Natural Language Processing
Lab_week6: Understanding Emotions in Tweets

Emotion Detection in Text

We routinely experience emotions such as happiness, anger, sadness etc. As humans, on reading "Why don't you ever text me!", we can either interpret it as a sad or an angry emotion in absence of context; and the same ambiguity exists for machines as well. Lack of facial expressions and voice modulations make detecting emotions in text a challenging problem. However, as we increasingly communicate using text messaging applications and digital agents, contextual emotion detection in text is gaining importance to provide emotionally aware responses to users.
The objective of this Lab is to apply Lexicon and Machine Learning methods to improve the classification results of emotions in Tweets.

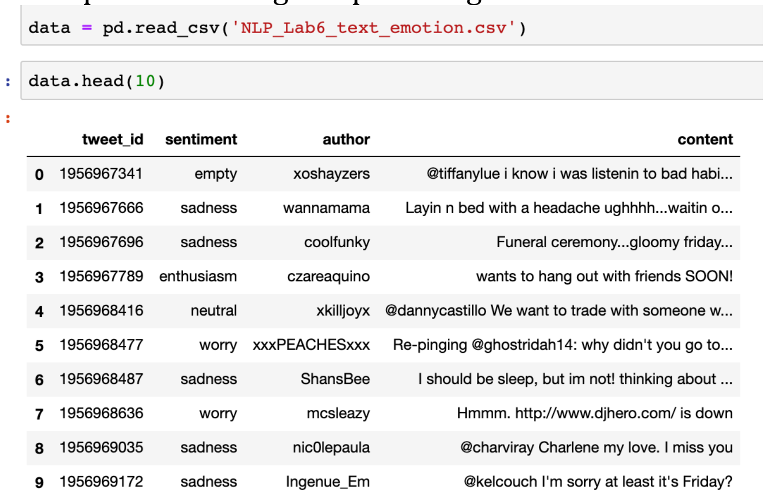
In this task, you are given a dataset (NLP_Lab6_text_emotion.csv), available on Canvas, of 40,000 tweets in total, labelled into 13 different human sentiments, you have to classify the emotion of user as one of the emotion classes.

Data Set Format

The Training dataset is a .csv file containing 4 columns :

1. **ID** - Contains a unique number to identify each training sample
2. **Sentiment** - Contains the human judged label of Emotion
3. **author** - Contains the Twitter user ID (we will not use it in this task)
4. **Content** - Contains the tweet

Examples of training samples are given below:



```
data = pd.read_csv('NLP_Lab6_text_emotion.csv')
```

```
data.head(10)
```

	tweet_id	sentiment	author	content
0	1956967341	empty	xoshayzers	@tiffanylue i know i was listenin to bad habi...
1	1956967666	sadness	wannamama	Layin n bed with a headache ughhhh...waitin o...
2	1956967696	sadness	coolfunky	Funeral ceremony...gloomy friday...
3	1956967789	enthusiasm	czareaquino	wants to hang out with friends SOON!
4	1956968416	neutral	xkilljoyx	@dannycastillo We want to trade with someone w...
5	1956968477	worry	xxxPEACHESxxx	Re-pinging @ghostridah14: why didn't you go to...
6	1956968487	sadness	ShansBee	I should be sleep, but im not! thinking about ...
7	1956968636	worry	mcsleazy	Hmmm. http://www.djhero.com/ is down
8	1956969035	sadness	nic0lepaula	@charviray Charlene my love. I miss you
9	1956969172	sadness	Ingenue_Em	@kelcouch I'm sorry at least it's Friday?

Implementation

Needed libraries

```
In [ ]: import pandas as pd
        import numpy as np
        import nltk
        from nltk.corpus import stopwords
        from textblob import Word
        import re
        from sklearn import preprocessing
        from sklearn.model_selection import train_test_split
        from sklearn.feature_extraction.text import TfidfVectorizer
        from sklearn.feature_extraction.text import CountVectorizer
        from sklearn.metrics import accuracy_score
```

Text pre-processing

First, let's clean and normalise the text by making everything lowercase, removing punctuation, and stop words.

```
# Making all letters lowercase
data['content'] = data['content'].apply(lambda x: " ".join(x.lower() for x in x.split()))
```

```
# Removing Punctuation, Symbols
data['content'] = data['content'].str.replace('[^\w\s]',' ')
```

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

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/haithem.afli/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
True
```

```
# Removing Stop Words using NLTK
stop = stopwords.words('english')
data['content'] = data['content'].apply(lambda x: " ".join(x for x in x.split() if x not in stop))
```

You can use NLTK or any other NLP Python libraries for this.

To gain any proper insight, we need to get all the words to their root form, i.e the variants of a word within the text (for example plural forms, past tense, etc) must all be converted to the base word it represents. This is called lemmatisation.

```
#Lemmatisation
data['content'] = data['content'].apply(lambda x: " ".join([Word(word).lemmatize() for word in x.split()]))
#Correcting Letter Repetitions
```

```
def de_repeat(text):
    pattern = re.compile(r"(.)\1{2,}")
    return pattern.sub(r"\1\1", text)

data['content'] = data['content'].apply(lambda x: " ".join(de_repeat(x) for x in x.split()))
```

We can revert repetition of letters in all words in our collection with the assumption that hardly any word has letters repeated more than twice, consecutively. Though not very accurate, it can help in some corrections.

Next consideration is the idea that if a word is appearing only once in the entire sample of data, then it most likely has no influence in determining the sentiment of the text. Hence we can remove all the rarely occurring words from the dataset which are generally proper nouns and other insignificant words with respect to the current context.

```
# Code to find the top 10,000 rarest words appearing in the data
freq = pd.Series(' '.join(data['content']).split()).value_counts()[-10000:]

# Removing all those rarely appearing words from the data
freq = list(freq.index)
data['content'] = data['content'].apply(lambda x: " ".join(x for x in x.split() if x not in freq))
```

Feature Extraction

Once you make the text data clean, precise, and error-free, each tweet is represented by a group of keywords. Now, we need to perform 'Feature Extraction', i.e extracting some parameters from the data that can be presented numerically. In this article, we consider two different features, TF-IDF & Count Vectors (Remember, we need numeric data!).

```
#Encoding output labels
lbl_enc = preprocessing.LabelEncoder()
y = lbl_enc.fit_transform(data.sentiment.values)
```

```
y
```

```
array([ 2, 10, 10, ...,  7,  5,  7])
```

Count Vectors is another feature we consider and as the name suggests we transform our tweet into an array having the count of appearances of each word in it. The intuition here is that the text that conveys similar emotions may have the same words repeated over and over again. This is more like the direct approach.

```
: print(X_train)
```

```
['dizzyupthegirl thankyoou' 'watching complete first season'
 'blkademic matter' ...
 'cape town beautiful sun shining amazing landscape everything quot easy going quot'
 'know quot extra hand quot comment would cause trouble upload house music beyond vol 3'
 'school absolutely sick want go home writing geography best']
```

```
: print(y_train)
```

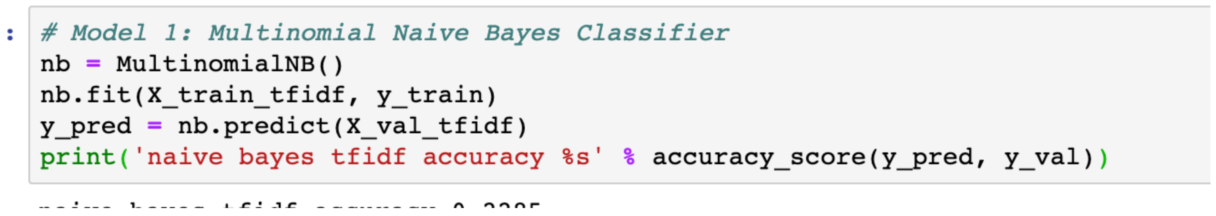
```
[ 8  8 12 ...  4  5 12]
```

```
: # Extracting TF-IDF parameters
tfidf = TfidfVectorizer(max_features=1000, analyzer='word',ngram_range=(1,3))
X_train_tfidf = tfidf.fit_transform(X_train)
X_val_tfidf = tfidf.fit_transform(X_val)

# Extracting Count Vectors Parameters
count_vect = CountVectorizer(analyzer='word')
count_vect.fit(data['content'])
X_train_count =  count_vect.transform(X_train)
X_val_count =  count_vect.transform(X_val)
```

Training The baseline Model

With the numerical representations of the tweets ready, we can directly use them as inputs for some classic machine learning models such as the Multinomial Naïve Bayes.



```
: # Model 1: Multinomial Naive Bayes Classifier
  nb = MultinomialNB()
  nb.fit(X_train_tfidf, y_train)
  y_pred = nb.predict(X_val_tfidf)
  print('naive bayes tfidf accuracy %s' % accuracy_score(y_pred, y_val))

  naive bayes tfidf accuracy 0.2385
```

Now Try to improve this system using more pre-processing techniques, more ML methods and emotion lexicons.

Note: we haven't touched any deep learning techniques yet. Some of the popular ones include RNNs, LSTM, GRU etc.
For more information you can have a look at:

A. Bouchekif, P. Joshi, L. Bouchekif, H. Afli, Epita-adapt at semeval-2019 task 3: Detecting emotions in textual conversations using deep learning models combination, in: Proceedings of the 13th International Workshop on Semantic Evaluation, 2019, pp. 215–219.

https://www.aclweb.org/anthology/S19-2035/