Twitter data analysis using Word2Vec and Neural network method

```
from google.colab import drive
drive.mount('/content/drive')
```

```
     Mounted at /content/drive
```

```
import pandas as pd
df=pd.read_csv('/content/drive/MyDrive/train.csv')
df.shape
```

```
     (31962, 3)
```

```
import numpy as np
import re
import nltk
```

```
df.head(10)
```

	id	label	tweet
0	1	0	when father dysfunctional selfish drags kids i...

```
def remove_pattern(input_txt, pattern):
    r = re.findall(pattern, input_txt)
    for i in r:
        input_txt = re.sub(i, '', input_txt)

    return input_txt
```

5	6	0	huge fare talking before they leave chaos disp...

```
# To remove unwanted characters
df['tweet'] = np.vectorize(remove_pattern)(df['tweet'], "@[\w]*")
df['tweet'] = df['tweet'].str.replace("[^a-zA-Z#]", " ")

df.head()
```

	id	label	tweet
0	1	0	when father dysfunctional selfish drags kids i...
1	2	0	thanks #lyft credit cause they offer wheelchai...
2	3	0	bihday your majesty
3	4	0	#model love take with time
4	5	0	factsguide society #motivation

```
df['tweet'] = df['tweet'].apply(lambda x: ' '.join([w for w in x.split() if len(w)>3]))
df.head()
```

	id	label	tweet
0	1	0	when father dysfunctional selfish drags kids i...
1	2	0	thanks #lyft credit cause they offer wheelchai...

```
# Tokenization
x = df['tweet'].apply(lambda x: x.split())
x
```

```
0        [when, father, dysfunctional, selfish, drags, ...
1        [thanks, #lyft, credit, cause, they, offer, wh...
2                                [bihday, your, majesty]
3                       [#model, love, take, with, time]
4                     [factsguide, society, #motivation]
                               ...
31957                                      [that, youuu]
31958    [nina, turner, airwaves, trying, wrap, herself...
31959            [listening, songs, monday, morning, work]
31960    [#sikh, #temple, vandalised, #calgary, #wso, c...
31961                                    [thank, follow]
Name: tweet, Length: 31962, dtype: object
```

```
# Stemming
from nltk.stem.porter import *
stemmer = PorterStemmer()
x = x.apply(lambda x: [stemmer.stem(i) for i in x])
x
```

```
0        [when, father, dysfunct, selfish, drag, kid, i...
1        [thank, #lyft, credit, caus, they, offer, whee...
2                                [bihday, your, majesti]
3                       [#model, love, take, with, time]
4                           [factsguid, societi, #motiv]
                               ...
31957                                      [that, youuu]
31958    [nina, turner, airwav, tri, wrap, herself, man...
31959                 [listen, song, monday, morn, work]
31960    [#sikh, #templ, vandalis, #calgari, #wso, cond...
```

```
31961                                      [thank, follow]
Name: tweet, Length: 31962, dtype: object
```

```
from gensim.models import Word2Vec
model_w2v = Word2Vec(x, min_count=1,size=30)
model_w2v.train(x, total_examples= len(x), epochs=20)
```

```
(4640849, 4904460)
```

```
model_w2v.most_similar("father")
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:1: DeprecationWarning: Call to deprecated `most_similar`
  """Entry point for launching an IPython kernel.
/usr/local/lib/python3.6/dist-packages/gensim/matutils.py:737: FutureWarning: Conversion of the second argument of iss
  if np.issubdtype(vec.dtype, np.int):
[('dad', 0.8887248039245605),
 ('daddi', 0.8561842441558838),
 ('#fathersday', 0.8322489261627197),
 ('fathersday', 0.8269415497779846),
 ('#father', 0.8206629157066345),
 ('day#com', 0.8106535077095032),
 ('#dad', 0.8073040246963501),
 ('gift', 0.7992919683456421),
 ('#grant', 0.7967487573623657),
 ('thnku', 0.7964769005775452)]
```

```
# Converting words into vectors
def word_vector(tokens, size):
    vec = np.zeros(size).reshape((1, size))
    count = 0.
    for word in tokens:
        try:
            vec += model_w2v[word].reshape((1, size))
            count += 1.
        except KeyError: # handling the case where the token is not in vocabulary
```

```
            continue
    if count != 0:
        vec /= count
    return vec


wordvec_arrays = np.zeros((len(x), 30))

for i in range(len(x)):
    wordvec_arrays[i,:] = word_vector(x[i], 30)

wordvec_df = pd.DataFrame(wordvec_arrays)
wordvec_df.shape
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:7: DeprecationWarning: Call to deprecated `__getitem__` (
  import sys
(31962, 30)
```

```
wordvec_df
```

	0	1	2	3	4	5	6	7	8	9	10
0	-1.283710	0.359937	-1.306183	-0.168716	0.546240	0.048362	1.239968	1.992669	0.504369	0.252197	-0.319099
1	-1.032753	0.193549	-0.960366	0.012670	0.149221	0.520383	0.466089	1.499934	0.688582	0.442480	-0.686092

```
y=df['label']
```

```
x = wordvec_df
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.33, random_state=42)
```

31958	-0.540916	0.402284	-0.877268	0.080874	0.608549	0.339064	0.179932	0.375324	0.742814	0.284399	-0.516893

```
import tensorflow as tf
import keras.layers as layers
from keras.models import Model
from keras.datasets import imdb
from keras.layers import Input,Embedding,Dense,Flatten
from sklearn.metrics import accuracy_score,classification_report
from sklearn.metrics import f1_score

epochs = 25
batch_size = 1024
loss = "binary_crossentropy"
optimizer = "adam"
metrics = ["accuracy"]

from keras import models


# Build neural network
model = models.Sequential()
model.add(Dense(512, activation='relu', input_shape=(30,)))
model.add(Dense(512, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.summary()
```

```
Model: "sequential_11"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_25 (Dense)             (None, 512)               15872     
_________________________________________________________________
dense_26 (Dense)             (None, 512)               262656    
_________________________________________________________________
dense_27 (Dense)             (None, 1)                 513       
=================================================================
Total params: 279,041
Trainable params: 279,041
Non-trainable params: 0
_________________________________________________________________
```

```
model.compile(loss=loss,optimizer=optimizer,metrics= metrics)
model.fit(X_train,y_train,epochs=epochs,batch_size=batch_size,validation_data=(X_test,y_test))
```

```
Epoch 1/25
21/21 [==============================] - 1s 48ms/step - loss: 0.2624 - accuracy: 0.9150 - val_loss: 0.1867 - val_accur
Epoch 2/25
21/21 [==============================] - 1s 41ms/step - loss: 0.1722 - accuracy: 0.9353 - val_loss: 0.1671 - val_accur
Epoch 3/25
21/21 [==============================] - 1s 40ms/step - loss: 0.1587 - accuracy: 0.9404 - val_loss: 0.1596 - val_accur
Epoch 4/25
21/21 [==============================] - 1s 41ms/step - loss: 0.1515 - accuracy: 0.9431 - val_loss: 0.1554 - val_accur
Epoch 5/25
21/21 [==============================] - 1s 40ms/step - loss: 0.1477 - accuracy: 0.9441 - val_loss: 0.1531 - val_accur
Epoch 6/25
21/21 [==============================] - 1s 39ms/step - loss: 0.1438 - accuracy: 0.9456 - val_loss: 0.1530 - val_accur
Epoch 7/25
21/21 [==============================] - 1s 39ms/step - loss: 0.1416 - accuracy: 0.9467 - val_loss: 0.1501 - val_accur
Epoch 8/25
21/21 [==============================] - 1s 40ms/step - loss: 0.1379 - accuracy: 0.9479 - val_loss: 0.1502 - val_accur
Epoch 9/25
21/21 [==============================] - 1s 40ms/step - loss: 0.1368 - accuracy: 0.9477 - val_loss: 0.1480 - val_accur
Epoch 10/25
21/21 [==============================] - 1s 40ms/step - loss: 0.1333 - accuracy: 0.9491 - val_loss: 0.1481 - val_accur
Epoch 11/25
21/21 [==============================] - 1s 40ms/step - loss: 0.1306 - accuracy: 0.9507 - val_loss: 0.1467 - val_accur
```

```
Epoch 12/25
21/21 [==============================] - 1s 40ms/step - loss: 0.1290 - accuracy: 0.9510 - val_loss: 0.1477 - val_accur
Epoch 13/25
21/21 [==============================] - 1s 39ms/step - loss: 0.1256 - accuracy: 0.9522 - val_loss: 0.1457 - val_accur
Epoch 14/25
21/21 [==============================] - 1s 40ms/step - loss: 0.1231 - accuracy: 0.9522 - val_loss: 0.1478 - val_accur
Epoch 15/25
21/21 [==============================] - 1s 40ms/step - loss: 0.1201 - accuracy: 0.9543 - val_loss: 0.1469 - val_accur
Epoch 16/25
21/21 [==============================] - 1s 40ms/step - loss: 0.1176 - accuracy: 0.9552 - val_loss: 0.1463 - val_accur
Epoch 17/25
21/21 [==============================] - 1s 39ms/step - loss: 0.1141 - accuracy: 0.9562 - val_loss: 0.1464 - val_accur
Epoch 18/25
21/21 [==============================] - 1s 40ms/step - loss: 0.1142 - accuracy: 0.9559 - val_loss: 0.1486 - val_accur
Epoch 19/25
21/21 [==============================] - 1s 39ms/step - loss: 0.1133 - accuracy: 0.9561 - val_loss: 0.1492 - val_accur
Epoch 20/25
21/21 [==============================] - 1s 39ms/step - loss: 0.1104 - accuracy: 0.9578 - val_loss: 0.1485 - val_accur
Epoch 21/25
21/21 [==============================] - 1s 40ms/step - loss: 0.1072 - accuracy: 0.9572 - val_loss: 0.1491 - val_accur
Epoch 22/25
21/21 [==============================] - 1s 40ms/step - loss: 0.1030 - accuracy: 0.9601 - val_loss: 0.1494 - val_accur
Epoch 23/25
21/21 [==============================] - 1s 40ms/step - loss: 0.1001 - accuracy: 0.9614 - val_loss: 0.1534 - val_accur
Epoch 24/25
21/21 [==============================] - 1s 40ms/step - loss: 0.0984 - accuracy: 0.9622 - val_loss: 0.1524 - val_accur
Epoch 25/25
21/21 [==============================] - 1s 39ms/step - loss: 0.0960 - accuracy: 0.9629 - val_loss: 0.1532 - val_accur
<tensorflow.python.keras.callbacks.History at 0x7f6f73976128>
```

```
y_pred = model.predict(X_test)
```

```
y_pred = [0 if i<0.5 else 1 for i in y_pred]
```

```
# Confusion matrix
from sklearn.metrics import confusion_matrix
```

```
confusion_matrix(y_test,y_pred)
```

```
array([[9704,  102],
       [ 463,  279]])
```

```
# Finding data accuracy
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
```

```
0.9464353431930224
```