Emotion Detection from Text

Context

Emotion detection from text is one of the challenging problems in Natural Language Processing. The reason is the unavailability of labeled dataset and the multi-class nature of the problem. Humans have a variety of emotions and it is difficult to collect enough records for each emotion and hence the problem of class imbalance arises. Here we have a labeled data for emotion detection and the objective is to build an efficient model to detect emotion.

** Content**
The data is basically a collection of tweets annotated with the emotions behind them. We have three columns tweet_id, sentiment, and content. In "content" we have the raw tweet. In "sentiment" we have the emotion behind the tweet. Refer to the starter notebook for more insights.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

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

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
###!mkdir ~/.kaggle
```

```
##!cp /kaggle.json ~/.kaggle/
```

```
####!chmod 600 ~/.kaggle/kaggle.json
```

```
####!pip install kaggle
```

```
####!pip install keras-tuner
```

```
####!kaggle datasets download -d pashupatigupta/emotion-detection-from-text
```

```
####! unzip /content/emotion-detection-from-text.zip
```

```
emotions = pd.read_csv("/content/tweet_emotions.csv")
```

```
emotions.head(3)
```

	tweet_id	sentiment	content
0	1956967341	empty	@tiffanylue i know i was listenin to bad habi...
1	1956967666	sadness	Layin n bed with a headache ughhhh...waitin o...
2	1956967696	sadness	Funeral ceremony...gloomy friday...

```
emotions.isnull().sum()
```

```
tweet_id     0
sentiment    0
content      0
dtype: int64
```

```
emotions = emotions[["content","sentiment"]]
```

```
emotions.sentiment.unique()
```

```
array(['empty', 'sadness', 'enthusiasm', 'neutral', 'worry', 'surprise',
       'love', 'fun', 'hate', 'happiness', 'boredom', 'relief', 'anger'],
      dtype=object)
```

```
emotions['encoded_sentiment'] = emotions['sentiment'].astype('category').cat.codes
```

```
emotions['encoded_sentiment'].value_counts()
```

```
8     8638
12    8459
5     5209
10    5165
7     3842
11    2187
4     1776
9     1526
6     1323
2      827
3      759
1      179
0      110
Name: encoded_sentiment, dtype: int64
```

```
emotions['sentiment'].value_counts()
```

```
neutral       8638
worry         8459
happiness     5209
sadness       5165
love          3842
```

```
surprise      2187
fun           1776
relief        1526
hate          1323
empty          827
enthusiasm     759
boredom        179
anger          110
Name: sentiment, dtype: int64
```

```
emotions.head()
```

	content	sentiment	encoded_sentiment
0	@tiffanylue i know i was listenin to bad habi...	empty	2
1	Layin n bed with a headache ughhhh...waitin o...	sadness	10
2	Funeral ceremony...gloomy friday...	sadness	10
3	wants to hang out with friends SOON!	enthusiasm	3
4	@dannycastillo We want to trade with someone w...	neutral	8

```
import string
import re
def clean_text(text):
    '''Make text lowercase, remove text in square brackets,remove links,remove punctuation
    and remove words containing numbers.'''
    text = text.lower()
    text = re.sub('\[.*?\]', '', text)
    text = re.sub('https?://\S+|www\.\S+', '', text)
    text = re.sub('<.*?>+', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub('\n', '', text)
    text = re.sub('\w*\d\w*', '', text)
    return text
```

```
emotions['content'] = emotions['content'].apply(lambda x: clean_text(x))
```

```
emotions['content'].head(2)
```

```
0    tiffanylue i know  i was listenin to bad habit...
1    layin n bed with a headache  ughhhhwaitin on y...
Name: content, dtype: object
```

```
data_texts = emotions["content"] # Features (not-tokenized yet)
data_labels = emotions["encoded_sentiment"] # Lables
```

```
data_texts.shape
```

```
(40000,)
```

```
data_labels.shape
```

```
(40000,)
```

```
data_texts.head(2)
```

```
0    tiffanylue i know  i was listenin to bad habit...
1    layin n bed with a headache  ughhhhwaitin on y...
Name: content, dtype: object
```

```
import tensorflow as tf
```

```
tf.__version__
```

```
'2.7.0'
```

```
from tensorflow.keras.layers import Embedding
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing.text import one_hot
from tensorflow.keras.layers import LSTM
from tensorflow.keras.layers import Dense
```

```
### Vocabulary size
voc_size=5000
```

```
### Onehot Representation
```

```
import string
import re
def clean_text(text):
    '''Make text lowercase, remove text in square brackets,remove links,remove punctuation
    and remove words containing numbers.'''
    text = text.lower()
    text = re.sub('\[.*?\]', '', text)
    text = re.sub('https?://\S+|www\.\S+', '', text)
    text = re.sub('<.*?>+', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub('\n', '', text)
    text = re.sub('\w*\d\w*', '', text)
    return text
```

```
emotions['content'] = emotions['content'].apply(lambda x: clean_text(x))
```

```
emotions['content'].head(2)
```

```
0    tiffanylue i know  i was listenin to bad habit...
1    layin n bed with a headache  ughhhhwaitin on y...
Name: content, dtype: object
```

```
data_texts = emotions["content"].to_list() # Features (not-tokenized yet)
data_labels = emotions["encoded_sentiment"].to_list() # Lables
```

```
from sklearn.model_selection import train_test_split

# Split Train and Validation data
train_texts, val_texts, train_labels, val_labels = train_test_split(data_texts, data_labels, test_size=0.2, rando

# Keep some data for inference (testing)
train_texts, test_texts, train_labels, test_labels = train_test_split(train_texts, train_labels, test_size=0.01,
```

```
train_texts
```

```
 extensis the app said i need  or later ',
 'supernatural tonight  yay',
 'lickmycupcakes specifically like these   also i adore this outfit  yum',
 'counting the hours of lost sunshine until the weekend',
 ' nothing yet',
 ' days of frisbee three nights of partying and sprained ligaments cant imagine a better long weekend',
 'watching snl   yay for jtimberlake hosting  i love him',
 'it looks like it might rain',
 '  well its pouring here rly rly wet',
 'dojie is that u trying to say i have a cold steal heart  lol',
 'leilanili thks  the follow and newest tweets i returned the love',
 'humm i adore mark hoppusday so im just going to throw it out there hoppusday have a nice hoppusday people',
 'orangelight because the one i was having at that moment woke me up',
 'anthothemantho hell yes im too late',
 ' sure i willl ',
 'this is dedicated to all those moms out there  happy mothers day   ',
 ' good but its supposed to storm later',
 'tommcfly tom buonotomato and i were wondering if youd do a tour in asia specifically the philippines any chance you will soon',
 'hoping all my friends who are mothers have a wonderful motherss day im spending mine with my  awesome sons   so happy about that',
 ' omg  im so sorry anything i can do to help',
 'computermuseum  one of my prized magsbookannuals dunno if any others were published prolly should google it',
 'zhundred no not professional at all',
 ' sadly all i have is the stanley steemer  number',
 'madush oh my now im offended  ha',
 'hopes  had a nice mothers day',
 ' todays a drag for me so bored im about to get into the romance book so i prob wont be on til the morn  night twitter babes ',
```

```
 'indieandyy  i hope when youre calling this the shitshow you mean that in a good way because this will be fun',
 'livvixo go for the  that you want to go to most my bro had to switch when he did his  time  its  ',
 'time to play the drums',
 'is walking to tesco with rhiannon and hannah to hide all evidence of lastnight',
 'lyssaloo i was gonna text u and ask what puff meant',
 'sarabareilles just the mere fact that you twittered and someone read it then it matters ps love your song gravity',
 ' i see your date is showing you a good time    still want a stripper picture',
 ' what happened',
 'bradiewebbstack i had had a baked dinner yummy  cant wait for new short stack tv what kind of dips shall it be',
 'nobody likes to feel low priority',
 'shopping tomorrow i think yes',
 'argh my embouchure fail makes me sad',
 'webchickbot the portuguese national library could use it also right now they seem to be in a ca  web example ',
 'work work work finally not sick though',
 'i fell i think my knee is broken  but i look fabulous',
 'gabbylucio now that you say that you do look like demi  hahaha yessss august  will be a blast',
 'kouzrah etherreal was my main preoccupation now its etherreals preoccupation humm me  schizophrenic ',
 'happy morning to everyone',
 ' awesome lemme see when youre done',
 'yorksville yawn is it that time already',
 'i am going to be disgraced with myself for life if i dont make it in next year being a perfectionist sucks good luckkkk',
 ' rperss where are youuuuuuu',
 'claireyjonesy mines curly atm  i want mine to be straight lmao',
 'angryfeet ooh that is good will wait a little bit to see how the moneyjob situation goes but thanks for that',
 'dragonblogger my pleasure  i really enjoy your random word poetry and am disappointed when i miss out on participating',
 'if you carry your childhood with you  you never become olda sutzkever',
 'hi beautiful hows it going  ashleylovegood',
 'impalaguy would luv to hear music too but iï¿½m out of batteries  the tv plays besides but i think this is some kind of vampire movie',
 'the puppy is sick  and some one put they hands on my momma gt',
 ' a great song by east clubbers',
 '  cuss u siad u werent having itlol',
 ...]
```

```
import nltk
import re
from nltk.corpus import stopwords
```

```
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
True
```

```
messages=data_texts.copy()
messages = pd.DataFrame(messages)
messages.reset_index(inplace=True)
messages = messages.rename(columns =  {0:'headlines'})
messages.head(2)
```

	index	headlines
0	0	tiffanylue i know i was listenin to bad habit

```
### Dataset Preprocessing
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
corpus = []
for i in range(0, len(messages)):
    print(i)
    headlines = re.sub('[^a-zA-Z]', ' ', messages['headlines'][i])
    headlines = headlines.lower()
    headlines = headlines.split()

    headlines = [ps.stem(word) for word in headlines if not word in stopwords.words('english')]
    headlines = ' '.join(headlines)
    corpus.append(headlines)
```

```
39942
39943
39944
39945
39946
39947
39948
39949
39950
39951
39952
39953
39954
39955
39956
39957
39958
39959
39960
39961
39962
39963
39964
39965
39966
39967
39968
39969
39970
39971
39972
39973
39974
39975
39976
39977
39978
39979
```

```
39980
39981
39982
39983
39984
39985
39986
39987
39988
39989
39990
39991
39992
39993
39994
39995
39996
39997
39998
39999
```

```
onehot_repr=[one_hot(words,voc_size)for words in corpus]
onehot_repr
```

```
 4171,
 1954,
 3872,
 890,
 816,
 4434,
 530,
 340,
 1072,
 3332,
 1868,
 4505,
 1109],
[2131, 3676, 2794, 2051, 1398],
[4856, 561, 2035, 2461, 982, 1309],
[1541,
 331,
 530,
 2215,
 3857,
 3039,
 3309,
 2871,
 3881,
 743,
 1864,
 331,
 1541,
 3449],
[3005, 4867, 1010, 3059, 1531, 2316, 2569, 4047, 4020, 384, 4867, 4572],
[4386, 1322, 3129, 3676],
[3249, 3458],
```

```
 [1661, 3288, 4326, 1251, 4791],
 [685, 1445, 3576, 1687, 2487, 164, 1848, 4091, 2829, 2746, 4825, 2360],
 [225,
  4867,
  225,
  1552,
  3384,
  2072,
  4432,
  2250,
  3297,
  1349,
  304,
  3264,
  1650,
  815],
 [652, 21, 1188, 4175, 3392, 1686],
 [119, 2250, 3468, 3754, 61, 360, 2280],
 [1406, 743, 3426, 1274, 3118, 3189, 4270, 2717, 2323, 2717, 944],
 [97, 3249, 4602, 1779],
 [318, 4418, 4291, 1694, 90, 3650],
 [3476, 2272, 4622, 711, 652, 488],
 [1807, 1579, 1579, 1579, 1890],
 [2304, 3541, 3302, 2758, 4368, 3900, 1925, 2157, 1531],
 [4250, 4543, 4867, 224, 2072],
 ...]
```

```
sent_length=20
embedded_docs=pad_sequences(onehot_repr,padding='pre',maxlen=sent_length)
print(embedded_docs)
```

```
[[   0    0    0 ... 2789 2930 1691]
 [   0    0    0 ...  353 2027 1046]
 [   0    0    0 ... 2112 1761 1157]
 ...
 [   0    0    0 ... 4994 2280 2605]
 [   0    0    0 ... 3066 1541 2708]
 [   0    0    0 ... 4983 3143 3892]]
```

```
len(embedded_docs)
```

```
40000
```

```
import numpy as np
X_final=np.array(embedded_docs)
```

```
X_final.shape
```

```
(40000, 20)
```

```
Y = pd.get_dummies(emotions['sentiment']).values
print('Shape of label tensor:', Y.shape)
```

```
Shape of label tensor: (40000, 13)
```

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

```
print(X_train.shape,y_train.shape)
```

```
(26800, 20) (26800, 13)
```

```
print(X_test.shape,y_test.shape)
```

```
(13200, 20) (13200, 13)
```

```
### Model Training
```

```
from tensorflow.keras.layers import Dropout
## Creating model
embedding_vector_features=40
model=Sequential()
model.add(Embedding(voc_size,embedding_vector_features,input_length=sent_length))
model.add(Dropout(0.3))
model.add(LSTM(300))
model.add(Dropout(0.3))
model.add(Dense(13,activation='sigmoid'))
opt = tf.keras.optimizers.RMSprop(learning_rate=0.001)
model.compile(loss='categorical_crossentropy',optimizer=opt,metrics=['accuracy'])
print(model.summary())
```

```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_6 (Embedding)     (None, 20, 40)            200000

 dropout_12 (Dropout)        (None, 20, 40)            0

 lstm_6 (LSTM)               (None, 300)               409200

 dropout_13 (Dropout)        (None, 300)               0

 dense_6 (Dense)             (None, 13)                3913

=================================================================
Total params: 613,113
Trainable params: 613,113
Non-trainable params: 0
```

```
_________________________________________________________________
None
```

```
### Finally Training
model.fit(X_train,y_train,validation_data=(X_test,y_test),epochs=30,batch_size=64)
```

```
419/419 [==============================] - 8s 18ms/step - loss: 1.9524 - accuracy: 0.3342 - val_loss: 1.9379 - val_accuracy: 0.3338
Epoch 3/30
419/419 [==============================] - 8s 19ms/step - loss: 1.8895 - accuracy: 0.3590 - val_loss: 1.9483 - val_accuracy: 0.3333
Epoch 4/30
419/419 [==============================] - 8s 19ms/step - loss: 1.8504 - accuracy: 0.3713 - val_loss: 1.9334 - val_accuracy: 0.3462
Epoch 5/30
419/419 [==============================] - 8s 19ms/step - loss: 1.8212 - accuracy: 0.3874 - val_loss: 1.9413 - val_accuracy: 0.3389
Epoch 6/30
419/419 [==============================] - 8s 19ms/step - loss: 1.7877 - accuracy: 0.3996 - val_loss: 1.9390 - val_accuracy: 0.3394
Epoch 7/30
419/419 [==============================] - 8s 19ms/step - loss: 1.7649 - accuracy: 0.4116 - val_loss: 1.9675 - val_accuracy: 0.3417
Epoch 8/30
419/419 [==============================] - 8s 19ms/step - loss: 1.7358 - accuracy: 0.4229 - val_loss: 1.9822 - val_accuracy: 0.3375
Epoch 9/30
419/419 [==============================] - 8s 19ms/step - loss: 1.7110 - accuracy: 0.4287 - val_loss: 2.0233 - val_accuracy: 0.3226
Epoch 10/30
419/419 [==============================] - 8s 19ms/step - loss: 1.6848 - accuracy: 0.4396 - val_loss: 2.0180 - val_accuracy: 0.3292
Epoch 11/30
419/419 [==============================] - 8s 18ms/step - loss: 1.6559 - accuracy: 0.4514 - val_loss: 2.0092 - val_accuracy: 0.3305
Epoch 12/30
419/419 [==============================] - 8s 19ms/step - loss: 1.6232 - accuracy: 0.4611 - val_loss: 2.0576 - val_accuracy: 0.3228
Epoch 13/30
419/419 [==============================] - 8s 19ms/step - loss: 1.5977 - accuracy: 0.4708 - val_loss: 2.0400 - val_accuracy: 0.3273
Epoch 14/30
419/419 [==============================] - 8s 19ms/step - loss: 1.5685 - accuracy: 0.4812 - val_loss: 2.0577 - val_accuracy: 0.3226
Epoch 15/30
419/419 [==============================] - 8s 19ms/step - loss: 1.5409 - accuracy: 0.4922 - val_loss: 2.1005 - val_accuracy: 0.3233
Epoch 16/30
419/419 [==============================] - 8s 19ms/step - loss: 1.5088 - accuracy: 0.5009 - val_loss: 2.1389 - val_accuracy: 0.3133
Epoch 17/30
419/419 [==============================] - 8s 19ms/step - loss: 1.4831 - accuracy: 0.5114 - val_loss: 2.1462 - val_accuracy: 0.3091
Epoch 18/30
419/419 [==============================] - 8s 19ms/step - loss: 1.4550 - accuracy: 0.5198 - val_loss: 2.1433 - val_accuracy: 0.3122
Epoch 19/30
419/419 [==============================] - 8s 19ms/step - loss: 1.4312 - accuracy: 0.5310 - val_loss: 2.1920 - val_accuracy: 0.3029
Epoch 20/30
419/419 [==============================] - 8s 19ms/step - loss: 1.4046 - accuracy: 0.5390 - val_loss: 2.2125 - val_accuracy: 0.3005
Epoch 21/30
419/419 [==============================] - 8s 19ms/step - loss: 1.3822 - accuracy: 0.5418 - val_loss: 2.2436 - val_accuracy: 0.3035
Epoch 22/30
419/419 [==============================] - 8s 19ms/step - loss: 1.3569 - accuracy: 0.5516 - val_loss: 2.2517 - val_accuracy: 0.3033
Epoch 23/30
419/419 [==============================] - 8s 19ms/step - loss: 1.3340 - accuracy: 0.5612 - val_loss: 2.2837 - val_accuracy: 0.3057
Epoch 24/30
419/419 [==============================] - 8s 19ms/step - loss: 1.3109 - accuracy: 0.5683 - val_loss: 2.3300 - val_accuracy: 0.2980
Epoch 25/30
419/419 [==============================] - 8s 19ms/step - loss: 1.2800 - accuracy: 0.5776 - val_loss: 2.3930 - val_accuracy: 0.2980
Epoch 26/30
419/419 [==============================] - 8s 19ms/step - loss: 1.2602 - accuracy: 0.5849 - val_loss: 2.3370 - val_accuracy: 0.2868
```

```
Epoch 27/30
419/419 [==============================] - 8s 19ms/step - loss: 1.2367 - accuracy: 0.5945 - val_loss: 2.4745 - val_accuracy: 0.2848
Epoch 28/30
419/419 [==============================] - 8s 19ms/step - loss: 1.2135 - accuracy: 0.5989 - val_loss: 2.4875 - val_accuracy: 0.2902
Epoch 29/30
419/419 [==============================] - 8s 19ms/step - loss: 1.1903 - accuracy: 0.6070 - val_loss: 2.4613 - val_accuracy: 0.2933
Epoch 30/30
419/419 [==============================] - 8s 19ms/step - loss: 1.1650 - accuracy: 0.6162 - val_loss: 2.5762 - val_accuracy: 0.2859
<keras.callbacks.History at 0x7fa5508a8e90>
```

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

```
y_pred
```

```
array([[1.92611315e-03, 3.71984718e-03, 1.37267709e-01, ...,
        1.06171705e-01, 1.71818510e-01, 2.15803906e-01],
       [7.01709211e-01, 8.66883755e-01, 8.40131938e-01, ...,
        8.51200461e-01, 9.05728281e-01, 9.08081412e-01],
       [7.43765781e-07, 5.64117227e-06, 3.42002568e-05, ...,
        9.14646804e-01, 9.47186071e-03, 5.47179401e-01],
       ...,
       [8.39444056e-07, 8.40893063e-06, 1.06430371e-05, ...,
        2.32592672e-02, 6.11070893e-04, 1.16183143e-02],
       [1.28669702e-02, 2.24537663e-02, 7.70429000e-02, ...,
        6.94419622e-01, 7.57760704e-02, 6.70586526e-01],
       [1.69496948e-03, 6.01941207e-03, 2.01589148e-02, ...,
        6.82627439e-01, 6.93482280e-01, 6.90256119e-01]], dtype=float32)
```

```
accr = model.evaluate(X_test,y_test)
print('Test set\n  Loss: {:0.3f}\n  Accuracy: {:0.3f}'.format(accr[0],accr[1]))
```

```
413/413 [==============================] - 3s 6ms/step - loss: 2.5762 - accuracy: 0.2859
Test set
  Loss: 2.576
  Accuracy: 0.286
```

```
predictions = tf.nn.softmax(y_pred)
```

```
predictions = tf.argmax(predictions, axis=1).numpy()
```

```
predictions
```

```
array([ 8,  4, 10, ...,  7, 10,  5])
```

```
class_names = {
    8  :"neutral",
12 :"worry",
5  :"happiness",
```

```
10 :"sadness",
7  :"love",
11 :"surprise",
4  :"fun",
9  :"relief",
6  :"hate",
2  :"empty",
3  :"enthusiasm",
1  :"boredom",
0  :"anger"
}
```

```
predictions = [class_names[prediction] for prediction in predictions]
```

```
# Inspecting predicted class names
print(predictions[:10])
```

```
['neutral', 'fun', 'sadness', 'neutral', 'worry', 'happiness', 'sadness', 'love', 'worry', 'happiness']
```

```
predictions = pd.DataFrame(predictions)
```

```
predictions = predictions.reset_index()
```

```
test_texts = pd.DataFrame(test_texts)
```

```
test_texts = test_texts.reset_index()
```

```
output = pd.merge(test_texts, predictions, left_on='index', right_on='index')
```

```
output.rename(columns = {'0_x':'test_texts', '0_y':'Emotions'}, inplace = True)
```

```
output
```

	level_0	index	test_texts	Emotions
0	0	0	cvjason inotherwordsc scrapplesandwic thenewn...	neutral
1	1	1	smilinnursannie good morning rock star nurse	fun
2	2	2	says good or should i say bad afternoon	sadness
3	3	3	saranw thank you you should know that i am bot...	neutral
4	4	4	tialebott haha you are just as bad as i am wel...	worry
...	...	...	...	...
317	317	317	i like airports i cant fucking wait until fri...	love
318	318	318	samfenton i didnt even finish cleaning my room...	happiness
319	319	319	honestly cant wait for wednesdays chemistry ex...	neutral

320 rows × 4 columns

```
output.to_csv(r'/content/emotions_detection_from_texts.csv', index=False)
```

✓ 0s completed at 3:31 PM