BDP Difference Vector

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("data.csv")
```

In [3]:
```
print(data)
```

```
                                                  tweets       class  targ
et
0      Be aware  dirty step to get money  #staylight ...  figurative  
0.0
1      #sarcasm for #people who don't understand #diy...  figurative  
0.0
2      @IminworkJeremy @medsingle #DailyMail readers ...  figurative  
0.0
3      @wilw Why do I get the feeling you like games?...  figurative  
0.0
4      -@TeacherArthurG @rweingarten You probably jus...  figurative  
0.0
...                                                  ...         ...  
...
81403  Photo: Image via We Heart It http://t.co/ky8Nf... (http://t.co/ky8N
f...)     sarcasm     1.0
81404  I never knew..I better put this out to the Uni...     sarcasm  
1.0
81405  hey just wanted to say thanks @ puberty for le...     sarcasm  
1.0
81406  I'm sure coverage like the Fox News Special “T...     sarcasm  
1.0
81407  @skeyno16 at u13?! I won't believe it until I ...     sarcasm  
1.0

[81408 rows x 3 columns]
```

In [4]:
```
data['class'].value_counts()
```

Out[4]:
```
class
figurative    21238
irony         20894
sarcasm       20681
regular       18595
Name: count, dtype: int64
```

BDP (Base Difference Protocol) Difference

In [5]:
```
import bdp_difference_vector as bdp
```

In [6]:
```
bdp_train_vectors = bdp.get_vectorized(data['tweets'])
```

```
[#=================================================] 0.2%

[##################################################] 100.0%
```

In [7]:
```
from sklearn.preprocessing import normalize

input_vec = normalize(bdp_train_vectors, axis=0)
```

In [8]:
```
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 [9]:
```
# x_train = split_data(input_vec, [(0, 14865), (21238, 35862), (42132, 5514
# x_test = split_data(input_vec, [(14866, 21237), (35863, 42131), (55148, 6
```

In [10]:
```
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 [11]:
```
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 [12]:
```
# y_train = np.concatenate((np.zeros(14866), np.ones(27641), np.zeros(14476
# y_test = np.concatenate((np.zeros(6372), np.ones(11848), np.zeros(6205)))
```

In [13]:
```
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 [14]:
```
print("train:", len(y_train))
print("test:", len(y_test))
print("total:", len(y_train) + len(y_test))
```

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

BDP Neural Network

In [15]:
```
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.
```

Create BDP Neural Network

In [16]:
```
bdp = 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(16, activation=keras.layers.LeakyReLU(alpha=0.1)),
    keras.layers.Dense(4, activation=keras.layers.LeakyReLU(alpha=0.1)),
    keras.layers.Dense(2, activation = 'sigmoid')

])

bdp.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(90, activation = 'relu'), keras.layers.Dense(80, activation=keras.layers.LeakyReLU(alpha=0.1)),

Check Model Summary

```
In [17]: bdp.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, 16)                528       
                                                                 
 dense_5 (Dense)             (None, 4)                 68        
                                                                 
 dense_6 (Dense)             (None, 2)                 10        
                                                                 
=================================================================
Total params: 82494 (322.24 KB)
Trainable params: 82494 (322.24 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Train Model

In [18]:

```
bdp.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 [==============================] - 10s 4ms/step - loss: 0.4559 -
accuracy: 0.7737
Epoch 2/22
2036/2036 [==============================] - 8s 4ms/step - loss: 0.3932 -
accuracy: 0.8012
Epoch 3/22
2036/2036 [==============================] - 8s 4ms/step - loss: 0.3746 -
accuracy: 0.8072
Epoch 4/22
2036/2036 [==============================] - 9s 4ms/step - loss: 0.3668 -
accuracy: 0.8103
Epoch 5/22
2036/2036 [==============================] - 9s 4ms/step - loss: 0.3593 -
accuracy: 0.8133
Epoch 6/22
2036/2036 [==============================] - 9s 4ms/step - loss: 0.3550 -
accuracy: 0.8145
Epoch 7/22
2036/2036 [==============================] - 9s 4ms/step - loss: 0.3504 -
accuracy: 0.8182
Epoch 8/22
2036/2036 [==============================] - 9s 4ms/step - loss: 0.3475 -
accuracy: 0.8185
Epoch 9/22
2036/2036 [==============================] - 9s 4ms/step - loss: 0.3437 -
accuracy: 0.8203
Epoch 10/22
2036/2036 [==============================] - 9s 4ms/step - loss: 0.3412 -
accuracy: 0.8215
Epoch 11/22
2036/2036 [==============================] - 9s 4ms/step - loss: 0.3377 -
accuracy: 0.8227
Epoch 12/22
2036/2036 [==============================] - 9s 4ms/step - loss: 0.3359 -
accuracy: 0.8244
Epoch 13/22
2036/2036 [==============================] - 8s 4ms/step - loss: 0.3336 -
accuracy: 0.8260
Epoch 14/22
2036/2036 [==============================] - 9s 4ms/step - loss: 0.3309 -
accuracy: 0.8260
Epoch 15/22
2036/2036 [==============================] - 8s 4ms/step - loss: 0.3286 -
accuracy: 0.8271
Epoch 16/22
2036/2036 [==============================] - 9s 4ms/step - loss: 0.3260 -
accuracy: 0.8286
Epoch 17/22
2036/2036 [==============================] - 8s 4ms/step - loss: 0.3243 -
accuracy: 0.8287
```

```
Epoch 18/22
2036/2036 [==============================] - 8s 4ms/step - loss: 0.3230 -
accuracy: 0.8302
Epoch 19/22
2036/2036 [==============================] - 8s 4ms/step - loss: 0.3210 -
accuracy: 0.8308
Epoch 20/22
2036/2036 [==============================] - 8s 4ms/step - loss: 0.3195 -
accuracy: 0.8313
Epoch 21/22
2036/2036 [==============================] - 8s 4ms/step - loss: 0.3170 -
accuracy: 0.8325
Epoch 22/22
2036/2036 [==============================] - 9s 4ms/step - loss: 0.3146 -
accuracy: 0.8337
```

Out[18]:

```
<keras.src.callbacks.History at 0x25792c83940>
```

Training Accuracy

In [19]:

```
bdp.evaluate(x_train.astype(np.float32), y_train.astype(np.float32))
```

```
2036/2036 [==============================] - 5s 2ms/step - loss: 0.3058 -
accuracy: 0.8383
```

Out[19]:

```
[0.3058025538921356, 0.838310956954956]
```

Testing Accuracy

In [20]:

```
prediction = bdp.predict(x_test.astype(np.float32))
```

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

In [21]:

```
prediction = np.argmax(prediction, axis = 1)
```

In [22]:

```
from sklearn.metrics import classification_report, confusion_matrix, accura
```

In [23]:

```
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       0.93      0.76      0.84      8385
         1.0       0.79      0.94      0.86      7898

    accuracy                           0.85     16283
   macro avg       0.86      0.85      0.85     16283
weighted avg       0.86      0.85      0.85     16283


Confusion Matrix: 
 [[6391 1994]
 [ 475 7423]]

Accuracy: 
 0.8483694650862863
```