorflow import keras
    from tensorflow.keras.models import Sequential
    from tensorflow.keras.layers import Embedding, SpatialDropout1D, LSTM, Dropout, Dense
    from keras import regularizers


    model = Sequential()
    vocab_size = max_number

    model.add(Embedding(input_dim=vocab_size, output_dim=32, input_length=max_sequence_length))

    # Second layer: SpatialDropout1D
    model.add(SpatialDropout1D(0.2))  # You can adjust the dropout rate (0.2 in this example)

    # Third layer: LSTM
    model.add(LSTM(50,kernel_regularizer=regularizers.l2(0.01)))  # Assuming you want an LSTM layer with 50 units

    # Fourth layer: Dropout
    model.add(Dropout(0.5))  # You can adjust the dropout rate (0.2 in this example)

    # Fifth layer: Dense (output layer)
    model.add(Dense(1, activation='sigmoid'))  # Assuming binary classification

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

لیستی که برای ترین و تست داشتیم را به فرمت nparray تبدیل میکنیم تا با تنسورفلو سازگارتر شود. همچنین early stopping نیز استفاده می کنیم. با استفاده از 5 ایپاک مدل ترین شده است:

```
[ ] import numpy as np
    import matplotlib.pyplot as plt
    from sklearn.metrics import precision_score, recall_score, f1_score, confusion_matrix
    from keras.callbacks import EarlyStopping

    # Define early stopping callback
    early_stopping = EarlyStopping(patience=3, monitor='val_loss')

    X_train = np.array(train_values)
    y_train = np.array(train_labels)
    X_test = np.array(test_values)
    y_test = np.array(test_labels)

    # Train the model
    history = model.fit(X_train, y_train, epochs=5, batch_size=32, validation_data=(X_test, y_test),callbacks=[early_stopping])

    # Plot accuracy
    plt.plot(history.history['accuracy'])
    plt.plot(history.history['val_accuracy'])
    plt.title('Model Accuracy')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.legend(['Train', 'Test'], loc='upper left')
    plt.show()

    # Make predictions
    y_pred_prob = model.predict(X_test)
    y_pred = (y_pred_prob > 0.5).astype(int)

    # Compute evaluation metrics
    precision = precision_score(y_test, y_pred)
    recall = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    confusion_mat = confusion_matrix(y_test, y_pred)

    # Print evaluation metrics
    print("Precision:", precision)
    print("Recall:", recall)
    print("F1-Score:", f1)
    print("Confusion Matrix:\n", confusion_mat)
```

```
Epoch 1/5
289/289 [==============================] - 24s 57ms/step - loss: 0.4696 - accuracy: 0.8553 - val_loss: 0.2405 - val_accuracy: 0.9155
Epoch 2/5
289/289 [==============================] - 4s 14ms/step - loss: 0.2014 - accuracy: 0.9301 - val_loss: 0.2558 - val_accuracy: 0.9138
Epoch 3/5
289/289 [==============================] - 4s 13ms/step - loss: 0.1557 - accuracy: 0.9476 - val_loss: 0.2468 - val_accuracy: 0.9056
Epoch 4/5
289/289 [==============================] - 2s 8ms/step - loss: 0.1251 - accuracy: 0.9585 - val_loss: 0.2723 - val_accuracy: 0.9125
```

نمودار دقت ترین و تست مدل به شکل زیر است. به دلیل اضافه کردن early stopping مدل در ایپاک 3 نتیجه را گزارش داده است.



مقادیر precision, recall, F1-Score و همچنین Confusion Matrix مدل به شکل زیر است:

```
Precision: 0.9304979253112033
Recall: 0.9634801288936627
F1-Score: 0.9467018469656993
Confusion Matrix:
 [[ 313  134]
 [  68 1794]]
```