Emotion Recognition (EDA and Text Pre-Processing)

Steps to run this code

1. Install Anaconda (a free and open-source distribution of the Python and R programming languages for scientific computing, that aims to simplify package management and deployment), follow the steps in this following link https://www.anaconda.com/distribution/, pick the package suitable to your OS
2. Install wordcloud library https://anaconda.org/conda-forge/wordcloud or https://pypi.org/project/wordcloud/
3. Download the dataset from http://saifmohammad.com/WebPages/EmotionIntensity-SharedTask.html

Load Training Data

```
import pandas as pd
import numpy as np
import string

cols = ['id','text','label','intensity']

anger = pd.read_csv('anger_train.txt', header=None, sep="\t", names= cols, index_col=0)
fear = pd.read_csv('fear_train.txt', header=None, sep="\t", names= cols, index_col=0)
sad = pd.read_csv('sadness_train.txt', header=None, sep="\t", names= cols, index_col=0)
joy = pd.read_csv('joy_train.txt', header=None, sep="\t", names= cols, index_col=0)

print (joy.head(20))
```

```
                                                      text label  intensity
id
30000  Just got back from seeing @GaryDelaney in Burs...   joy      0.980
30001  Oh dear an evening of absolute hilarity I don'...   joy      0.958
30002  Been waiting all week for this game ♡♡♡ #ch...   joy      0.940
30003  @gardiner_love : Thank you so much, Gloria! Yo...   joy      0.938
30004  I feel so blessed to work with the family that...   joy      0.938
30005  Today I reached 1000 subscribers on YT!! , #go...   joy      0.926
30006  @Singaholic121 Good morning, love! Happy first...   joy      0.924
30007  #BridgetJonesBaby is the best thing I've seen ...   joy      0.922
30008  Just got back from seeing @GaryDelaney in Burs...   joy      0.920
30009  @IndyMN I thought the holidays could not get a...   joy      0.917
30010                I'm just still . So happy .\nA blast   joy      0.917
30011                    It's meant to be!! #happy #happy   joy      0.917
30012                 💢⚖Yeah‼ PAUL‼⚖💢  #glorious #BB18   joy      0.917
30013  My morning started off amazing!! Hopefully the...   joy      0.917
30014  😱 @cailamarsai you've had me 😂 😂 the whole tim...   joy      0.900
```

```
30015           @iamTinaDatta love you so much #smile 😊😊   joy      0.896
30016  @WyoWiseGuy @LivingVertical however, REI did o...   joy      0.896
30017  2 days until #GoPackGo and 23 days until #GoGi...   joy      0.880
30018  @TheMandyMoore You are beyond wonderful.  Your...   joy      0.879
30019  @luckiiCHARM_ Luckii, I'm changing in so many ...   joy      0.877
```

```
frames = [anger, fear, sad, joy]
data_training = pd.concat(frames)
data_training.reset_index(inplace=True)
print (data_training.head(20))
data_training.label.value_counts()
```

```
       id                                               text  label intensity
0   10000  How the fu*k! Who the heck! moved my fridge!.....  anger     0.938
1   10001  So my Indian Uber driver just called someone t...  anger     0.896
2   10002  @DPD_UK I asked for my parcel to be delivered ...  anger     0.896
3   10003  so ef whichever butt wipe pulled the fire alar...  anger     0.896
4   10004  Don't join @BTCare they put the phone down on ...  anger     0.896
5   10005                               My blood is boiling  anger     0.875
6   10006  When you've still got a whole season of Wentwo...  anger     0.875
7   10007  @bt_uk why does tracking show my equipment del...  anger     0.875
8   10008  @TeamShanny legit why i am so furious with him...  anger     0.875
9   10009  How is it suppose to work if you do that? Wtf ...  anger     0.875
10  10010  im so mad about power rangers. im incensed. im...  anger     0.667
11  10011  Wont use using @mothercareuk @Mothercarehelp a...  anger     0.854
12  10012  Bitches aggravate like what inspires you to be...  anger     0.854
13  10013  Why does @dapperlaughs have to come to Glasgow...  anger     0.938
14  10014                                   Fuking fuming 😤  anger     0.854
15  10015  Zero help from @ups customer service. Just pus...  anger     0.854
16  10016  @ArizonaCoyotes not to mention the GRA guy sto...  anger     0.854
17  10017  I hate my lawn mower. If it had a soul, I'd co...  anger     0.833
18  10018  why are people so offended by kendall he ends ...  anger     0.833
19  10019  I'm about to block everyone everywhere posting...  anger     0.812
fear       1147
anger       857
joy         823
sadness     786
Name: label, dtype: int64
```

```
punc = string.punctuation
data_training['word_count'] = data_training['text'].apply(lambda x : len(x.split()))
data_training['char_count'] = data_training['text'].apply(lambda x : len(x.replace(" ","")))
data_training['punc_count'] = data_training['text'].apply(lambda x : len([a for a in x if a i

data_training[['word_count', 'char_count', 'punc_count']].head(10)
```

	word_count	char_count	punc_count
0	18	79	12
1	23	97	4
2	19	90	4
3	24	111	13
4	24	102	6
5	4	16	0

```
from collections import Counter
# join_text = " ".join(data_training.text)
join_text = " ".join(data_training[data_training['label']=="sadness"]['text'].values)
counter_obj = Counter(join_text.split(" "))
counter_obj.most_common(50)
# print (join_text)
```

```
[('the', 354),
 ('to', 292),
 ('a', 272),
 ('I', 251),
 ('and', 225),
 ('of', 170),
 ('', 161),
 ('is', 154),
 ('in', 153),
 ('for', 108),
 ('my', 100),
 ('you', 99),
 ('that', 89),
 ('it', 88),
 ('on', 88),
 ('be', 83),
 ('have', 79),
 ('with', 76),
 ('not', 72),
 ('me', 67),
 ('so', 63),
 ('but', 59),
 ("I'm", 56),
 ('at', 54),
 ('get', 52),
 ('this', 49),
 ('are', 47),
 ('was', 47),
 ('when', 45),
 ('your', 45),
 ("it's", 43),
 ('all', 42),
 ('or', 42),
 ('&amp;', 41),
 ('will', 41),
 ('just', 40),
 ('i', 40),
 ('they', 39),
 ('can', 39),
 ("don't", 38),
 ('The', 38),
 ('do', 37),
 ('as', 35),
 ('no', 35),
 ('like', 35),
 ('#lost', 34),
 ('about', 33),
 ('-', 32),
 ('he', 31),
 ('by', 29)]
```

```
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
import matplotlib.pyplot as plt

stopwords = set(STOPWORDS)

# Create and generate a word cloud image:
wordcloud = WordCloud(max_font_size=50, background_color="black", stopwords = stopwords, widt

# Display the generated image:
plt.figure( figsize=(20,10) )
plt.imshow(wordcloud)
plt.axis("off")
plt.show()
```

Text Pre-Processing

```
import nltk
```

```
import re
from nltk.tokenize import WordPunctTokenizer
from nltk.corpus import stopwords
tok = WordPunctTokenizer()
pat1 = r'@[A-Za-z_0-9]+'
pat2 = r'https?://[A-Za-z0-9./]+'
pat3 = r'[0-9]+'
combined_pat = r'|'.join((pat1, pat2, pat3))
stop_words = set(stopwords.words('english'))

def tweet_cleaner(data_frame):
    print ("Cleaning and parsing the tweets...\n")
    clean_data = []
    for index, row in data_frame.iterrows():
        stripped = re.sub(combined_pat, '', row.text)
        lower_case = stripped.lower()
        words = tok.tokenize(lower_case)
        filtered_words = [w for w in words if not w in stop_words]
        clean_data.append((" ".join(filtered_words)).strip())

    print ("Done!")
    return clean_data
```

```
clean_data_training_list= tweet_cleaner(data_training)
```

```
Cleaning and parsing the tweets...

Done!
```

```
data_training.text = pd.DataFrame(clean_data_training_list)
data_training.head(10)
```

	id	text	label	intensity	word_count	char_count	punc_count
0	10000	fu * k ! heck ! moved fridge !... knock landlo...	anger	0.938	18	79	12
1	10001	indian uber driver called someone n word . ' m...	anger	0.896	23	97	4
2	10002	asked parcel delivered pick store address # fu...	anger	0.896	19	90	4
3	10003	ef whichever butt wipe pulled fire alarm davis...	anger	0.896	24	111	13
4	10004	' join put phone , talk rude . taking money ac...	anger	0.896	24	102	6

```
data_training.label.value_counts()
```

```
fear       1147
anger       857
joy         823
sadness     786
Name: label, dtype: int64
```

```
data_training.to_csv('emotion_training.csv',encoding='utf-8')
```

Feature Extraction (Bag of Words)

1. "It was the best of times"
2. "It was the worst of times"
3. "It was the age of wisdom"
4. "It was the age of foolishness"

Vocabulary 'It', 'was', 'the', 'best', 'of', 'times', 'worst', 'age', 'wisdom', 'foolishness'

BoW representation

1. "It was the best of times" = [1, 1, 1, 1, 1, 1, 0, 0, 0, 0]
2. "It was the worst of times" = [1, 1, 1, 0, 1, 1, 1, 0, 0, 0]
3. "It was the age of wisdom" = [1, 1, 1, 0, 1, 0, 0, 1, 1, 0]
4. "It was the age of foolishness" = [1, 1, 1, 0, 1, 0, 0, 1, 0, 1]

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(max_features=500)
X_BoW = vectorizer.fit_transform(data_training.text)
print(vectorizer.get_feature_names())
```

```
['absolutely', 'accept', 'act', 'actually', 'afraid', 'alarm', 'almost', 'already', 'als
```

```
print(len(vectorizer.get_feature_names()))
```

```
500
```

```
transformed_BoW = vectorizer.transform(["The weather sure matches the mood in this state toda
print (transformed_BoW)
```

```
  (0, 406)	1
  (0, 432)	1
```

```
print (transformed_BoW.toarray())
```

```
[[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]]
```

Feature Extraction (Tf-IDF)

Tf-IDF stands for **term frequency-inverse document frequency**, and the tf-idf weight is a weight often used in information retrieval and text mining. This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. **The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus**

TF: Term Frequency, which measures how frequently a term occurs in a document. Since every document is different in length, it is possible that a term would appear much more times in long documents than shorter ones. Thus, the term frequency is often divided by the document length (aka. the total number of terms in the document) as a way of normalization:

TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document).

IDF: Inverse Document Frequency, which measures how important a term is. While computing TF, all terms are considered equally important. However it is known that certain terms, such as "is", "of", and "that", may appear a lot of times but have little importance. Thus we need to weigh down the frequent terms while scale up the rare ones, by computing the following:

IDF(t) = log_e(Total number of documents / Number of documents with term t in it). (source:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer_tfidf = TfidfVectorizer(max_features=500)
X_tfidf = vectorizer_tfidf.fit_transform(data_training.text)
print(vectorizer_tfidf.get_feature_names())
```

```
['absolutely', 'accept', 'act', 'actually', 'afraid', 'alarm', 'almost', 'already', 'al
```

```
transformed_tfidf = vectorizer_tfidf.transform(["The weather sure matches the mood in this st
print (transformed_tfidf)
```

```
  (0, 432)      0.6329234562771724
  (0, 406)      0.7742143750242294
```

```
print (transformed_tfidf.toarray())
```

```
[[0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
```