Twitter sentimental

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import nltk
import warnings
warnings.filterwarnings('ignore')
from sklearn.model_selection import train_test_split
import tensorflow as tf
from nltk.tokenize import word_tokenize
import nltk
nltk.download('punkt')
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense , Input , LSTM , Embedding,
Dropout , Activation, Flatten,Bidirectional
```

```
----------------------------------------------------------------------
-----
ModuleNotFoundError                      Traceback (most recent call
last)
Input In [4], in <cell line: 8>()
      6 warnings.filterwarnings('ignore')
      7 from sklearn.model_selection import train_test_split
----> 8 import tensorflow as tf
      9 from nltk.tokenize import word_tokenize
     10 import nltk

ModuleNotFoundError: No module named 'tensorflow'
```

```
# reading data

data=pd.read_csv('Twitter_Data.csv')
data.head()
```

```
----------------------------------------------------------------------
-----
FileNotFoundError                        Traceback (most recent call
last)
Input In [5], in <cell line: 1>()
----> 1 data=pd.read_csv('Twitter_Data.csv')
      2 data.head()

File C:\ProgramData\Anaconda3\lib\site-packages\pandas\util\
_decorators.py:311, in
deprecate_nonkeyword_arguments.<locals>.decorate.<locals>.wrapper(*arg
s, **kwargs)
    305 if len(args) > num_allow_args:
    306     warnings.warn(
    307         msg.format(arguments=arguments),
    308         FutureWarning,
    309         stacklevel=stacklevel,
    310     )
--> 311 return func(*args, **kwargs)

File C:\ProgramData\Anaconda3\lib\site-packages\pandas\io\parsers\
readers.py:680, in read_csv(filepath_or_buffer, sep, delimiter,
header, names, index_col, usecols, squeeze, prefix, mangle_dupe_cols,
dtype, engine, converters, true_values, false_values,
skipinitialspace, skiprows, skipfooter, nrows, na_values,
keep_default_na, na_filter, verbose, skip_blank_lines, parse_dates,
infer_datetime_format, keep_date_col, date_parser, dayfirst,
cache_dates, iterator, chunksize, compression, thousands, decimal,
lineterminator, quotechar, quoting, doublequote, escapechar, comment,
encoding, encoding_errors, dialect, error_bad_lines, warn_bad_lines,
on_bad_lines, delim_whitespace, low_memory, memory_map,
float_precision, storage_options)
    665 kwds_defaults = _refine_defaults_read(
    666     dialect,
    667     delimiter,
   (...)
    676     defaults={"delimiter": ","},
    677 )
    678 kwds.update(kwds_defaults)
--> 680 return _read(filepath_or_buffer, kwds)

File C:\ProgramData\Anaconda3\lib\site-packages\pandas\io\parsers\
readers.py:575, in _read(filepath_or_buffer, kwds)
    572 _validate_names(kwds.get("names", None))
    574 # Create the parser.
--> 575 parser = TextFileReader(filepath_or_buffer, **kwds)
    577 if chunksize or iterator:
    578     return parser

File C:\ProgramData\Anaconda3\lib\site-packages\pandas\io\parsers\
readers.py:933, in TextFileReader.__init__(self, f, engine, **kwds)
    930     self.options["has_index_names"] = kwds["has_index_names"]
    932 self.handles: IOHandles | None = None
--> 933 self._engine = self._make_engine(f, self.engine)

File C:\ProgramData\Anaconda3\lib\site-packages\pandas\io\parsers\
readers.py:1217, in TextFileReader._make_engine(self, f, engine)
   1213     mode = "rb"
   1214 # error: No overload variant of "get_handle" matches argument
types
   1215 # "Union[str, PathLike[str], ReadCsvBuffer[bytes],
ReadCsvBuffer[str]]"
   1216 # , "str", "bool", "Any", "Any", "Any", "Any", "Any"
-> 1217 self.handles = get_handle(  # type: ignore[call-overload]
   1218     f,
   1219     mode,
   1220     encoding=self.options.get("encoding", None),
   1221     compression=self.options.get("compression", None),
   1222     memory_map=self.options.get("memory_map", False),
   1223     is_text=is_text,
   1224     errors=self.options.get("encoding_errors", "strict"),
   1225     storage_options=self.options.get("storage_options", None),
   1226 )
   1227 assert self.handles is not None
   1228 f = self.handles.handle

File C:\ProgramData\Anaconda3\lib\site-packages\pandas\io\
common.py:789, in get_handle(path_or_buf, mode, encoding, compression,
memory_map, is_text, errors, storage_options)
    784 elif isinstance(handle, str):
    785     # Check whether the filename is to be opened in binary
mode.
    786     # Binary mode does not support 'encoding' and 'newline'.
    787     if ioargs.encoding and "b" not in ioargs.mode:
    788         # Encoding
--> 789         handle = open(
    790             handle,
    791             ioargs.mode,
    792             encoding=ioargs.encoding,
    793             errors=errors,
    794             newline="",
    795         )
    796     else:
    797         # Binary mode
    798         handle = open(handle, ioargs.mode)

FileNotFoundError: [Errno 2] No such file or directory:
'Twitter_Data.csv'
```

```
data.shape
```

```
(32004, 2)
```

```
df=data
```

Change our dependent variable to categorical. (0 to “Neutral,”-1 to “Negative”, 1 to “Positive”)

```
df['category'].loc[df['category']==-1.0]="negative"
df['category'].loc[df['category']==0.0]="neutral"
df['category'].loc[df['category']==1.0]="positive"
df.head()
```

```
                                           clean_text  category
0  when modi promised “minimum government maximum...  negative
```

```
1  talk all the nonsense and continue all the dra...   neutral
2  what did just say vote for modi  welcome bjp t...  positive
3  asking his supporters prefix chowkidar their n...  positive
4  answer who among these the most powerful world...  positive
```

Do Missing value analysisand drop all null/missing values

```
df.isnull().sum()
```

```
clean_text    1
category      1
dtype: int64
```

```
df=df.dropna()
```

```
df.isnull().sum()
```

```
clean_text    0
category      0
dtype: int64
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 32002 entries, 0 to 32002
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   clean_text  32002 non-null  object
 1   category    32002 non-null  object
dtypes: object(2)
memory usage: 750.0+ KB
```

```
#doing text cleaning (removing every symbol except alphnumeric and 
converting all words to lower case)

df['lower_text'] = df['clean_text'].str.lower()

df['stopped_text'] = df['clean_text'].str.replace('\W', ' ', 
regex=True)
df.head()
```

```
                                          clean_text  category  \
0  when modi promised “minimum government maximum...  negative   
1  talk all the nonsense and continue all the dra...   neutral   
2  what did just say vote for modi  welcome bjp t...  positive   
3  asking his supporters prefix chowkidar their n...  positive   
4  answer who among these the most powerful world...  positive   

                                          lower_text  \
0  when modi promised “minimum government maximum...   
1  talk all the nonsense and continue all the dra...   
2  what did just say vote for modi  welcome bjp t...   
```

```
3  asking his supporters prefix chowkidar their n...
4  answer who among these the most powerful world...

                                        stopped_text
0  when modi promised  minimum government maximum...
1  talk all the nonsense and continue all the dra...
2  what did just say vote for modi  welcome bjp t...
3  asking his supporters prefix chowkidar their n...
4  answer who among these the most powerful world...
```

```
df['tokenized'] = df['stopped_text'].apply(word_tokenize)
df=df.drop(['lower_text','stopped_text'],axis=1)
```

```
df.head()
```

```
                                          clean_text  category  \
0  when modi promised “minimum government maximum...  negative
1  talk all the nonsense and continue all the dra...   neutral
2  what did just say vote for modi  welcome bjp t...  positive
3  asking his supporters prefix chowkidar their n...  positive
4  answer who among these the most powerful world...  positive

                                           tokenized
0  [when, modi, promised, minimum, government, ma...
1  [talk, all, the, nonsense, and, continue, all,...
2  [what, did, just, say, vote, for, modi, welcom...
3  [asking, his, supporters, prefix, chowkidar, t...
4  [answer, who, among, these, the, most, powerfu...
```

```
nltk.download('stopwords')
from nltk.corpus import stopwords
stop = stopwords.words('english')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
stop_words = set(stopwords.words("english"))
df['stopped_text'] = df['tokenized'].apply(
    lambda x: [word for word in x if word not in stop_words])
df.head()
```

```
                                          clean_text  category  \
0  when modi promised “minimum government maximum...  negative
1  talk all the nonsense and continue all the dra...   neutral
2  what did just say vote for modi  welcome bjp t...  positive
3  asking his supporters prefix chowkidar their n...  positive
4  answer who among these the most powerful world...  positive

                                           tokenized  \
0  [when, modi, promised, minimum, government, ma...
1  [talk, all, the, nonsense, and, continue, all,...
```

```
2  [what, did, just, say, vote, for, modi, welcom...
3  [asking, his, supporters, prefix, chowkidar, t...
4  [answer, who, among, these, the, most, powerfu...

                                        stopped_text
0  [modi, promised, minimum, government, maximum,...
1      [talk, nonsense, continue, drama, vote, modi]
2  [say, vote, modi, welcome, bjp, told, rahul, m...
3  [asking, supporters, prefix, chowkidar, names,...
4  [answer, among, powerful, world, leader, today...
```

Creating new column and finding the length of each sentence (how many words they contain)

```
df['length']=df['clean_text'].str.split().str.len()
```

```
df.head()
```

```
                                          clean_text  category  \
0  when modi promised “minimum government maximum...  negative
1  talk all the nonsense and continue all the dra...   neutral
2  what did just say vote for modi  welcome bjp t...  positive
3  asking his supporters prefix chowkidar their n...  positive
4  answer who among these the most powerful world...  positive

                                           tokenized  \
0  [when, modi, promised, minimum, government, ma...
1  [talk, all, the, nonsense, and, continue, all,...
2  [what, did, just, say, vote, for, modi, welcom...
3  [asking, his, supporters, prefix, chowkidar, t...
4  [answer, who, among, these, the, most, powerfu...

                                        stopped_text  length
0  [modi, promised, minimum, government, maximum,...      33
1      [talk, nonsense, continue, drama, vote, modi]      13
2  [say, vote, modi, welcome, bjp, told, rahul, m...      22
3  [asking, supporters, prefix, chowkidar, names,...      34
4  [answer, among, powerful, world, leader, today...      14
```

Split data into dependent(X) and independent(y) dataframe

```
x=df['stopped_text']
y=df['category']
```

Do operations on text data

One-hot encoding for each sentence

```
from keras.preprocessing.text import Tokenizer
tokenizer=Tokenizer()
tokenizer.fit_on_texts(df['stopped_text'])
```

```
df['stopped_text']=tokenizer.texts_to_sequences(df['stopped_text'])
df['stopped_text']
```

```
0        [1, 71, 272, 30, 1573, 547, 855, 3079, 1128, 1...
1                                 [213, 957, 661, 1254, 7, 1]
2        [46, 7, 1, 1176, 3, 347, 11, 447, 2986, 1, 37,...
3        [261, 325, 3081, 45, 866, 1, 94, 1726, 3197, 1...
4          [277, 710, 642, 115, 72, 110, 693, 3924, 1, 66]
                                ...
31998      [36, 16899, 1885, 3656, 16899, 131, 7, 3, 7, 1]
31999             [4051, 540, 6, 3592, 1873, 263, 4347, 1]
32000                        [1, 3230, 1710, 235, 87, 337]
32001    [683, 862, 262, 455, 39, 7013, 8681, 9771, 533...
32002             [45, 9, 1, 8065, 135, 621, 116, 255, 93]
Name: stopped_text, Length: 32002, dtype: object
```

Add padding from the front side (use Tensorflow)

Build an LSTM model and compile it(describe features, input length, vocabulary size, information drop-out layer, activation function for output,)

```
vocab_size=df['length'].sum()
vocab_size
```

```
661962
```

```
model = Sequential()
model.add(Embedding(len(tokenizer.index_word)+1, input_length=
100 ,output_dim =50))
model.add(Bidirectional(LSTM(100)))
model.add(Flatten())
model.add(Dense(250, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(1, activation='softmax'))
```

```
from tensorflow import keras
model.compile(optimizer=keras.optimizers.Adam(),

loss=keras.losses.BinaryCrossentropy(),metrics=["accuracy"])
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 100, 50)           1968900

 bidirectional (Bidirectiona  (None, 200)              120800
 l)

 flatten (Flatten)           (None, 200)               0
 dense (Dense)               (None, 250)               50250

 dropout (Dropout)           (None, 250)               0

 dense_1 (Dense)             (None, 1)                 251

=================================================================
Total params: 2,140,201
Trainable params: 2,140,201
Non-trainable params: 0
_________________________________________________________________
```

Do dummy variable creation for the dependent variable

```
df['category'].mask(df['category'] == 'negative',-1,  inplace=True)
df['category'].mask(df['category'] == 'neutral',0,  inplace=True)
df['category'].mask(df['category'] == 'positive',1,  inplace=True)
df['category']
```

```
0        -1
1         0
2         1
3         1
4         1
         ..
31998    -1
31999     1
32000    -1
32001     1
32002     1
Name: category, Length: 32002, dtype: object
```

split the data into tests and train

```
x_train,x_test,y_train,y_test=train_test_split(df['stopped_text'],df['
category'],test_size=0.2,random_state=10)
x_train.shape,y_train.shape
```

```
((25601,), (25601,))
```

```
from keras_preprocessing.sequence import pad_sequences
x_train = pad_sequences( x_train, maxlen=100 ,dtype='float32')
x_test = pad_sequences( x_test, maxlen=100 ,dtype='float32')

x_train = np.asarray(x_train).astype(np.float32)
x_test = np.asarray(x_test).astype(np.float32)


y_train = np.asarray(y_train).astype('float32').reshape((-1,1))
y_test = np.asarray(y_test).astype('float32').reshape((-1,1))

x_train
```

```
array([[0.0000e+00, 0.0000e+00, 0.0000e+00, ..., 5.3840e+03,
3.6990e+03,
        9.0160e+03],
       [0.0000e+00, 0.0000e+00, 0.0000e+00, ..., 1.0000e+00,
4.4910e+03,
        4.9300e+02],
       [0.0000e+00, 0.0000e+00, 0.0000e+00, ..., 1.3640e+03,
4.0980e+03,
        2.0950e+03],
       ...,
       [0.0000e+00, 0.0000e+00, 0.0000e+00, ..., 1.6600e+02,
1.4047e+04,
        1.0000e+00],
       [0.0000e+00, 0.0000e+00, 0.0000e+00, ..., 3.3110e+03,
7.7000e+01,
        1.0490e+03],
       [0.0000e+00, 0.0000e+00, 0.0000e+00, ..., 1.4270e+03,
2.8100e+02,
        3.4330e+03]], dtype=float32)
```

```
x_test
```

```
array([[0.000e+00, 0.000e+00, 0.000e+00, ..., 3.100e+01, 4.900e+01,
        1.000e+00],
       [0.000e+00, 0.000e+00, 0.000e+00, ..., 4.760e+02, 2.080e+02,
        2.480e+02],
       [0.000e+00, 0.000e+00, 0.000e+00, ..., 1.000e+00, 3.000e+01,
        4.630e+02],
       ...,
       [0.000e+00, 0.000e+00, 0.000e+00, ..., 7.500e+02, 1.000e+01,
        1.221e+03],
       [0.000e+00, 0.000e+00, 0.000e+00, ..., 6.700e+01, 1.274e+03,
        8.360e+02],
       [0.000e+00, 0.000e+00, 0.000e+00, ..., 2.840e+02, 1.160e+02,
        3.020e+02]], dtype=float32)
```

Train new model

```
model.fit(x_train,y_train, batch_size=200,
          epochs=10, shuffle=True,
          validation_data=(x_test,y_test), verbose=1)
```

```
Epoch 1/10
129/129 [==============================] - 11s 26ms/step - loss: -
0.0741 - accuracy: 0.4269 - val_loss: -1.1892 - val_accuracy: 0.4126
Epoch 2/10
129/129 [==============================] - 3s 20ms/step - loss: -
85.1543 - accuracy: 0.4269 - val_loss: -206.4855 - val_accuracy:
0.4126
Epoch 3/10
129/129 [==============================] - 3s 19ms/step - loss: -
```

```
463.9545 - accuracy: 0.4269 - val_loss: -555.7766 - val_accuracy:
0.4126
Epoch 4/10
129/129 [==============================] - 3s 19ms/step - loss: -
1050.2677 - accuracy: 0.4269 - val_loss: -867.7546 - val_accuracy:
0.4126
Epoch 5/10
129/129 [==============================] - 3s 20ms/step - loss: -
2197.0349 - accuracy: 0.4269 - val_loss: -2029.1207 - val_accuracy:
0.4126
Epoch 6/10
129/129 [==============================] - 3s 20ms/step - loss: -
4072.7893 - accuracy: 0.4269 - val_loss: -3192.6331 - val_accuracy:
0.4126
Epoch 7/10
129/129 [==============================] - 3s 20ms/step - loss: -
6542.6826 - accuracy: 0.4269 - val_loss: -4792.5557 - val_accuracy:
0.4126
Epoch 8/10
129/129 [==============================] - 3s 23ms/step - loss: -
9645.4492 - accuracy: 0.4269 - val_loss: -6751.6812 - val_accuracy:
0.4126
Epoch 9/10
129/129 [==============================] - 3s 22ms/step - loss: -
12929.0225 - accuracy: 0.4269 - val_loss: -8601.6953 - val_accuracy:
0.4126
Epoch 10/10
129/129 [==============================] - 3s 20ms/step - loss: -
15941.8525 - accuracy: 0.4269 - val_loss: -10446.2666 - val_accuracy:
0.4126
```

```
<keras.callbacks.History at 0x7f5e38751b50>
```

Normalize the prediction as same as the originaldata(prediction might be in decimal, so whoever is nearest to 1 is predicted as yes and set other as 0)

```
results1 = model.evaluate(y_test, y_test, batch_size=200)
```

```
33/33 [==============================] - 1s 3ms/step - loss: 947.3184
- accuracy: 0.4126
```

Measure performance metrics and accuracy

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

```
201/201 [==============================] - 2s 5ms/step
```

```
array([[1.],
       [1.],
       [1.],
       ...,
       [1.],
       [1.],
       [1.]], dtype=float32)
```

print Classification report

```
import sklearn
d=sklearn.metrics.classification_report(y_test, y_pred)
print(d)
```

```
              precision    recall  f1-score   support

        -1.0       0.00      0.00      0.00      1554
         0.0       0.00      0.00      0.00      2206
         1.0       0.41      1.00      0.58      2641

    accuracy                           0.41      6401
   macro avg       0.14      0.33      0.19      6401
weighted avg       0.17      0.41      0.24      6401
```