Doc2vec

Created by: Harsh Bari

From: SVNIT, Gujarat

Mtech Data Science - p23ds004 (2023-25)

Subject: NLP Project

Last Updated: 29/03/2024

In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
data = pd.read_csv("resultant_train_dataframe_with_doc2vec.csv")
```

In [3]:
```
data
```

Out[3]:

	tweets_preprocessed	class	tokens	doc2vec_1	doc2vec_2	doc2vec_3	c
0	aware dirty step get money staylight staywhite...	figurative	['aware', 'dirty', 'step', 'get', 'money', 'st...	0.094989	-0.130547	0.209554	
1	sarcasm people nt understand diy artattack htt...	figurative	['sarcasm', 'people', 'nt', 'understand', 'diy...	0.141715	-0.156211	0.124129	
2	iminworkjeremy medsingle dailymail reader sens...	figurative	['iminworkjeremy', 'medsingle', 'dailymail', '...	0.071912	-0.202637	0.029908	
3	wilw get feeling like game sarcasm	figurative	['wilw', 'get', 'feeling', 'like', 'game', 'sa...	-0.180418	0.049932	-0.014645	
4	teacherarthurg rweingarten probably missed tex...	figurative	['teacherarthurg', 'rweingarten', 'probably', ...	0.315940	-0.020790	-0.059559	
...	...	...	...	...	...	...	
81403	photo image via heart http tcoky8nf8z9oi child...	sarcasm	['photo', 'image', 'via', 'heart', 'http', 'tc...	-0.140894	0.102283	0.163148	
81404	never knew better put universe lol maybe date ...	sarcasm	['never', 'knew', 'better', 'put', 'universe',...	-0.286387	0.332654	0.163479	
81405	hey wanted say thanks puberty letting apart it...	sarcasm	['hey', 'wanted', 'say', 'thanks', 'puberty', ...	0.305770	-0.039960	-0.050912	
81406	sure coverage like fox news special hidden har...	sarcasm	['sure', 'coverage', 'like', 'fox', 'news', 's...	0.245714	-0.104332	-0.617117	
81407	skeyno16 u13 wo nt believe see p sarcasm	sarcasm	['skeyno16', 'u13', 'wo', 'nt', 'believe', 'se...	-0.085722	-0.135554	-0.029625	

81408 rows × 153 columns

In [4]:
```
vec = ['doc2vec_{}'.format(i) for i in range(1, 151)]
input_vec = data[vec].values
```

In [5]:
```
# vec
```

Split Data

In [6]:
```
input_vec = np.array(input_vec)
```

In [7]:
```
def split_data(array_2d, ranges_to_copy):
    copied_ranges = []

    # Loop through each range and copy the corresponding elements
    for start, end in ranges_to_copy:
        copied_range = array_2d[start:end+1]  # Adjust end index to include
        copied_ranges.append(copied_range)

    # Concatenate the copied ranges along the first axis to create the fina
    copied_array = np.concatenate(copied_ranges, axis=0)

    return copied_array
```

In [8]:
```
x_train = split_data(input_vec, [(0, 16989), (21238, 37952), (42132, 57007)
x_test = split_data(input_vec, [(16990, 21237), (37953, 42131), (57008, 607
```

In [9]:
```
print("x train:", len(x_train))
print("x test:", len(x_test))
print("Total:", len(x_train) + len(x_test))
```

```
x train: 65125
x test: 16283
Total: 81408
```

In [10]:
```
y_train = np.concatenate((np.zeros(16990), np.ones(31591), np.zeros(16544))
y_test = np.concatenate((np.zeros(4248), np.ones(7898), np.zeros(4137)))
```

In [11]:
```
print("train:", len(y_train))
print("test:", len(y_test))
print("total:", len(y_train) + len(y_test))
```

```
train: 65125
test: 16283
total: 81408
```

Training With Neural Network

In [12]:
```
import tensorflow as tf
from tensorflow import keras
```

```
WARNING:tensorflow:From C:\Users\Harsh Bari\AppData\Local\Programs\Python
\Python310\lib\site-packages\keras\src\losses.py:2976: The name tf.losses.
sparse_softmax_cross_entropy is deprecated. Please use tf.compat.v1.losse
s.sparse_softmax_cross_entropy instead.
```

Neural Network for Average Word Embedding

In [13]:
```
d2v = keras.Sequential([
    keras.layers.Dense(256, input_shape = (150, ), activation = 'relu'),
    keras.layers.Dense(128, activation = 'relu'),
    keras.layers.Dense(64, activation = 'relu'),
    keras.layers.Dense(32, activation = 'relu'),
    keras.layers.Dense(64, activation=keras.layers.LeakyReLU(alpha=0.1)),
    keras.layers.Dense(32, activation=keras.layers.LeakyReLU(alpha=0.1)),
    keras.layers.Dense(2, activation = 'sigmoid')
])

d2v.compile(optimizer = 'adam',
                      loss = 'sparse_categorical_crossentropy',
                      metrics = ['accuracy'])
```

```
WARNING:tensorflow:From C:\Users\Harsh Bari\AppData\Local\Programs\Python
\Python310\lib\site-packages\keras\src\backend.py:873: The name tf.get_def
ault_graph is deprecated. Please use tf.compat.v1.get_default_graph instea
d.

WARNING:tensorflow:From C:\Users\Harsh Bari\AppData\Local\Programs\Python
\Python310\lib\site-packages\keras\src\optimizers\__init__.py:309: The nam
e tf.train.Optimizer is deprecated. Please use tf.compat.v1.train.Optimize
r instead.
```

keras.layers.Dense(110, activation = 'relu'), keras.layers.Dense(80, activation=keras.layers.LeakyReLU(alpha=0.1)),

Check Model Summary

In [14]:
```
d2v.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense (Dense)               (None, 256)               38656     
                                                                 
 dense_1 (Dense)             (None, 128)               32896     
                                                                 
 dense_2 (Dense)             (None, 64)                8256      
                                                                 
 dense_3 (Dense)             (None, 32)                2080      
                                                                 
 dense_4 (Dense)             (None, 64)                2112      
                                                                 
 dense_5 (Dense)             (None, 32)                2080      
                                                                 
 dense_6 (Dense)             (None, 2)                 66        
                                                                 
=================================================================
Total params: 86146 (336.51 KB)
Trainable params: 86146 (336.51 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Train Model

```
In [15]: d2v.fit(x_train.astype(np.float32), y_train.astype(np.float32), epochs=22)
```

```
Epoch 1/22
WARNING:tensorflow:From C:\Users\Harsh Bari\AppData\Local\Programs\Python
\Python310\lib\site-packages\keras\src\utils\tf_utils.py:492: The name tf.
ragged.RaggedTensorValue is deprecated. Please use tf.compat.v1.ragged.Rag
gedTensorValue instead.

WARNING:tensorflow:From C:\Users\Harsh Bari\AppData\Local\Programs\Python
\Python310\lib\site-packages\keras\src\engine\base_layer_utils.py:384: The
name tf.executing_eagerly_outside_functions is deprecated. Please use tf.c
ompat.v1.executing_eagerly_outside_functions instead.

2036/2036 [==============================] - 9s 3ms/step - loss: 0.4561 -
accuracy: 0.7652
Epoch 2/22
2036/2036 [==============================] - 8s 4ms/step - loss: 0.3947 -
accuracy: 0.7964
Epoch 3/22
2036/2036 [==============================] - 10s 5ms/step - loss: 0.3678 -
accuracy: 0.8101
Epoch 4/22
2036/2036 [==============================] - 8s 4ms/step - loss: 0.3453 -
accuracy: 0.8208
Epoch 5/22
2036/2036 [==============================] - 7s 3ms/step - loss: 0.3273 -
accuracy: 0.8287
Epoch 6/22
2036/2036 [==============================] - 8s 4ms/step - loss: 0.3120 -
accuracy: 0.8355
Epoch 7/22
2036/2036 [==============================] - 7s 3ms/step - loss: 0.2977 -
accuracy: 0.8415
Epoch 8/22
2036/2036 [==============================] - 9s 4ms/step - loss: 0.2861 -
accuracy: 0.8466
Epoch 9/22
2036/2036 [==============================] - 9s 4ms/step - loss: 0.2744 -
accuracy: 0.8505
Epoch 10/22
2036/2036 [==============================] - 8s 4ms/step - loss: 0.2652 -
accuracy: 0.8546
Epoch 11/22
2036/2036 [==============================] - 13s 6ms/step - loss: 0.2586 -
accuracy: 0.8576
Epoch 12/22
2036/2036 [==============================] - 11s 5ms/step - loss: 0.2511 -
accuracy: 0.8598
Epoch 13/22
2036/2036 [==============================] - 11s 5ms/step - loss: 0.2418 -
accuracy: 0.8637
Epoch 14/22
2036/2036 [==============================] - 8s 4ms/step - loss: 0.2364 -
accuracy: 0.8661
Epoch 15/22
2036/2036 [==============================] - 9s 5ms/step - loss: 0.2321 -
accuracy: 0.8679
Epoch 16/22
2036/2036 [==============================] - 10s 5ms/step - loss: 0.2266 -
accuracy: 0.8706
Epoch 17/22
2036/2036 [==============================] - 7s 4ms/step - loss: 0.2217 -
accuracy: 0.8726
```

```
Epoch 18/22
2036/2036 [==============================] - 7s 3ms/step - loss: 0.2176 -
accuracy: 0.8757
Epoch 19/22
2036/2036 [==============================] - 8s 4ms/step - loss: 0.2119 -
accuracy: 0.8773
Epoch 20/22
2036/2036 [==============================] - 7s 3ms/step - loss: 0.2078 -
accuracy: 0.8801
Epoch 21/22
2036/2036 [==============================] - 7s 4ms/step - loss: 0.2043 -
accuracy: 0.8816
Epoch 22/22
2036/2036 [==============================] - 10s 5ms/step - loss: 0.2009 -
accuracy: 0.8839
```

Out[15]:
```
<keras.src.callbacks.History at 0x24968989a80>
```

Training Accuracy

In [16]:
```
d2v.evaluate(x_train.astype(np.float32), y_train.astype(np.float32))
```

```
2036/2036 [==============================] - 6s 3ms/step - loss: 0.1951 -
accuracy: 0.8917
```

Out[16]:
```
[0.19513297080993652, 0.8917466402053833]
```

Testing Accuracy

In [17]:
```
prediction = d2v.predict(x_test.astype(np.float32))
```

```
509/509 [==============================] - 1s 2ms/step
```

In [18]:
```
prediction = np.argmax(prediction, axis = 1)
```

In [19]:
```
from sklearn.metrics import classification_report, confusion_matrix, accura
```

In [20]:
```
print(classification_report(y_test.astype(np.float32), prediction))
print()
print("Confusion Matrix: \n", confusion_matrix(y_test.astype(np.float32), p
print("\nAccuracy: \n", accuracy_score(y_test.astype(np.float32), predictio
```

```
              precision    recall  f1-score   support

         0.0       0.84      0.76      0.80      8385
         1.0       0.77      0.85      0.81      7898

    accuracy                           0.81     16283
   macro avg       0.81      0.81      0.81     16283
weighted avg       0.81      0.81      0.81     16283


Confusion Matrix: 
 [[6414 1971]
 [1198 6700]]

Accuracy: 
 0.8053798440090892
```