Sentiment Analysis for text data (COVID-19 Tweets)

Prepared By:

Mohammad Saleh

The National ICT Up skilling program – Data Science section

Al-Hussain Technical University

Prepared for:

Dr. Ibrahim AbuAlhaol

Dr. Mohammad AbuAlhaol

Dr. Rawan Al-Kurd

Mr. Ghassan Gammoh

Mr. Omar Mesmar

Jan 28, 2021

Abstract

This project addresses the problem of analyzing feelings on Twitter. The goal of this project was to predict the emotions of this specific Twitter dataset using Python. Sentiment analysis can predict many different emotions associated with the text, but in this report only 3 disciplines were considered: positive, negative and neutral. The training data set was large and the data in it were very good, which greatly affected the ease of building a good classifier. After creating a lot of custom features, using word bag representations and applying some classification algorithms, the classification accuracy was achieved at the level of 89%. The custom features include: Analyzing general sentiments as companies try to know the response of their products to the market, predict political elections and forecast social and economic phenomena such as stock market. This report will describe my project, explain the process used to complete my project, analyze the steps I took in completing my project, and provide two improvements for future projects.

Table of Contents

1. Introduction

This project addresses the problem of analyzing feelings on Twitter. The goal of this project was to predict the emotions of this specific Twitter dataset using Python. Sentiment analysis can predict many different emotions associated with the text, but in this report only 3 disciplines were considered: positive, negative and neutral. The training data set was large and the data in it were very good, which greatly affected the ease of building a good classifier. After creating a lot of custom features, using word bag representations and applying some classification algorithms, the classification accuracy was achieved at the level of 89%. The custom features include: Analyzing general sentiments as companies try to know the response of their products to the market, predict political elections and forecast social and economic phenomena such as stock market. This report will describe my project, explain the process used to complete my project, analyze the steps I took in completing my project, and provide two improvements for future projects.

Statement of the Problem

Text analysis to see if the person's feelings are positive, negative or neutral

Significance of the project

The biggest benefit of this analysis is knowing people's opinions about the virus and analyzing texts to find out their reactions and classify them as positive, negative or neutral

2. Background

In this project, I had the opportunity to work in the sentiment analysis field. While dealing with the reviews of people, wit is important to interpret what the user tends to portray so that we can give him the best results. Apart from this, sentiment analysis has been an interesting field of study. This is still an evolving subject, it has functions that are too complicated to understand by the machines such as sarcasm, negative emotions, hyperbole etc. Because I am part of the industry, I know the potential in sentiment analysis. It adds a lot of value to the industry. Sentiment analysis bases its results on factors that are so inherently humane, it is bound to become one the major drivers of many business decisions in future.

3. Methodology

Sentiment analysis is a wonderful field in data science. There are several steps that must be taken to analyze sentiment in tweets. First, you must have data about the tweets that you should work on, and it is preferable to have big data because the bigger the better the analysis. You display the data, read it and understand its characteristics, and begin to understand the relationships between the columns. Then we start by cleaning the data from punctuation marks, links and special symbols, and many other things that are not necessary for us in the analysis and will consume from RAM only. After we clean the data, we start building models based on clean data. There are several classifiers that can be used such as Logistic Regression, Stochastic Gradient Decent, Naive Beyers and XGboost. Then, you compare the result for each Classifier you used and take the Classifier which has the highest accuracy.

4. Conclusions and recommendations

The increase in microblogging sites like Twitter provides an unparalleled opportunity to shape and use methods and technologies that seek and mine emotions. The work presented in this paper outlines an approach to analyzing sentiment on Twitter data. To reveal feelings, we extracted relevant data from Tweets and added features. The overall sentiment of the tweets was then calculated using a model presented in this report. This work is exploratory in nature and the prototype that has been evaluated is a prototype. The models showed that predicting text sentiment is a non-trivial task of machine learning. A lot of pre-processing is needed just to be able to run an algorithm. The main problem with sentiment analysis is the formulation of an automated representation of a text. The simple word set was definitely not enough to get satisfactory results; thus, a lot of additional features were created based on common sense (number of symbols, exclamation marks, number of question marks, etc.). I believe a slight improvement in the classification accuracy could be developed for the given training data set, but since it included highly skewed data (a small number of negative cases), the difference is likely to be in the order of a few percentages. The thing that could boost the classification results is adding a lot of additional examples (increasing the training data set), moreover - a lot of information expressing feelings is definitely missing.

5. References

[1] Covid 19 tweets Analysis | Kaggle

6. **use the Natural Language Toolkits such as NLTK.**

Import the important libraries

```
import numpy as np
import pandas as pd #
import os
import seaborn as sns
import re
import matplotlib.pyplot as plt
import plotly.express as px
from nltk.stem import LancasterStemmer, SnowballStemmer, RegexpStemmer, WordNetLemmatizer
import nltk
nltk.download('punkt')
from nltk import sent_tokenize
from nltk import word_tokenize
from nltk.tokenize import TweetTokenizer
from nltk.corpus import stopwords
import string
import emoji
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from wordcloud import WordCloud,STOPWORDS
```

- **A basic process on text-based content**

```
df.info()
```

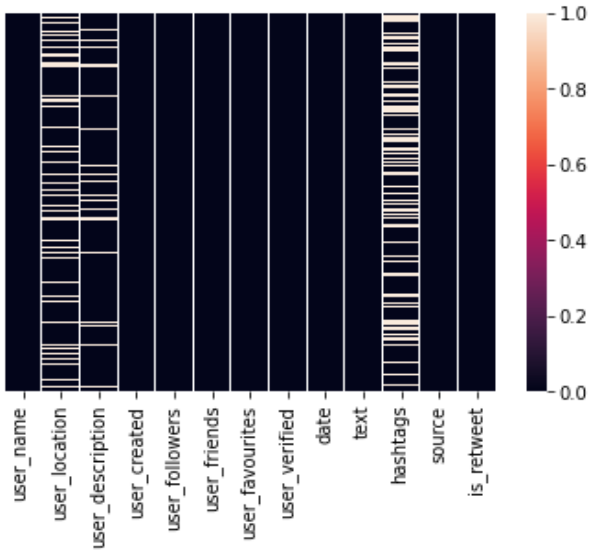
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 179108 entries, 0 to 179107
Data columns (total 13 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   user_name         179108 non-null  object
 1   user_location     142337 non-null  object
 2   user_description  168822 non-null  object
 3   user_created      179108 non-null  object
 4   user_followers    179108 non-null  int64
 5   user_friends      179108 non-null  int64
 6   user_favourites   179108 non-null  int64
 7   user_verified     179108 non-null  bool
 8   date              179108 non-null  object
 9   text              179108 non-null  object
 10  hashtags          127774 non-null  object
 11  source            179031 non-null  object
 12  is_retweet        179108 non-null  bool
dtypes: bool(2), int64(3), object(8)
memory usage: 15.4+ MB
```

- **text data/tweet cleaning**

lit's describe the nullvalues using heatmap to show the missing value for each column

```
sns.heatmap(df.isnull(), cbar=True, yticklabels=False)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdf16d039e8>
```



- **Descibe the percentage of unique values**

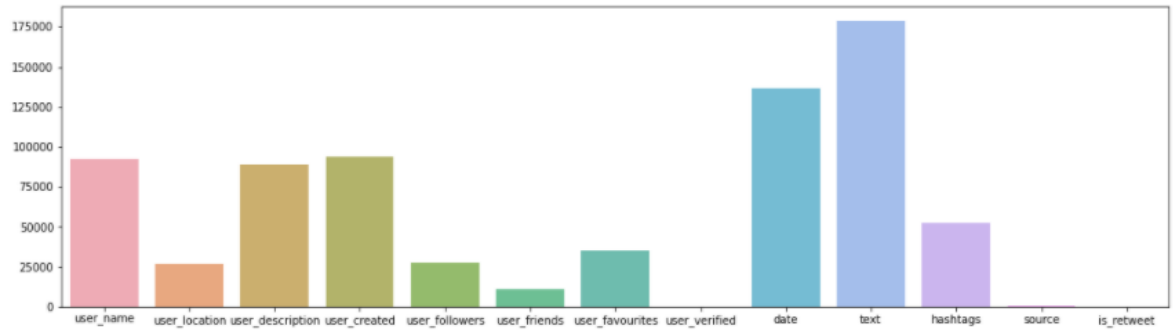
Here we will show the percentage of unique values for each column and represent it by subplot

```
f, ax = plt.subplots(1,1, figsize=(18,5))
sns.barplot(df.columns, y=df.nunique(), alpha=0.8)
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/_decorators.py:43: FutureWarning:

Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other argu

<matplotlib.axes._subplots.AxesSubplot at 0x7fdf0c2f05f8>
```



7. Use Regular expression to clean text

Here we will add two columns, the first column is the text after cleaning, and the second column is to make sure that there are no emojis

```
[ ]  def nlp(df):
         df['token'] = df['text'].apply(lambda x: x.lower())
         df['token'] = df['token'].apply(lambda x: x.replace('\n', ' '))
         df['token'] = df['token'].str.replace('http\S+|www.\S+', '', case=False)
         df['token'] = df['token'].apply(lambda x: x.replace('&gt;', ''))
         df['emoji'] = df['token'].apply(lambda x: text_has_emoji(x))
         df['token'] = df['token'].apply(lambda x: str(x).replace(" s ", " "))
         return df
```

- **Remove emoji and Punctuation**

The emoji library is a library from Python that helps us determine if there are emojis in the text that we want to work on or not, and it helps us and gives us options to delete the symbol and others

```
import re
def char_is_emoji(character):
    return character in emoji.UNICODE_EMOJI
#does the text contain an emoji?
def text_has_emoji(text):
    for character in text:
        if character in emoji.UNICODE_EMOJI:
            return True
    return False
#remove the emoji
def deEmojify(inputString):
    return inputString.encode('ascii', 'ignore').decode('ascii')
```

This function will determine whether or not there are punctuation marks, and if any, we will delete them

```
[ ]  punct =[]
     punct += list(string.punctuation)
     punct += '’'
     punct.remove("'")
     def remove_punctuations(text):
         for punctuation in punct:
             text = text.replace(punctuation, ' ')
         return text
```

- **Tokenization**

and here we will tokenized the text columns

```
tt = TweetTokenizer()
tweets1['token'].apply(tt.tokenize)
```

```
0         [if, i, smelled, the, scent, of, hand, sanitiz...
1         [hey, and, wouldnt, it, have, made, more, sens...
2         [trump, never, once, claimed, covid, was, a, h...
3         [the, one, gift, covid, has, give, me, is, an,...
4         [july, media, bulletin, on, novel, coronavirus...
                                ...
179103    [thanks, for, nominating, me, for, the, wearam...
179104                 [the, year, of, insanity, lol, covid]
179105    [a, powerful, painting, by, juan, lucena, its,...
179106    [more, than, students, test, positive, for, co...
179107                      [i, stop, when, i, see, a, stop]
Name: token, Length: 179108, dtype: object
```

Here we will analyze the text, and find out whether the feelings are negative, positive or Neutral

```
def categoriser(diction):
    if(diction['neg']>0):
        return("Negative")
    elif(diction['pos']>0):
        return('Positive')
    else:
        return('Neutral')
```

Sentiment Analysis is the process of 'computationally' determining whether a piece of writing is positive, negative or neutral. It's also known as opinion mining, deriving the opinion or attitude of a speak. VADER (Valence Aware Dictionary and sentiment Reasoner) is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media. VADER uses a combination of A sentiment lexicon is a list of lexical features (e.g., words) which are generally labeled according to their semantic orientation as either positive or negative.

VADER not only talks about the Positivity and Negativity score but also tells us about how positive or negative a sentiment is.

Lambda functions are useful when working on a collection of elements through higher-order functions, which can take one or more functions as arguments. This function is then invoked on the collection. When combined with Pandas functions such as.map (). apply (), or.applymap (), a Lambda function can be a powerful tool to derive new values.

- **polarity scores function**

here we apply the polarity_scores function and add it in to new columns called token , thenwe add new column call sentiment and apply the function categoriser om it to detirmen the text sentiment if its positive negative or neutral

```
def SentiAnlyser(df):
    analyser= SentimentIntensityAnalyzer()
    df['sentiment']=df['token'].apply(lambda x: analyser.polarity_scores(x))
    df['sentiment']=df['sentiment'].apply(lambda x:categoriser(x))
    return df
```

```
import nltk
nltk.download('vader_lexicon')
```

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
True
```

here we will apply SentiAnalyzer function on the data and put it in new varible called tweets2

```
[ ] tweets2=SentiAnlyser(tweets1)
```

```
tweets2.head()
```

	text	token	emoji	sentiment
0	If I smelled the scent of hand sanitizers toda...	if i smelled the scent of hand sanitizers toda...	False	Positive
1	Hey @Yankees @YankeesPR and @MLB - wouldn't it...	hey and wouldnt it have made more sense to ha...	False	Negative
2	@diane3443 @wdunlap @realDonaldTrump Trump nev...	trump never once claimed covid was a hoax we a...	False	Negative
3	@brookbanktv The one gift #COVID19 has give me...	the one gift covid has give me is an appreciat...	False	Positive
4	25 July : Media Bulletin on Novel #CoronaVirus...	july media bulletin on novel coronavirusupda...	False	Positive

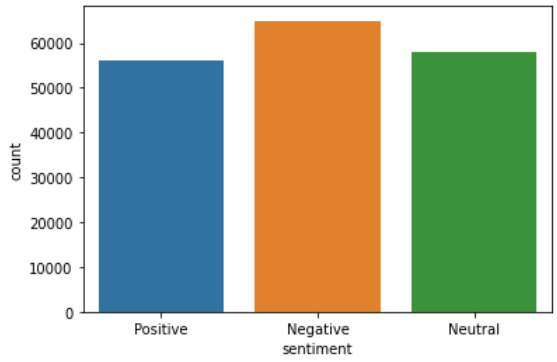
As we can see here, we have a new column called sentiment, to determine if the text is negative, positive, or neuterl, or Neutral

- **plot for the sentiment column**

Here we will draw a plot for the sentiment column, to see if the values are palanced or not , if it's not , we will apply PCA

```
sns.countplot(x='sentiment',data=tweets2)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdefb5c96a0>
```



- **Tokenize and Steeming**

Stemming is the process of producing morphological variants of a root/base word. Stemming programs are commonly referred to as stemming algorithms or stemmers. A stemming algorithm reduces the words “chocolates”, “chocolatey”, “choco” to the root word, “chocolate” and “retrieval”, “retrieved”, “retrieves” reduce to the stem “retrieve”.

```
[29] from nltk.stem import PorterStemmer
     from nltk.tokenize import sent_tokenize, word_tokenize

     ps = PorterStemmer()
```

here is the function that we will be using it to tokenize and steem the text

```
[30] def tokenize(text):
         return word_tokenize(text)

     def stemming(words):
         stem_words = []
         for w in words:
             w = ps.stem(w)
             stem_words.append(w)

         return stem_words
```

add the tokenization and stemming columns

after we apply the tokenization and stemming , we will add another column called new_tokenized that we apply the function above on it

```
[31] tweets2['new_tokenized'] = tweets2['token'].apply(tokenize)
```

```
[65] tweets2['steeming'] = tweets2['new_tokenized'].apply(stemming)
```

thats how the data set looking after we add the tokenization and steeming columns

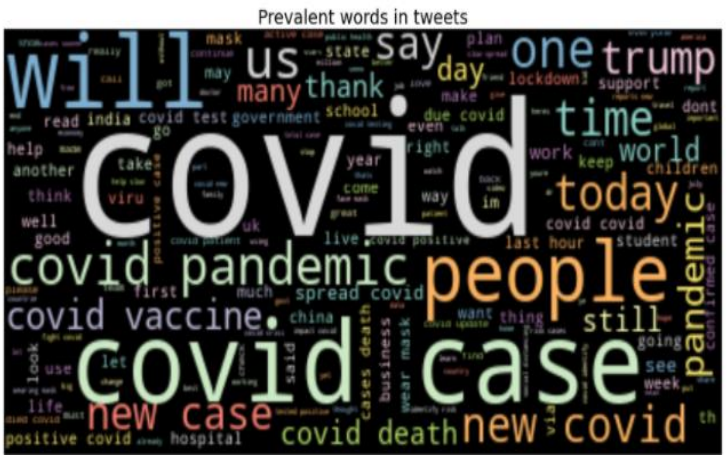
```
tweets2.head()
```

	text	token	emoji	sentiment	new_tokenized	steeming
0	If I smelled the scent of hand sanitizers toda...	if i smelled the scent of hand sanitizers toda...	False	Positive	[if, i, smelled, the, scent, of, hand, sanitiz...	[if, i, smell, the, scent, of, hand, sanit, to...
1	Hey @Yankees @YankeesPR and @MLB - wouldn't it...	hey and wouldnt it have made more sense to ha...	False	Negative	[hey, and, wouldnt, it, have, made, more, sens...	[hey, and, wouldnt, it, have, made, more, sens...
2	@diane3443 @wdunlap @realDonaldTrump Trump nev...	trump never once claimed covid was a hoax we a...	False	Negative	[trump, never, once, claimed, covid, was, a, h...	[trump, never, onc, claim, covid, wa, a, hoax,...
3	@brookbanktv The one gift #COVID19 has give me...	the one gift covid has give me is an appreciat...	False	Positive	[the, one, gift, covid, has, give, me, is, an,...	[the, one, gift, covid, ha, give, me, is, an, ...
4	25 July : Media Bulletin on Novel #CoronaVirus...	july media bulletin on novel coronavirusupda...	False	Positive	[july, media, bulletin, on, novel, coronavirus...	[juli, media, bulletin, on, novel, coronavirus...

8. Wordcloud

here we will make a wordcloud for the text , and we will display it without the words 'covid19' , 'https' and 'coronavirus' , after that we will display this wordcloud

```
text = ",".join(review for review in tweets2.token if 'covid19' not in review and 'amp' not in review and 'coronavirus' not in review and 'need' not in review and 'now' not in revie
wordcloud = WordCloud(max_words=200, colormap='Set3',background_color="black").generate(text)
plt.figure(figsize=(15,10))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.figure(1,figsize=(12, 12))
plt.title('Prevalent words in tweets',fontsize=19)
plt.show()
```



9. Classification

Experiment 1: Naive Bayes classifier

The multinomial Naive Bayes classifier is suitable for classification with discrete features (e.g., word counts for text classification). The multinomial distribution normally requires integer feature counts. However, in practice, fractional counts such as tf-idf may also work.

After we apply multinomial Naive Bayes classifier, we will see the confusion matrix and the accuracy score for this classifier.

```
training accuracy Score    :  0.8634409502672976
Validation accuracy Score :  0.771788286527832
              precision    recall  f1-score   support

    Negative       0.86      0.71      0.78     14879
     Neutral       0.71      0.87      0.78     10244
    Positive       0.75      0.77      0.76     10699

    accuracy                           0.77     35822
   macro avg       0.77      0.78      0.77     35822
weighted avg       0.78      0.77      0.77     35822
```

Experiment 2: stochastic gradient descent (SGD) classifier

This estimator implements regularized linear models with stochastic gradient descent (SGD) learning: the gradient of the loss is estimated each sample at a time and the model is updated along the way with a decreasing strength schedule (aka learning rate). SGD allows minibatch (online/out-of-core) learning, see the partial fit method.

After we apply stochastic gradient descent (SGD) classifier, we will see the confusion matrix and the accuracy score for this classifier.

```
Training accuracy Score    :  0.8839035216280725
Validation accuracy Score :  0.8684328066551281
              precision    recall  f1-score   support

    Negative       0.78      0.93      0.85     10247
     Neutral       0.95      0.83      0.89     14385
    Positive       0.87      0.85      0.86     11190

    accuracy                           0.87     35822
   macro avg       0.87      0.87      0.87     35822
weighted avg       0.88      0.87      0.87     35822
```

Experiment 3: XGboost classifier

XGBoost stands for “Extreme Gradient Boosting”. XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements Machine Learning algorithms under the Gradient Boosting framework. It provides a parallel tree boosting to solve many data science problems in a fast and accurate way.

After we apply XGBoost classifier, we will see the confusion matrix and the accuracy score for this classifier.

```
Training accuracy Score    :  0.6272838937509596
Validation accuracy Score :  0.623834515102451
              precision    recall  f1-score   support

    Negative       0.40      0.85      0.54      5691
     Neutral       0.98      0.53      0.68     23475
    Positive       0.47      0.77      0.58      6656

    accuracy                           0.62     35822
   macro avg       0.62      0.72      0.60     35822
weighted avg       0.79      0.62      0.64     35822
```

Experiment 4: Logistic Regression classifier

Logistic regression is a statistical model that in its basic form uses a **logistic** function to model a binary dependent variable, although many more complex extensions exist.
In **regression** analysis, **logistic regression** (logit **regression**) is estimating the parameters of a **logistic** model (a form of binary **regression**).
After we apply Logistic Regression classifier, we will see the confusion matrix and the accuracy score for this classifier.

```
Training accuracy Score    :  0.942052957023017
Validation accuracy Score :  0.8901792194740662
              precision    recall  f1-score   support

    Negative       0.84      0.91      0.88     11316
     Neutral       0.95      0.88      0.91     13503
    Positive       0.88      0.88      0.88     11003

    accuracy                           0.89     35822
   macro avg       0.89      0.89      0.89     35822
weighted avg       0.89      0.89      0.89     35822
```

10. Compare between the classifier

```
models = pd.DataFrame({
    'Model': [ 'Logistic Regression',
               'Naive Bayes',
              'Stochastic Gradient Decent', 'XGBoost'],
    'Test accuracy': [ logreg_accuracy,
               accuracy,
              sgd_accuracy, xgb_accuracy,]})

models.sort_values(by='Test accuracy', ascending=False)
```

	Model	Test accuracy
0	Logistic Regression	0.903411
2	Stochastic Gradient Decent	0.890095
1	Naive Bayes	0.753615
3	XGBoost	0.625482

```
accuracy = models ['Test accuracy'].tolist()
model  = models ['Model'].tolist()

plt.plot( model,accuracy, label = 'Comparison between the classifier',
      color='r', marker='o', markerfacecolor='k',
      linestyle='--', linewidth=3)
```

```
[<matplotlib.lines.Line2D at 0x7f46c26467b8>]
```