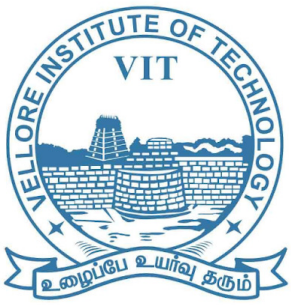


VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Automatic sentiment analysis of Twitter messages

Social and Information Networks

Under Guidance of
Professor Manjula R

Team Members,

19BCE2222 - Hemaksh Chaturvedi
19BCE0638 - Mohamamd Ayaazuddin
19BCE0777- Rishabh Raj

TABLE OF CONTENT

1. Introduction

The medical field in today's day and age has come a long way with providing cures to innumerable numbers of diseases. But still, all these cures have been for physical types of illnesses and not the ones present in our mind. Mental illnesses are still not widely accepted as an actual illness and are thought to be just another part of life that a human goes through. Depression is one of the biggest ones that has affected the world. Although there are known and effective treatments available nowadays for depression, a significant portion of the population never receives such treatment. A few reasons for this are - lack of resources, lack of trained health-care providers and social stigma associated with mental health.

2. Problem statement

We've all faced different levels of depression or some form of sadness in our life. In this project, we aim to find out those in need by doing a sentimental analysis of twitter messages. Our target is to classify and distinguish people into different risk zones of depression via sentimental analysis.

We will attempt to conduct sentiment analysis on "tweets" using various different text preprocessing and Naïve Bayes Classifier algorithms. We attempt to classify the polarity of the tweet where it is either positive or negative. If the tweet has both positive and negative elements, the more dominant sentiment should be picked as the final label.

We use the dataset from Kaggle which was crawled and labeled positive/negative. The data provided comes with emoticons, usernames and hashtags which are required to be processed and converted into a standard form. We also need to extract useful features from the text such as unigrams, bigrams and trigrams which is a form of representation of the "tweet". We use Naïve-Bayes Classifier to conduct sentiment analysis using the extracted features. However, just relying on an individual model did not give a high accuracy.

3. Objective

With the growing technology, there is a huge volume of data present around us. The Internet these days has become a platform for exchanging learnings, ideas and thoughts . Social media websites like twitter , facebook , instagram are becoming popular as they allow people to share and express views about the things which they want to. This survey focuses mainly on sentiment analysis of twitter data whether the messages (tweets) sound positive ,negative or neutral . We will analyse this by using some of the existing Machine Learning algorithms like Naive Bayes , Support Vector Machines by performing data analysis on the twitter dataset .The main objective is to provide a model with accuracy so that these challenges can solved to the highest possible level and help as many as people as we can.

4. Scope

The scope of this project is to highlight and flag the tweets of users facing depression or sadness of some form using sentimental analysis and to categorise them into different levels of risk. We hope to find out sooner than later of those at high risk in their depression so that they can get the help and support they require.

Many people are not very vocal about their depression and often stay in denial, so through the help of certain keywords we may be able to flag them and make them confront their emotions and get them the help they need.

5. Literature Survey

Paper 1:

Title: Automatic Sentiment Analysis of Twitter Messages	Author:,Leandro De Castro, Lima, A. C	Journal, year: ResearchGate 2017

<table>
<tr><td colspan="2">Problems and Objectives:
The objective of this paper is to perform experiments which will consider 3 techniques for sentiment analysis.
1) Emoticon based approach
2) Word Based approach
3) Hybrid approach</td></tr>
<tr><td>Methodology:
The model comprises 3 modules.
1) Support Counting Module:
Responsible for checking the percentage of tweets that contain at least one emoticon.
2) Database Selection Module:
The dataset is divided into two sets: training and testing.
3) Classification Module:
This module is responsible for classifying the tweets whose labels are unknown.</td><td>Limitations:

Limitation of this approach is that it was only able to observe/measure the messages in terms of positive and negative only.</td></tr>
<tr><td>Performance:
For performance, they used the following measures:
1. Precision
2. Recall
3. F-measure</td><td>Future Work:
Possible future works for this technique is that the one next step of this research is to add the label Neutral to the classification.</td></tr>
</table>

Paper 2:

<table>
<tr><td>Title: Sentiment Analysis of Tweets using Machine Learning Approach</td><td>Author: Megha Rathi, Aditya Malik, Daksh Varshney, Rachita Sharma, Sarthak Mendiratta</td><td>Journal, year: 2018</td></tr>
<tr><td colspan="3">Problems and Objectives: \
The main emphasis of this research is on the classification of emotions of tweets' data gathered from Twitter. Researchers have used existing machine learning techniques for sentiment analysis but the results showed that existing machine learning techniques were not providing better results of sentiment classification. In order to improve classification results in the domain of sentiment analysis,</td></tr>
</table>

Methodology: Here they have used a machine learning techniques for increasing the efficiency and reliability of proposed approach They have merged Support Vector Machine with Decision Tree in order to get better classification results in terms of f-measure and accuracy in contrast to individual classifiers.	Limitations: The limitation of this paper is that certain subheaders weren't considered for evaluation
Performance: For performance, they used the following measures: 1. Accuracy 2. F-measure	Future Work: A possible future work for this paper is that the limitation of evaluating each subheading while reducing time of evaluation

Paper 3:

Title: Techniques for Sentiment Analysis of Twitter Data: A Comprehensive Survey	Author:,Mitali Desai	Journal, year: ResearchGate 2019
Problems and Objectives: In this paper, we present a sentiment analysis process for Twitter data. Twitter is a micro-blogging site that is rapidly growing in terms of number of users. Moreover, Tweets are mostly public and limited to 140 characters that simplify the identification of emotions in text.		

Methodology:	Limitations:
1) Firstly, the collected Twitter data is pre-processed to perform the data cleaning. 2) Secondly, the important features are extracted from the clean text, applying any of the feature selection methods. 3) Thirdly, the portion of the data is manually labeled as positive or negative Tweets to prepare a training set. 4) Finally, the extracted features and the labeled training set are provided as an input to the built classifier to classify the remaining data	The abundance of data, use of short forms, timing of different posts, and diversity of language make the sentiment analysis process difficult for Twitter data.
Performance: In common, the performance of sentiment classification techniques is estimated using four indicators as follows: Accuracy, Precision, Recall and F1-score	Future Work: The future opportunities in the domain of sentiment analysis include developing a technique to perform sentiment classification that can be applicable to any data regardless of domain. In addition, language diversity in social media data is a key issue which is required to be eliminated in future.

Paper 4:

Title:	Author:,	Journal, year:
Sentiment Analysis on Twitter Data	Varsha Sahayak, Vijaya Shete. Apashabi Pathan	International Journal of Innovative Research in Advanced Engineering (IJIRAE), 2015

Problems and Objectives:
In this paper, the authors have discussed about a paradigm to extract the sentiment from a famous micro blogging service, Twitter, where users post their opinions for everything.An approach is introduced that automatically classifies the sentiments of Tweets taken from Twitter dataset

Methodology: Messages or tweets are classified as positive, negative or neutral with respect to a query term.This is done by using: 1. Parts Of Speech (POS)-specific prior polarity features. 2. A tree kernel to prevent the need for monotonous feature engineering. 3. Naive Bayes 4. Maximum Entropy 5. Support Vector Machines	Limitations: The classification of tweets is only on the basis of positive/negative or neutral using a scoring system of 1 through 3
Performance: Based on the dictionary assignment of score, the proposed system interprets whether the tweet is positive, negative or neutral	Future Work: Since emoticons and emojis have become a major part of expression they should also be evaluated with accuracy

Paper 5:

Title: Machine Learning Algorithms for Opinion Mining and Sentiment Classification	Author:, Jayashri Khairnar and Mayura Kinikar.	Journal, year: IEEE, June 2020
Problems and Objectives: Sentimental analysis over feedback given on tweets.		

Methodology:	Limitations:
1. Removing the Stopwords using nltk.corpus (Extra Words) 2. Classifying the words into clusters using Support Vector Classifier 3. Enabling and Categorizing words based on N-Grams Technique considering n-words of a sentence. 4. Creating a vector of multiple words and not a single word in the sentence. 5. Reviews can be added by only those users who have bought the product before and cannot be reviewed by a user who hasn't bought the product. 6. Every time a new review is added, the model gives a score to that review, and that is added to the existing score and the average of the score is calculated and this gives us the rating of the product which is then displayed on the product page.	Identifying positive and negative polarity with higher accuracy of other Machine Learning Algorithms
Performance: In this case, it is found that the Support Vector Classifier has greater accuracy compared to other classifiers.	Future Work: This could also be implemented over the amazon website for sentimental analysis of the product

6. Implementation

Process :

Section A: Preparing The Test Set

- Step A.1: Getting the authentication credentials
- Step A.2: Authenticating our Python script
- Step A.3: Creating the function to build the Test set

Section B: Preparing The Training Set

Section C: Pre-processing Tweets in The Data Sets

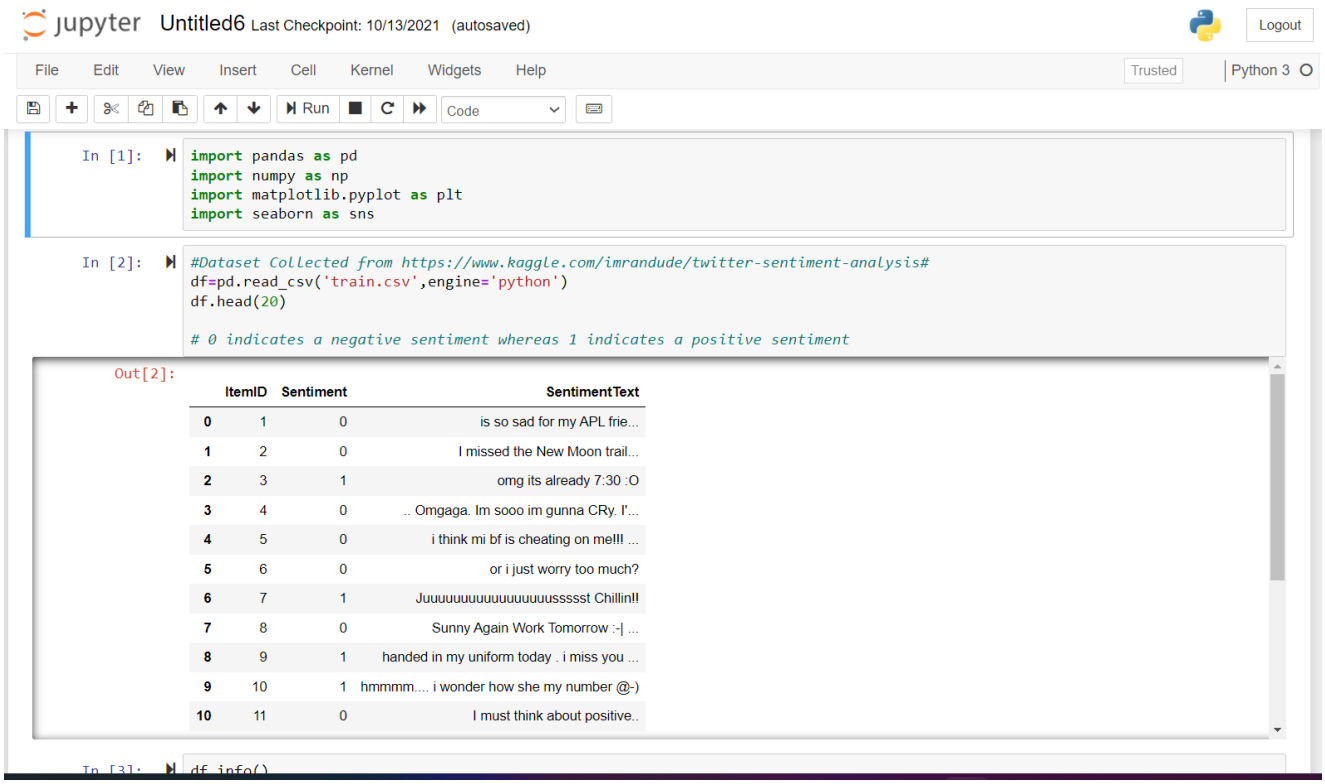
Section D: Naive Bayes Classifier

- Step D.1: Building the vocabulary
- Step D.2: Matching tweets against our vocabulary
- Step D.3: Building our feature vector
- Step D.4: Training the classifier

Section E: Testing The Model

6.1. Dataset

The information given is as comma-isolated qualities recorded with tweets and their related feelings. The preparation dataset is a csv record of type tweet_id, sentiment_tweet where the tweet_id is a remarkable whole number distinguishing the tweet, opinion is either 1 (positive) or 0 (negative), and tweet is the tweet encased in "". We will utilize this single dataset to cross approve our model.



6.2. Pre-processing

Basic feature extraction

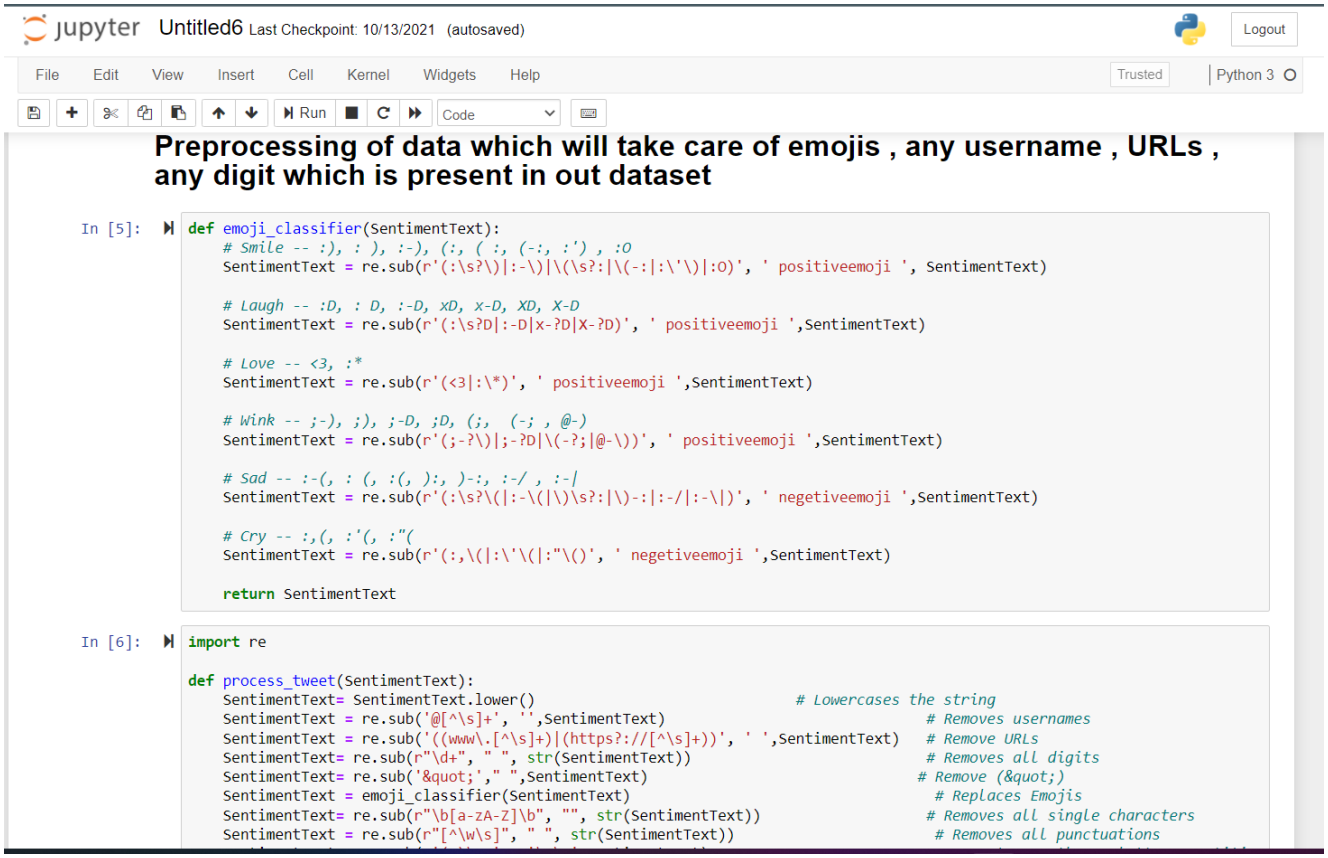
Pre-handling Raw tweets scratched from twitter for the most part result in a boisterous dataset. This is because of the easygoing idea of individuals' use of online media. Tweets have specific uncommon attributes, for example, retweets, hashtags, emojis, client makes reference to, and so forth which must be appropriately removed. Hence, crude twitter information must be standardized to make a dataset which can be handily educated by different classifiers. We have applied a broad number of pre-handling steps to normalize the dataset and decrease its size.
We initially do some broad pre-handling on tweets which is as per the following.
Convert the tweet to bring down the case.
Supplant at least 2 specks (.) with space.
Strip spaces and statements (" and ') from the closures of the tweet.

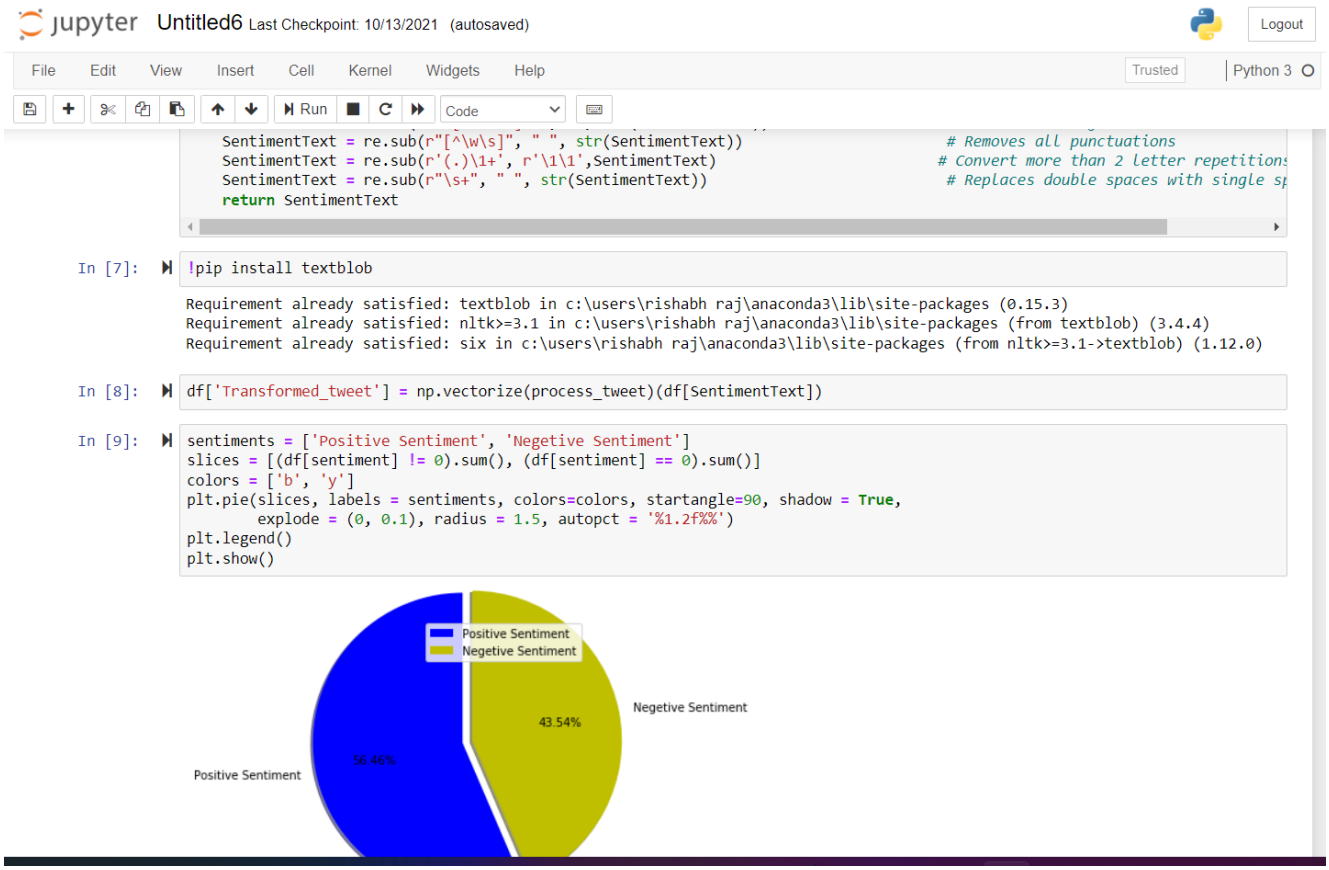
We handle special twitter features as follows.

Usernames and mentions:
Each twitter user has a handle related to them. Users frequently notice different clients in their tweets by @handle. We erase all users as they are one of a kind and contribute nothing to feeling.

URL:
Twitter Users frequently share hyperlinks to different pages in their tweets. A specific URL isn't significant for text order as it would prompt exceptionally inadequate highlights. Consequently, we erase every one of the URLs in tweets. Along these lines estimation turns out to be less asset hungry.

Emoticon:
Users regularly utilize various emojis in their tweets to pass on various feelings. It is difficult to comprehensively match every one of the various emojis utilized via web-based media as the number is truly expanding. In any case, we match some normal emojis which are utilized as often as possible. We supplant the coordinated emojis with either 'positive emoticon' or 'negative emoticon' contingent upon whether it is passing on a positive or a negative feeling. A rundown of all emojis matched by our technique is given underneath.


jupyter Untitled6 Last Checkpoint: 10/13/2021 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
Preprocessing of data which will take care of emojis , any username , URLs , any digit which is present in out dataset
In [5]:
def emoji_classifier(SentimentText):
# Smile -- :), :), :-), (:, (:, (-:, :') , :O
SentimentText = re.sub(r'(:\s?\)|:-\)|\(\s?:|\(-:|:\'\)|:O)', ' positiveemoji ', SentimentText)
# Laugh -- :D, : D, :-D, xD, x-D, XD, X-D
SentimentText = re.sub(r'(:\s?D|:-D|x-?D|X-?D)', ' positiveemoji ',SentimentText)
# Love -- <3, :*
SentimentText = re.sub(r'(<3|:*)', ' positiveemoji ',SentimentText)
# Wink -- ;-), ;), ;-D, ;D, (;, (-; , @-)
SentimentText = re.sub(r'(;-?\)|;-?D|\(-?;|@-\))', ' positiveemoji ',SentimentText)
# Sad -- :-(, : (, :(,):,)-:, :-/ , :-|
SentimentText = re.sub(r'(:\s?\(|:-\(|\)\s?:|\)-:|:-/|:-\|)', ' negetiveemoji ',SentimentText)
# Cry -- :,(, :'(, :"(
SentimentText = re.sub(r'(:,\(|:\'\(|:"\()', ' negetiveemoji ',SentimentText)
return SentimentText
In [6]:
import re
def process_tweet(SentimentText):
SentimentText= SentimentText.lower() # Lowercases the string
SentimentText = re.sub('@[^\s]+', '',SentimentText) # Removes usernames
SentimentText = re.sub('((www\.[^\s]+)|(https?://[^\s]+))', ' ',SentimentText) # Remove URLs
SentimentText= re.sub(r"\d+", " ", str(SentimentText)) # Removes all digits
SentimentText= re.sub('"'," ",SentimentText) # Remove (")
SentimentText = emoji_classifier(SentimentText) # Replaces Emojis
SentimentText= re.sub(r"\b[a-zA-Z]\b", "", str(SentimentText)) # Removes all single characters
SentimentText = re.sub(r"[^\w\s]", " ", str(SentimentText)) # Removes all punctuations



6.3. Word Cloud of the dataset (positive and negative):

Word clouds (also known as text clouds or tag clouds) work in a simple way: the more a specific word appears in a source of textual data (such as a speech, blog post, or database), the bigger and bolder it appears in the word cloud.

```
In [11]: from wordcloud import WordCloud

positive_words =' '.join([text for text in df['Transformed_tweet'][df[sentiment] == 1]])
wordcloud = WordCloud(width=700, height=400, random_state=21,
            max_font_size=110,background_color="rgba(255, 255, 255, 0)"
            , mode="RGBA").generate(positive_words)
plt.figure(dpi=600)
plt.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.title("Positive Words generated from the tweets")
plt.show()
```

```
<Figure size 3600x2400 with 0 Axes>
```


```
File  Edit  View  Insert  Cell  Kernel  Widgets  Help                Trusted
```

```
In [12]: negetive_words =' '.join([text for text in df['Transformed_tweet'][df[sentiment] == 0]])
wordcloud = WordCloud(width=800, height=500, random_state=21,
            max_font_size=110,background_color="rgba(255, 255, 255, 0)"
            , mode="RGBA").generate(negetive_words)
plt.figure(dpi=600)
plt.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.title("Negetive Words Generated from the tweet")
plt.show()
```

```
<Figure size 3600x2400 with 0 Axes>
```

6.4. Advanced text processing

Normalization:
Stemming and Lemmatization are Text Normalization (or at times called Word Normalization) procedures in the field of Natural Language Processing that are utilized to plan text, words, and reports for additional handling.

Stemming:
Stemming alludes to the evacuation of does the trick, such as "ing", "ly", "s", and so on by a basic standard based methodology. Stemming calculations work by removing the end or the start of the word, considering a rundown of normal prefixes and additions that can be found in a curved word. This unpredictable cutting can be effective on certain events, yet not generally, and that is the reason we confirm that this methodology presents a few limits. For this reason, we will utilize PorterStemmer from the NLTK library.

Lemmatization:
Lemmatization is a more compelling choice than stemming in light of the fact that it changes over the word into its root word, rather than simply stripping the additions. It utilizes the jargon and does a morphological investigation to get the root word. Lemmatization thinks about the morphological examination of the words. To do as such, it is important to have itemized word references which the calculation can glance through to interface the structure back to its lemma. Subsequently, we normally incline toward utilizing lemmatization over stemming.

6.5. Feature Extraction

We remove two sorts of elements from our dataset, in particular unigrams and bigrams. We make a recurrence conveyance of the unigrams and bigrams present in the dataset and pick top N unigrams and bigrams for our examination.

Unigrams:

Likely the least complex and the most regularly utilized highlights for text order is the presence of single words or tokens in the text. We extricate single words from the preparation dataset and make a recurrence appropriation of these words.

Bigrams:

Bigrams are word sets in the dataset which happen in progression in the corpus. These highlights are a decent method for displaying nullification in regular language like in the expression – This isn't great. A sum of 1954953 one of a kind bigrams were separated from the dataset. Out of these, the greater part of the bigrams at end of recurrence range are commotion and happen not many occasions to impact grouping. We in this manner utilize just top 10000 bigrams from these to make our jargon.

N-Grams:

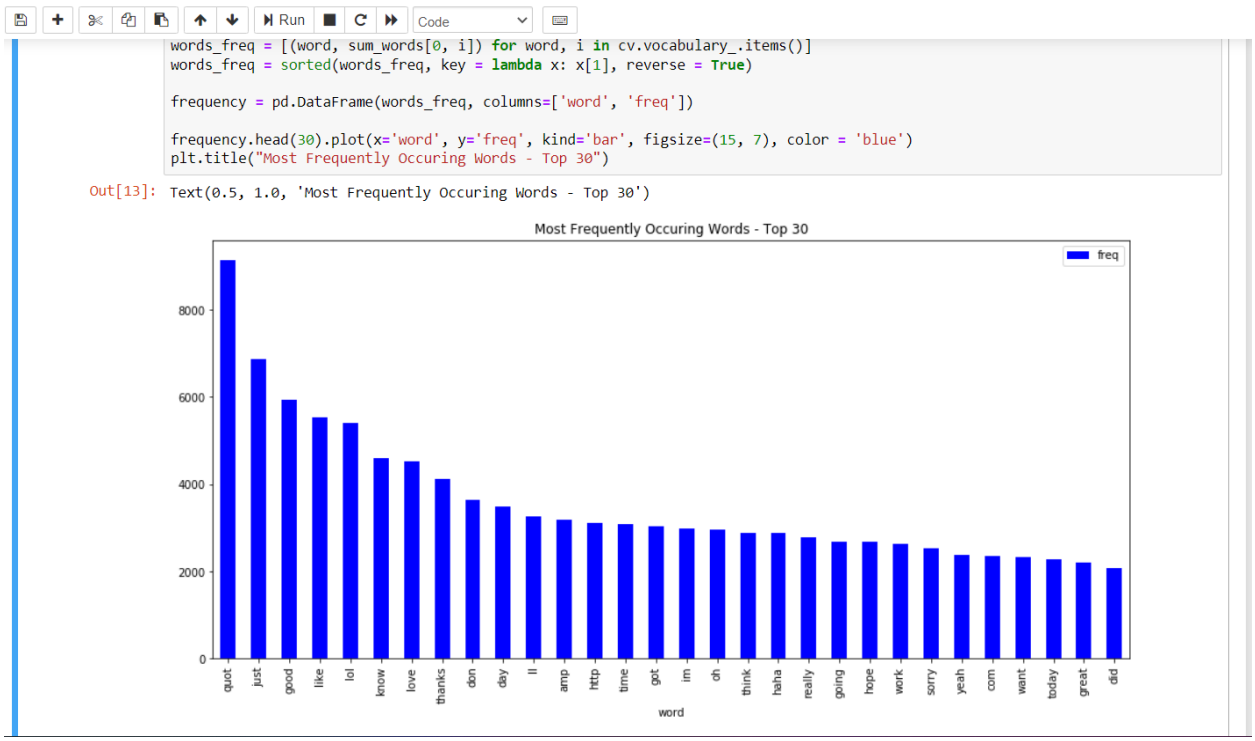
We separate three kinds of highlights from our dataset, to be specific unigrams bigrams and trigrams. N-grams are the blend of different words utilized together. Ngrams with N=1 are called unigrams. Additionally, bigrams (N=2), trigrams (N=3, etc can likewise be utilized.

Feature Extraction

```
In [14]: from sklearn.feature_extraction.text import CountVectorizer

In [15]: count_vectorizer = CountVectorizer(ngram_range=(1,2))
         final_vectorized_data = count_vectorizer.fit_transform(df['Transformed_tweet'])
         final_vectorized_data

Out[15]: <99989x456288 sparse matrix of type '<class 'numpy.int64'>'
                with 2111070 stored elements in Compressed Sparse Row format>
```

Most Used Words

```
In [13]:
from sklearn.feature_extraction.text import CountVectorizer


cv = CountVectorizer(stop_words = 'english')
words = cv.fit_transform(df.SentimentText)

sum_words = words.sum(axis=0)

words_freq = [(word, sum_words[0, i]) for word, i in cv.vocabulary_.items()]
words_freq = sorted(words_freq, key = lambda x: x[1], reverse = True)

frequency = pd.DataFrame(words_freq, columns=['word', 'freq'])

frequency.head(30).plot(x='word', y='freq', kind='bar', figsize=(15, 7), color = 'blue')
plt.title("Most Frequently Occuring Words - Top 30")
```



7. Code

Link to the ipython notebook :

https://drive.google.com/file/d/1De2xbMcwaBKy2MrmLZbdMfdrlGhcXogU/view?usp=sharing

8. Result

Classification Report

	Precision	Recall	F1-score	Support
0	0.73	0.75	0.74	8731
1	0.80	0.78	0.79	11267
Accuracy			0.77	19998
Micro avg	0.76	0.76	0.76	19998
Weighted avg	0.77	0.77	0.77	19998

Comparison (unigram only)

Normalization	Stop Words	Vectorizer	Accuracy
None	No	Count	0.7601260126012601
Lemmatizing	No	Count	0.757025702570257
Stemming	No	Count	0.755025502550255
Lemmatizing	Yes	Count	0.7483748374837483
None	No	Tf-Idf	0.7480748074807481
Stemming	Yes	Count	0.7477747774777478
Lemmatizing	Yes	Tf-Idf	0.7456245624562456
Stemming	No	Tf-Idf	0.7435743574357436
Lemmatizing	No	Tf-Idf	0.7403740374037404
Stemming	Yes	Tf-Idf	0.7383738373837384

9. Conclusion

The given tweets were a combination of words, emojis, URLs, hashtags, client notices, and images. Prior to preparing the dataset we pre-handled the tweets to make it reasonable for taking care of models. We carried out AI calculations Naive Bayes Classifier to order the extremity of the tweet. We utilized three sorts of elements specifically unigrams, bigrams and trigrams for characterization and saw that increasing the component vector with bigrams worked on the precision and trigrams debased the exactness. When the element has been removed it is addressed as a scanty vector. It has been seen that presence in the scanty vector portrayal recorded a preferable presentation over recurrence. We at last accomplished an accuracy of 76.66% utilizing Naive Bayes classifier with the fitting settings and 78.17% accuracy with the LogisticRegression model.

In [16]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(final_vectorized_data, df[sentiment],
                                                    test_size=0.25, random_state=1)
```

In [17]:
```
from sklearn.naive_bayes import MultinomialNB  # Naive Bayes Classifier

model_naive = MultinomialNB().fit(X_train, y_train)
predicted_naive = model_naive.predict(X_test)
```

In [19]:
```
from sklearn.metrics import accuracy_score

score_naive = accuracy_score(predicted_naive, y_test)
print("Accuracy with Naive-bayes: ",score_naive)
```

```
Accuracy with Naive-bayes:  0.7666613329066325
```

Logistic Regression Model

In [20]:
```
from sklearn.linear_model import LogisticRegression
LogReg_clf = LogisticRegression(random_state = 0)

LogReg_clf.fit(X_train, y_train)
y_pred = LogReg_clf.predict(X_test)
```

```
C:\Users\Rishabh Raj\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning: Default solver will be
changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

In [21]:
```
from sklearn.metrics import confusion_matrix, accuracy_score


acc =  accuracy_score(y_test, y_pred)
print("Accuracy here is : ",acc)
```

```
Accuracy here is :  0.7817025362028962
```

10. References

1) Sahayak, V., Shete, V., & Pathan, A. (2015). Sentiment analysis on twitter data. International Journal of Innovative Research in Advanced Engineering (IJIRAE), 2(1), 178-183.
2) Lima, A. C., & de Castro, L. N. (2012, November). Automatic sentiment analysis of Twitter messages. In 2012 Fourth International Conference on Computational Aspects of Social Networks (CASoN) (pp. 52-57). IEEE.
3) Rathi, M., Malik, A., Varshney, D., Sharma, R., & Mendiratta, S. (2018, August). Sentiment analysis of tweets using machine learning approach. In 2018 Eleventh international conference on contemporary computing (IC3) (pp. 1-3). IEEE.
4) Desai, M., & Mehta, M. A. (2016, April). Techniques for sentiment analysis of Twitter data: A comprehensive survey. In 2016 International Conference on Computing, Communication and Automation (ICCCA) (pp. 149-154). IEEE.
5) Khairnar, J., & Kinikar, M. (2013). Machine learning algorithms for opinion mining and sentiment classification. International Journal of Scientific and Research Publications, 3(6), 1-6.
6) https://www.analyticsvidhya.com/blog/2021/06/twitter-sentiment-analysis-a-nlp-use-case-for-beginners/