HW2

April 6, 2023

Import necessary library

[1]:
```
import os
import numpy as np
import pandas as pd
import tensorflow as tf
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from keras.layers import Dense
from keras.models import Sequential
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
import matplotlib.pyplot as plt
from keras.layers import Lambda
```

0.0.1 Load Data

[2]:
```
data_dir = "C:\\Users\\alan\\AI Project\\dataset\\aclImdb"
```

[3]:
```
def load_data(path):
  data = []

  files = [f for f in os.listdir(path)]
  for file in files:
    with open(path+file, "r", encoding='utf8') as f:
      data.append(f.read())

  return data
```

Merge all Data into one DataFrame

[4]:
```
df_train_pos = pd.DataFrame({'review': load_data(data_dir+"\\train\\pos\\"),␣
 ↪'label': 1})
df_train_neg = pd.DataFrame({'review': load_data(data_dir+"\\train\\neg\\"),␣
 ↪'label': 0})

df_test_pos = pd.DataFrame({'review': load_data(data_dir+"\\test\\pos\\"),␣
 ↪'label': 1})
```

```
df_test_neg = pd.DataFrame({'review': load_data(data_dir+"\\test\\neg\\"),␣
 ↪'label': 0})

# Merging all df's for data cleaning and preprocessing step.
data = pd.concat([df_train_pos, df_train_neg, df_test_pos, df_test_neg],␣
 ↪ignore_index=True)
print("Total reviews in df: ", data.shape)
data.head()
```

```
Total reviews in df:  (50000, 2)
```

[4]:

	review	label
0	Bromwell High is a cartoon comedy. It ran at t…	1
1	Homelessness (or Houselessness as George Carli…	1
2	Brilliant over-acting by Lesley Ann Warren. Be…	1
3	This is easily the most underrated film inn th…	1
4	This is not the typical Mel Brooks film. It wa…	1

[5]:
```
train_df, test_df = train_test_split(data, test_size=0.2, random_state=42)
```

[6]:
```
print(train_df,test_df)
```

```
                                                  review  label
39087  Ah, here it is! A movie, which is said by peop…      0
30893  I saw this movie on PBS the first time. Then I…      1
45278  At the beginning of 'Loggerheads', we're intro…      0
16398  For the life of me, I cannot get why they woul…      0
13653  I always wrote this series off as being a comp…      0
…                                                    …    …
11284  I saw this movie at midnight on On Demand the …      1
44732  Some aspects of this production are good, such…      0
38158  I was not old enough to really appreciate the …      0
860    Nice movie with a great soundtrack which spans…      1
15795  Even though this was a made-for-TV production,…      0

[40000 rows x 2 columns]
review  label
33553  When I first saw the ad for this, I was like '…      1
9427   "A Girl's Folly" is a sort of half-comedy, hal…      1
199    I started watching the show from the first sea…      1
12447  This is a more interesting than usual porn mov…      1
39489  I suppose for 1961 this film was supposed to b…      0
…                                                    …    …
28567  River's Edge is an excellent film and it's a s…      1
25079  I kid you not. Yes, "Who's That Girl" has the …      1
18707  This is just a butchering of a wonderful story…      0
15200  Home Alone 3 is one of my least favourite movi…      0
5857   A complex story laid on the background of part…      1
```

[10000 rows x 2 columns]

[7]:
```
x_train = train_df['review'].str.lower().values
y_train = train_df['label'].values

x_test = test_df['review'].str.lower().values
y_test = test_df['label'].values
```

Bag-of-Words

[8]:
```
# The BoW and TF-IDF techniques automatically tokenize the text
# and create feature vectors based on the occurrences of words or word␣
 ↪combinations (n-grams) in the text.
bow_vectorizer = CountVectorizer(max_features=20000, ngram_range=(1, 1))
bow_vectorizer.fit(x_train)

x_train_bow = bow_vectorizer.transform(x_train)
x_test_bow = bow_vectorizer.transform(x_test)
```

TF-IDF

[9]:
```
tfidf_vectorizer = TfidfVectorizer(max_features=20000, ngram_range=(1, 1))
tfidf_vectorizer.fit(x_train)

x_train_tfidf = tfidf_vectorizer.transform(x_train)
x_test_tfidf = tfidf_vectorizer.transform(x_test)
```

0.0.2 Build Model: (BOW and TF-IDF)

[10]:
```
def build_and_train_model(x_train, y_train, x_test, y_test, input_dim):
    model = Sequential()
    model.add(Dense(128, input_dim=input_dim, activation='relu'))
    model.add(Dropout(0.2))
    model.add(Dense(64, activation='relu'))
    model.add(Dropout(0.2))
    model.add(Dense(1, activation='sigmoid'))

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

    history = model.fit(x_train, y_train,
                        batch_size=32,
                        epochs=5,
                        validation_data=(x_test, y_test))
    return model, history
```

model for Bag-of-Words

[11]:
```
# convert BoW feature sets
x_train_bow = x_train_bow.toarray()
x_test_bow = x_test_bow.toarray()
```

[12]:
```
input_dim_bow = x_train_bow.shape[1]
model_bow, history_bow = build_and_train_model(x_train_bow, y_train,␣
 ↪x_test_bow, y_test, input_dim_bow)
```

```
Epoch 1/5
1250/1250 [==============================] - 36s 28ms/step - loss: 0.3146 -
accuracy: 0.8705 - val_loss: 0.2518 - val_accuracy: 0.9011
Epoch 2/5
1250/1250 [==============================] - 38s 31ms/step - loss: 0.1675 -
accuracy: 0.9337 - val_loss: 0.2873 - val_accuracy: 0.8912
Epoch 3/5
1250/1250 [==============================] - 35s 28ms/step - loss: 0.0961 -
accuracy: 0.9651 - val_loss: 0.3435 - val_accuracy: 0.8920
Epoch 4/5
1250/1250 [==============================] - 40s 32ms/step - loss: 0.0489 -
accuracy: 0.9829 - val_loss: 0.4607 - val_accuracy: 0.8914
Epoch 5/5
1250/1250 [==============================] - 36s 28ms/step - loss: 0.0352 -
accuracy: 0.9876 - val_loss: 0.5742 - val_accuracy: 0.8830
```

model for TF-IDF

[13]:
```
# convert TF-IDF feature sets
x_train_tfidf = x_train_tfidf.toarray()
x_test_tfidf = x_test_tfidf.toarray()
```

[14]:
```
input_dim_tfidf = x_train_tfidf.shape[1]
model_tfidf, history_tfidf = build_and_train_model(x_train_tfidf, y_train,␣
 ↪x_test_tfidf, y_test, input_dim_tfidf)
```

```
Epoch 1/5
1250/1250 [==============================] - 39s 31ms/step - loss: 0.2946 -
accuracy: 0.8753 - val_loss: 0.2387 - val_accuracy: 0.9031
Epoch 2/5
1250/1250 [==============================] - 37s 30ms/step - loss: 0.1389 -
accuracy: 0.9466 - val_loss: 0.2824 - val_accuracy: 0.8947
Epoch 3/5
1250/1250 [==============================] - 35s 28ms/step - loss: 0.0643 -
accuracy: 0.9764 - val_loss: 0.3698 - val_accuracy: 0.8909
Epoch 4/5
1250/1250 [==============================] - 33s 26ms/step - loss: 0.0205 -
accuracy: 0.9926 - val_loss: 0.5382 - val_accuracy: 0.8881
Epoch 5/5
1250/1250 [==============================] - 32s 26ms/step - loss: 0.0068 -
accuracy: 0.9976 - val_loss: 0.7097 - val_accuracy: 0.8855
```

Evaluate the model: BOW

[15]:
```
score_bow, acc_bow = model_bow.evaluate(x_test_bow, y_test, batch_size=32)
print('BoW Model - Test score:', score_bow)
print('BoW Model - Test accuracy:', acc_bow)
```

```
313/313 [==============================] - 1s 4ms/step - loss: 0.5742 -
accuracy: 0.8830
BoW Model - Test score: 0.5741738080978394
BoW Model - Test accuracy: 0.8830000162124634
```

Evaluate the model: TF-IDF

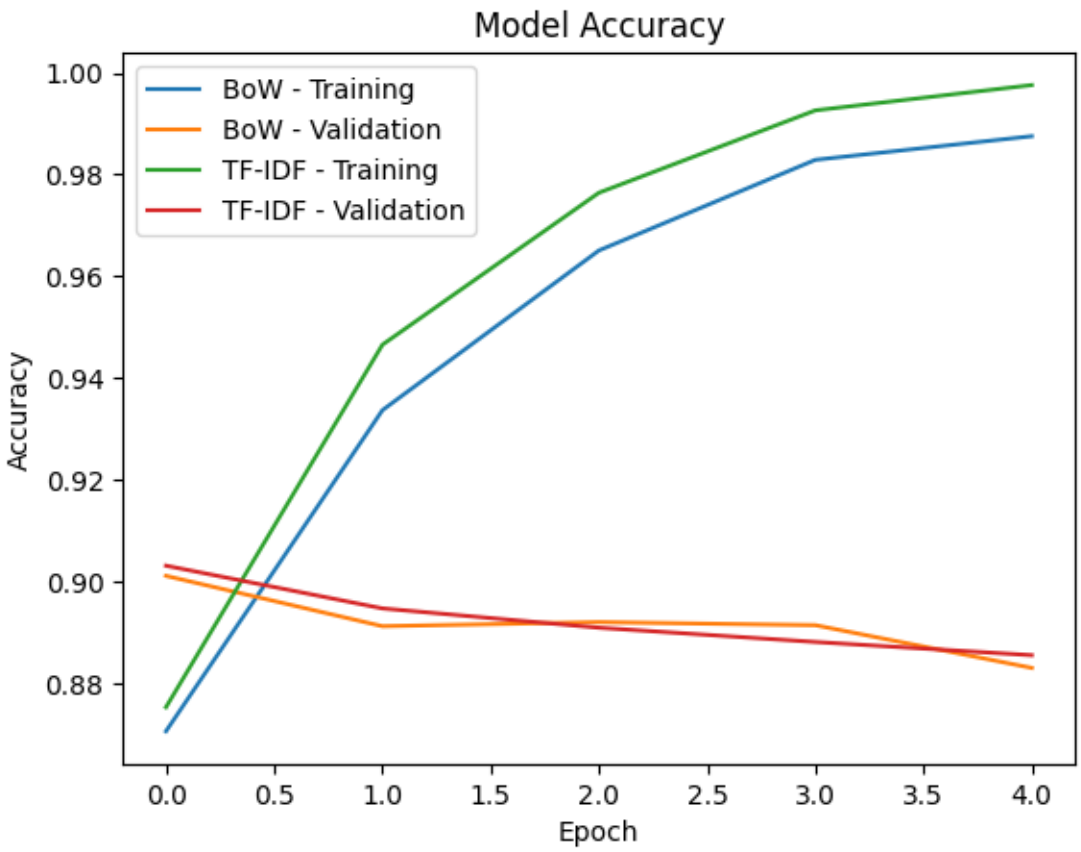
[16]:
```
score_tfidf, acc_tfidf = model_tfidf.evaluate(x_test_tfidf, y_test,␣
↪batch_size=32)
print('TF-IDF Model - Test score:', score_tfidf)
print('TF-IDF Model - Test accuracy:', acc_tfidf)
```

```
313/313 [==============================] - 1s 3ms/step - loss: 0.7097 -
accuracy: 0.8855
TF-IDF Model - Test score: 0.7097117304801941
TF-IDF Model - Test accuracy: 0.8855000138282776
```

Accuracy

[19]:
```
# Plot accuracy
plt.figure()
plt.plot(history_bow.history['accuracy'], label='BoW - Training')
plt.plot(history_bow.history['val_accuracy'], label='BoW - Validation')
plt.plot(history_tfidf.history['accuracy'], label='TF-IDF - Training')
plt.plot(history_tfidf.history['val_accuracy'], label='TF-IDF - Validation')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title('Model Accuracy')
plt.legend()
```

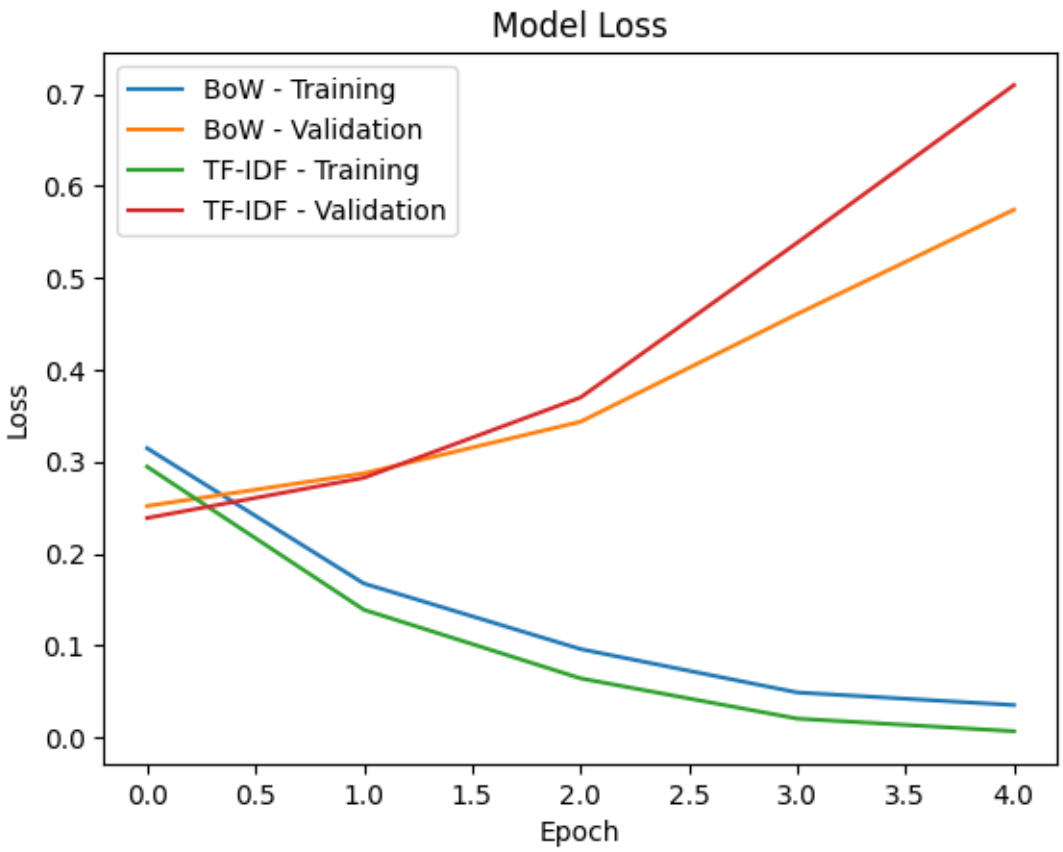
[19]:
```
<matplotlib.legend.Legend at 0x1adc56a8b80>
```



Loss

[20]:
```
plt.figure()
plt.plot(history_bow.history['loss'], label='BoW - Training')
plt.plot(history_bow.history['val_loss'], label='BoW - Validation')
plt.plot(history_tfidf.history['loss'], label='TF-IDF - Training')
plt.plot(history_tfidf.history['val_loss'], label='TF-IDF - Validation')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Model Loss')
plt.legend()
```

[20]:
```
<matplotlib.legend.Legend at 0x1adc5ebb700>
```


Model Loss
BoW - Training
BoW - Validation
TF-IDF - Training
TF-IDF - Validation
Loss
Epoch
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

[ ]: