

# SENTIMENT ANALYSIS ON TWITTER POSTS

<sup>1</sup>Ravikant Patil, <sup>2</sup>Manish Kumar, <sup>3</sup>Dr. Jyoti Kharade, <sup>4</sup>Rasika Patil

<sup>1</sup>Bharati Vidyapeeth's Institute of Management & Information Technology, Mumbai University.

<sup>2</sup> Bharati Vidyapeeth's Institute of Management & Information Technology, Mumbai University.

<sup>3</sup>BVIMIT Associate Professor, Navi Mumbai

<sup>4</sup>BVIMIT Associate Professor, Navi Mumbai

## ABSTRACT

This project aims to perform a Twitter sentiment analysis i.e. mining Tweets through live streaming data using Twitter API modulus and prepare a report on determining whether a piece of writing is positive, negative, or neutral. The approach is to collect data from live Twitter feeds on topics related to public opinions about products, services, brands, politics, or any topic that people can express opinions about. This data can be very useful for commercial applications like marketing analysis, public relations, product reviews, net promoter scoring, product feedback, and customer service. The project would use Python, Twitter API for data extraction, and regex for data cleaning. Twitter provides a streaming API to stream real-time Twitter data. This library named Tweepy is available in python to access Streaming API and download Twitter data. The filter on this data is based on a list of keywords supplied.

**Keywords:** SCORM, Learning Objects, Aggregation, Sequencing

## INTRODUCTION

Sentiment Analysis is the process of 'computationally' determining whether a piece of writing is positive, negative, or neutral. Also known as "Opinion Mining", *Sentiment Analysis* refers to the use of Natural Language Processing to determine the attitude, opinions, and emotions of a speaker, writer, or other subjects within an online mention.

Essentially, it is the process of determining whether a piece of writing is positive or negative. This is also called the Polarity of the content.

As humans, we can classify text as positive/negative subconsciously. For example, the sentence "The kid had a gorgeous smile on his face", will most likely give us a positive sentiment. In layman's terms, we kind of arrive at such a conclusion by examining the words and averaging out the positives and the negatives. For instance, the words "gorgeous" and "smile" are more likely to be positive, while words like "the", "kid" and "face" are neutral. Therefore, the overall sentiment of the sentence is likely to be positive.

A common use for this technology comes from its deployment in the social media space to discover how people feel about certain topics, particularly through users' word-of-mouth in textual posts, or in the context of Twitter, their *tweets*.

"It is the application of Natural Language Processing (NLP) that builds systems that try to identify and extract the sentiment content or opinions behind a series of the word." Usually, besides identifying the opinion, these systems extract attributes of the expression e.g.:

These systems extract attributes of the expression e.g.:

- Polarity: if the speaker expresses a positive or negative opinion,
- Subject: the thing that is being talked about,
- Opinion holder: the person, or entity that expresses the opinion.

With the help of sentiment analysis systems, this unstructured information could be automatically transformed into structured data of public opinions about products, services, brands, politics, or any topic that people can express opinions about.

## 1. Types of Sentiment Analysis

There are many types of sentiment analysis and it ranges from systems that focus on polarity (positive, negative, neutral) to systems that detect feelings and emotions (*angry, happy, sad*, etc.) or identifies intentions (e.g. *interested* v. *not interested*).

The various types of sentiment classification, explore how to convert unstructured text into structured opinions, and address the current challenges in the problem.

Fine-grained Sentiment Analysis  
Emotion detection  
Aspect-based Sentiment Analysis

## Intent analysis

### Fine-grained Sentiment Analysis

Sometimes we may be interested in being more precise about the level of the polarity of the opinion, so instead of positive, neutral, or negative opinions you could consider the following categories:

- Very positive
- Positive
- Neutral
- Negative
- Very negative

Ex: mapped onto a 5-star rating in a review, e.g.: Very Positive = 5 stars and Very Negative = 1 star

### Emotion detection

Emotion detection aims at detecting emotions like happiness, frustration, anger, sadness, and the like.

Many emotion detection systems resort to lexicons (i.e. lists of words and the emotions they convey) or complex machine learning algorithms.

One of the downsides of resorting to lexicons is that the way people express their emotions varies a lot and so do the lexical items they use.

### Aspect-based Sentiment Analysis

ABSA is based on identifying aspects of given target entities and estimating the sentiment polarity for each mentioned aspect. This can be decomposed into two tasks: aspect extraction and aspect sentiment classification. Aspect extraction pertains to recognizing aspects of the entity and more generally can be seen as an information extraction task. Aspect sentiment classification determines whether the opinions on different aspects are positive, negative, or neutral.

I.e., when analyzing the sentiment in subjects, for example, products, you might be interested in not only whether people are talking with a positive, neutral, or negative polarity about the product, but also which particular aspects or features of the product people talk about. That's what aspect-based sentiment analysis is about.

Example: *"The battery life of this camera is too short."*

The sentence is expressing a negative opinion about the camera, but more precisely, about the battery life, which is a particular feature of the camera.

## Intent analysis

The intent analysis detects what people want to do with a text rather than what people say with that text.

Examples:

*"Your customer support is a disaster. I've been on hold for 20 minutes".*

*"I would like to know how to replace the cartridge".*

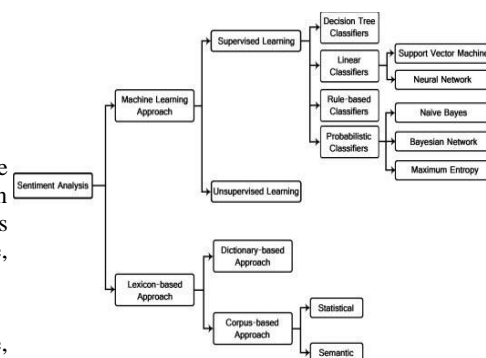
*"Can you help me fill out this form?"*

A human being has no problems detecting the complaint in the first text, the question in the second text, and the request in the third text. However, machines can have some problems identifying those. Sometimes, the intended action can be inferred from the text, but sometimes, inferring it requires some contextual knowledge.

## 2) Sentiment Analysis Algorithms

There are many methods and algorithms to implement sentiment analysis systems, which can be classified as:

- **Rule-based** systems that perform sentiment analysis based on a set of manually crafted rules.
- **Automatic** systems that rely on machine learning techniques to learn from data.
- **Hybrid** systems that combine both rule-based and automatic approaches.



**Fig No. 01**

Rule-based Approaches

Usually, rule-based approaches define a set of rules in some kind of scripting language that identifies subjectivity, polarity, or the subject of an opinion.

The rules may use a variety of inputs, such as the following:

Classic NLP techniques like *stemming*, *tokenization*, *part of speech tagging*, and *parsing*.

Other resources, such as lexicons (i.e. lists of words and expressions).

A basic example of a rule-based implementation would be the following:

1. Define two lists of polarized words (e.g. negative words such as *bad*, *worst*, *ugly*, Etc. and positive words such as *good*, *best*, *beautiful*, etc.).
2. Given a text:
  1. Count the number of positive words that appear in the text.
  2. Count the number of negative words that appear in the text.
3. If the number of positive word appearances is greater than the number of negative word appearances returns a positive sentiment, conversely returns a negative sentiment. Otherwise, return neutral.

This system is very naïve since it doesn't take into account how words are combined in a sequence. More advanced processing can be made, but these systems get very complex quickly. They can be very hard to maintain as new rules may be needed to add support for new expressions and vocabulary. Besides, adding new rules may have undesired outcomes as a result of the interaction with previous rules. As a result, these systems require important investments in manually tuning and maintaining the rules.

### Automatic Approaches

Automatic methods, contrary to rule-based systems, don't rely on manually crafted rules, but on machine learning techniques. The sentiment analysis task is usually modeled as a classification problem where a classifier is fed with a text and returns the corresponding category, e.g. positive, negative, or neutral (in case polarity analysis is being performed).

Said machine learning classifier can usually be implemented with the following steps and components:

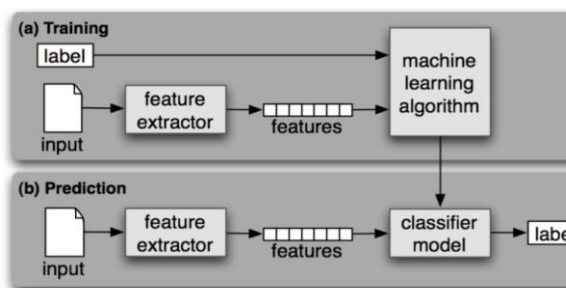


Fig No. 02

### The Training and Prediction Processes

In the training process (a), our model learns to associate a particular input (i.e. a text) to the corresponding output (tag) based on the test samples used for training. The feature extractor transfers the text input into a feature vector. Pairs of feature vectors and tags (e.g. *positive*, *negative*, or *neutral*) are fed into the machine learning algorithm to generate a model.

In the prediction process (b), the feature extractor is used to transform unseen text inputs into feature vectors. These feature vectors are then fed into the model, which generates predicted tags (again, *positive*, *negative*, or *neutral*).

### Feature Extraction from Text

The first step in a machine learning text classifier is to transform the text into a numerical representation, usually a vector. Usually, each component of the vector represents the frequency of a word or expression in a predefined dictionary (e.g. a lexicon of polarized words). This process is known as feature extraction or text vectorization and the classical approach has been bag-of-words or n-grams with their frequency.

More recently, new feature extraction techniques have been applied based on word embeddings (also known as *word vectors*). This kind of representation makes it possible for words with similar meanings to have a similar representation, which can improve the performance of classifiers.

### Classification Algorithms

The classification step usually involves a statistical model like Naïve Bayes, Logistic Regression, Support Vector Machines, or Neural Networks:

Naïve Bayes: a family of probabilistic algorithms that uses Naive Bayes's Theorem to predict the category of a text.

Linear Regression: a very well-known algorithm in statistics used to predict some value (Y) given a set of features (X).

Support Vector Machines: a non-probabilistic model which uses a representation of text examples as points in a multidimensional space. These examples are mapped so that the examples of the different categories (sentiments) belong to distinct

regions of that space. Then, new texts are mapped into groups-positive, negative, good, bad, like, and dislike. onto that same space and predicted to belong to a category based on which region they fall into.

**Deep Learning:** a diverse set of algorithms that attempts to imitate how the human brain works by employing artificial neural networks to process data.

## Hybrid Approaches

The concept of hybrid methods is very intuitive: just combine the best of both worlds, the rule-based and the automatic ones. Usually, by combining both approaches, the methods can improve accuracy and precision.

## Methodology

### How does Sentimental Analysis Works?

In General, There are 5 steps to analyze sentiment data:



#### • Data Collection

Consumers usually express their sentiments on public forums like blogs, cussion boards, and product reviews as well as on their private logs – Social network sites like Facebook and Twitter. Opinions and feelings are expressed differently, with different vocabulary, the context of writing, usage of short forms, and slang, making the data huge and disorganized. Manual analysis of sentiment data is virtually impossible. Therefore, special programming languages like ‘R’ are used to process and analyze the data.

#### • Text Preparation

Text preparation is nothing but filtering the extracted data before analysis. It includes identifying and eliminating non-textual content and content that is irrelevant to the area of study from the data.

#### • Sentiment Detection

At this stage, each sentence of the review and opinion is examined for subjectivity. Sentences with subjective expressions are retained and that convey objective expressions are discarded. Sentiment analysis is done at different levels using common computational techniques like Unigrams, lemmas, negation, and so on.

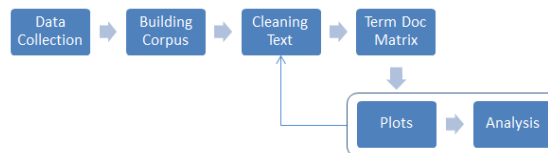
#### • Sentiment Classification

Sentiments can be broadly classified into two groups, positive and negative. At this stage of sentiment analysis methodology, each subjective sentence detected is classified

#### • Presentation of Output

The main idea of sentiment analysis is to convert unstructured text into meaningful information. After the completion of the analysis, test text results are displayed on graphs like pie charts, bar charts, and line graphs.

How does Sentimental Analysis work in R Programming/Python?



#### Data Collection

Collect the data from Social media like Twitter, Facebook, YouTube, etc., and read it into R.

For Twitter, we use the Twitter package for data collection SocialMediaLab package for YouTube, Facebook, etc.

Building a Corpus

Corpus is defined as a “collection of text documents”. The plural form of the corpus is corpora

totm package (text mining package) uses the corpus( ) function to create a corpus

Text cleaning

A significant amount of techniques is applied to data to reduce the noise of text, reduce dimensionality, and assist in the improvement of classification effectiveness. The most popular techniques include:

Lowercase

Remove numbers, symbols

Remove punctuation

Stemming

Part of speech tagging

Remove Stopwords

Remove the white space

**Term Document Matrix**

• “The term-document matrix then is a 2-dimensional matrix whose rows are the terms & columns are the documents, so each entry (i, j) represents the frequency of term i in document j.”

• Term Document Matrix (TDM) as an implementation of the Bag of Words concept. It converts unstructured data into structured data. i.e., it creates a numerical representation of the documents in our corpus.

We use tm package for TDM

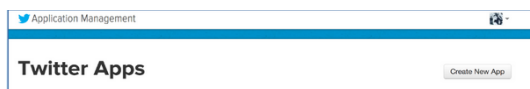
Plot and Analysis

We make use of bar plot for plotting and we do sentimental analysis

After plotting, when we come across unwanted words/text that we don't need, we sent it back for cleaning. Once the cleaning process is finished, again we need to define the TDM, then plot, and finally sentimental analysis

## 1. Getting Twitter API keys

- To be able to gather the tweets from Twitter, we need to get the Twitter API Keys.
- To use Twitter's API, we have to create a developer account on the Twitter apps site
  - Log in or make a Twitter account at <https://apps.twitter.com/>.
  - Create a new app (button on the top right).



- Fill in the app creation page with a unique name, a website name (use a placeholder website if you don't have one), and a project description. Accept the terms and conditions and proceed to the next page

## Create an application

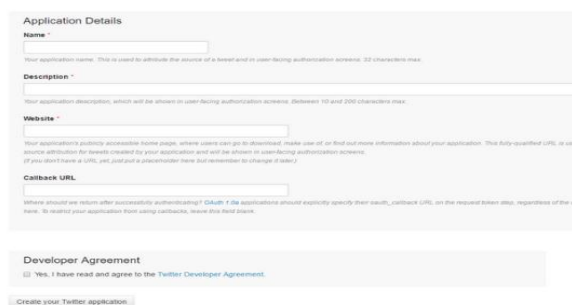


Fig No. 03

- After registering, create an access token and grab your application's Consumer Key, Consumer Secret, Access token, and Access token secret from the Keys and Access Tokens tab.

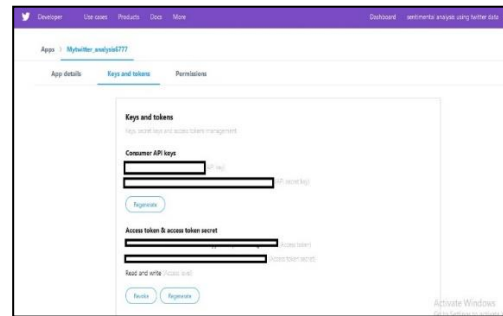


Fig No. 04

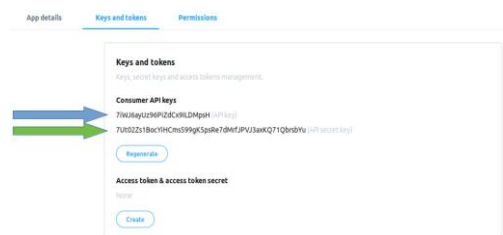


Fig no. 05

Now we are moving towards Jupyter Notebook

Installing Twitter library

We will be using a Python library called **Twitter** to connect to Twitter API and download the data from Twitter. There are many other libraries in various programming languages that let you use Twitter API. We choose theater for this tutorial because it is simple to use yet fully supports the Twitter API.

**Install Twitter by using pip to pull it from PyPI:**

```
$ pip install twitter
```

Import the Twitter API module

```
import twitter
```

To use authentication, instantiate Twitter. API as with a

consumer key and secret; and the OAuth key and secret:

```
In [7]: api = twitter.Api(consumer_key='W0L1E3jQV01dDg3aPwR09Pp+',
                        consumer_secret='W0L1E3jQV01dDg3aPwR09Pp+',
                        access_token='W0L1E3jQV01dDg3aPwR09Pp+',
                        access_token_secret='W0L1E3jQV01dDg3aPwR09Pp+')

In [8]: print(api.VerifyCredentials())
```

Verify whether the connection established or not

```
print(API.VerifyCredentials())
```

Output: If the connection built



```
print(api.VerifyCredentialInfo())
```

```
{
  "created_at": "Fri Jan 26 21:11:08 0000 2018",
  "default_profile": true,
  "default_profile_image": true,
  "followers_o": 1,
  "friends_count": 1,
  "id": "95700047633507328",
  "id_str": "95700047633507328",
  "name": "Anil Kumar",
  "profile_background_color": "F5F5F5",
  "profile_image_url": "https://s3.amazonaws.com/twitter/default/profile_image/default_profile_image.png",
  "profile_image_url_https": "https://s3.amazonaws.com/twitter/default/profile_image/default_profile_image.png",
  "profile_link_color": "3498db",
  "profile_sidebar_fill_color": "FFFFFF",
  "profile_sidebar_border_color": "34495E",
  "profile_sidebar_fill_color": "34495E",
  "profile_text_color": "333333",
  "profile_use_background_image": true,
  "screen_name": "AnilKun692"
}
```

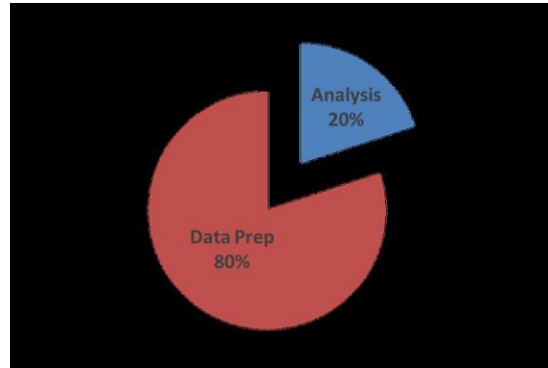
**Fig No. 06**

**To check trending hashtags based on location:**

```
In [32]: locationTrending = api.GetTrendsWeird('23424048', exclude=None)

In [33]: trending = []
         [trending.append(trend.name) for trend in locationTrending]
         print("Trending")
         trending

Trending
```



**Fig No. 07**

### Output The trending list

[illegible]

**Fig No. 08**

### Extract the Tweets fTwittertter:

The tweets from the Twitter are extracted using the Hashtags, Usernames, or according to the location

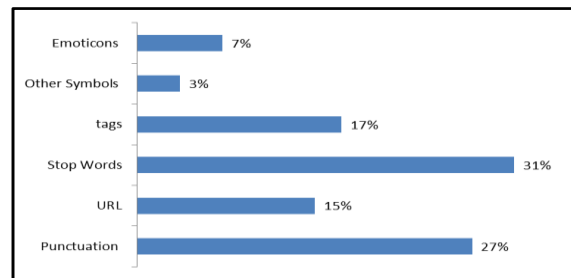
```
In [18]: def createTestData(search):
try:
    fetchData = api.GetSearch(search, count=1000)
    print "We extract " + str(len(fetchData)) + " for " + search
    return [{"text":status.text} for status in fetchData]
    print ("In try")
except:
    print("Data is not Extracted")
    return None

search_string = input("Enter the term u want to search")
data = createTestData(search_string)

Enter the term u want to searchINDWSND
We extract 100 for INDWSND
```

In this project text collected from Twitter, the website is cleaned using the Regex library in python, and Stopword is removed using nltk. corpus. The available text mining package in Regex helps to remove all the unwanted data step by step. The implementation of the same can be inspected very easily using inspect command.

The following chart shows the percentage of various types of unwanted data which was present in the text top 10 from the website tweets.



### To fetch your friends (after being authenticated):

```
>>> users = api.GetFriends()
```

```
>>> print([u.name for u in users])
```

## Data Cleaning

One of the first steps when working with the data related to text mining is data preprocessing. It is one of the essential steps required before data is ready for the analysis as the majority of available text data is very noisy, stop-words, another language, and highly not in structure. So to get the best insights and information from the available data and also to build better algorithms it is very important to clean the data. This is because the social network is most of the time used for informal communication where typing errors bad grammar uses of slang presence of unwanted content which includes URL stop words and many expressions.

It is easily observed that most of the cleaning is done to remove the stop words in the English language. The second most where's the punctuation symbols which are most of the time used in the tweets. URL in the tweets and tags symbols combined takes around one-third of the unwanted data. The letter with a size is less than 2 is removed. The word other than English is removed i.e. containing digits and other language words by using a simple regex program. The libraries used for cleaning the data:

```
import re
```

```
import nltk
```

```
from nltk import sent_tokenize
```

```
from string import punctuation
```

```
from nltk.tokenize import word_tokenize
```

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

```
In [19]: stopwords.words('english') + list(punctuation) + ['AT_USER', 'URL']
df =
for i in range(len(data)):
    df[df['text']] = df['text'].lower()
    df = re.sub('http://[^\s]+', '', df) # remove URLs
    df = re.sub('@[^\s]+', '', df) # remove the # in hashtag
    df = re.sub('^\s+', '', df) # remove words then English
    df = re.sub('^\s+', '', df) # Remove words with 2 or fewer letters
    df = re.sub('^\s+', '', df) # Remove whitespace (including new line characters)
    df = ''.join(c for c in df if c <= '\uffff')
```

```
In [20]: paramword_tokenize(df)
para
['corn',
 'will',
 'pick',
 'winners',
 'and',
 'give',
 'emerson',
 'troubled',
 'hushy',
 'claw',
 'last',
 'fault',
 'trend',
 'mean',
 'rider',
 'pakistanis',
 'support',
 'mean',
 'hel']
```

Before we move on to the actual classification section, there is some cleaning up to do. This step is critical and usually takes a long time when building Machine Learning models. However, this will not be a problem in our task, as the data we have is relatively consistent. In other words, we know exactly what we need from it. I will express this matter later on.

Let's talk about what matters and what doesn't matter in Sentiment Analysis. Words are the most important part (to an extent that we will talk about in the upcoming section). However, when it comes to things like punctuation, you cannot get the sentiment from punctuation. Therefore, punctuation does not matter to Sentiment Analysis. Moreover, tweet components like images, videos, URLs, usernames, emojis, etc. do not contribute to the polarity (whether it is positive or negative) of the tweet. However, this is only true for this application. For instance, in another application, you could have a Deep Learning image classifier that learns and predicts whether this image that the tweet contains stands for something positive (e.g. a rainbow) or negative (e.g. a tank). When it comes to technicality, both Sentiment Analysis and Deep Learning fall under Machine Learning. You can perform Sentiment Analysis through Deep Learning, but that's a story for another day.

So we know what we need to keep in the tweets we have and what we need to take out. This applies to both Training and Test sets. So let's make our pre-processor class:

That was a handful, so let's break it down into parts. We start with imported libraries. re is Python's Regular Expressions (RegEx) library, which takes care of parsing strings and efficiently modifying them without having to explicitly iterate through the characters comprising the particular string. We also imported nltk, i.e. Natural Processing Toolkit, which is one of the most commonly used Python libraries out

there. It takes care of any processing that we need to perform on text to change its form or extract certain components from it. The class constructor removes stop words. This is a relatively big topic that you can read up on later, as it is more into Natural Language Processing and less related to our topic.

The process Tweets function just loops through all the tweets input into it, calling its neighboring function process Tweet on every tweet in the list. The latter does the actual pre-processing by first making all the text in lower-case letters. This is mere because, in almost all programming languages, "cAr" is not interpreted the same way as "car". Therefore, it is better to normalize all characters to be lower-case across all our data. Secondly, URLs and usernames are removed from the tweet. This is for the reasons we disclosed earlier in the article. Afterward, the number sign (i.e. #) is removed from every hashtag, to avoid hashtags being processed differently. Last but not least, duplicate characters are rid of, to ensure that no important word goes unprocessed even if it is spelled out unusually (e.g. "car" becomes "car"). Finally, the tweet's text is broken into words (tokenized) to ease its processing in the upcoming stages.

Let's take an example. The following tweet could be present in the data set:

"@person1 retweeted @person2: Corn has got to be the most delicious crop in the world!!!! #corn #thoughts..."

Our pre-processor will result in the tweet looking like this:

"AT\_USER rt AT\_USER corn has got to be the most delicious crop in the world corn thoughts"

And finally, the tokenization will result in:

```
{“corn”, “most”, “delicious”, “crop”, “world”, “corn”, “thoughts”}
```

Note that our code removed duplicate characters in words as we mentioned earlier (i.e. “delicious” became “delicious”). However, it did not remove duplicate words (i.e. “corn”) from the text, but rather kept them. This is because duplicate words play a role in determining the polarity of the text (as we will see in the upcoming section).

## 1. Process of Analyzing tweets in Python

The use of sentiment analysis is becoming more widely leveraged because the information it yields can result in the monetization of products and services. Organizations use surveys, opinion polls, and social media as a mechanism to obtain feedback on their products and services. Sentiment analysis or opinion mining is the computational study of opinions, sentiments, and emotions expressed in text. The reviews or opinions can be positive or negative and analyzing the same is known as ‘Sentiment Analysis’.

For this text analysis purpose did text mining package is available in python. The Corpus is needed to be created using this text mining package on Twitter. Now, this Corpus is a collection of documents containing tweets labeled as positive, negative, or neutral. The corpus tweets are trained using the Naive Bayes Classifiers I.e. the labeled data is trained and the model is built.

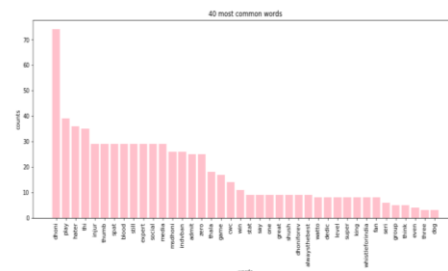
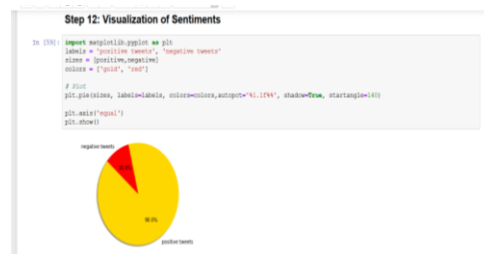
After defining the Corpus the data preprocessing task is completed i.e. data is clean

using nltk and regex. The tweets that are tested i.e positive negative or neutral are predicted and the accuracy is checked. The most frequent 40 words are shown in the histogram and word cloud the most frequent word in the tweets appeared larger according to their frequency in our text set. The last step represents of sentiment result i.e. the polarity of the tweets and shown in the pie chart

## Stemming Of Words :

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”, and “choco” to the root word, “chocolate” and “retrieval”, “retrieved”, “retrieves” reduce to the stem “retrieve”.

## Top 40 Most Frequent Words in #Dhoni



We see that the word **haters** is the plural form of **hater** so, by stemming of words we merge the same type of word.

## Word Cloud :

Word Cloud is a data visualization technique used for representing text data in which the size of each word indicates its frequency or importance. Significant textual data points can be highlighted using a word cloud. Word clouds are widely used for analyzing data from social network websites.

For generating word cloud in Python, modules needed are – matplotlib, pandas, and word cloud. To install these packages, run the following commands :

```
pip install matplotlib
```

```
pip install pandas
```

```
pip install wordcloud
```

## Visualization of words on Histogram



Frequency of first 40 gram.





## Conclusion

Sentiment detection has a wide variety of applications in information systems, including classifying reviews, summarizing review and other real time applications here are likely to be many other applications that is not discussed. It is found that sentiment classifiers are severely dependent on domains or topics. From the above work it is evident that neither classification model consistently outperforms the other, different types of features have distinct distributions. It is also found that different types of features and classification algorithms are combined in an efficient way in order to overcome their individual drawbacks and benefit from each other's merits, and finally enhance the sentiment classification performance.

In future, more work is needed on further improving the performance measures. Sentiment analysis can be applied for new applications. Although the techniques and algorithms used for sentiment analysis are advancing fast, however, a lot of problems in this field of study remain unsolved. The main challenging aspects exist in the use of other languages, dealing with negation expressions; producing a summary of opinions based on product features/attributes, the complexity of sentence/ document, handling of implicit product features, etc. More future research could be dedicated to these challenges.

## References

- [1] L.Colazzo,A. Molinari and N. Villa. "Collaboration vs. Participation: the Role of Virtual Communities in a Web 2.0 world", International Conference on Education Technology and Computer, 2009, pp.321-325.
- [2] [nlp.stanford.edu/courses/cs224n/2011/reports/patlai.pdf](http://nlp.stanford.edu/courses/cs224n/2011/reports/patlai.pdf)
- [3] National Daily, Economic Times: [articles.economictimes.indiatimes.com/Collections/Facebook](http://articles.economictimes.indiatimes.com/Collections/Facebook)
- [4] J. Read. "Using emoticons to reduce dependency in machine learning techniques for sentiment classification". In Proceedings of ACL-05, 43rd Meeting of the Association for Computational Linguistics. Association for Computational Linguistics, 2005
- [5] K. Dave, S. Lawrence, and D.M. Pennock. "Mining the peanut gallery: Opinion extraction and semantic classification of product reviews". In Proceedings of the 12<sup>th</sup> International Conference on World Wide Web (WWW), 2003, pp. 519–528.
- [6] A. Pak and P.Paroubek. "Twitter as a Corpus for Sentiment Analysis and Opinion Mining". In Proceedings of the Seventh Conference on International Language Resources and Evaluation, 2010, pp.1320–1326.
- [7] R. Parikh and M. Movassate, "Sentiment Analysis of User- Generated Twitter Updates using Various Classification Techniques", CS224N Final Report, 2009
- [9] A. Go, R. Bhayani, L.Huang. "Twitter Sentiment Classification Using Distant Supervision". Stanford University, Technical Paper ,2009
- [10] L. Barbosa, J. Feng. "Robust Sentiment Detection on Twitter from Biased and Noisy Data". COLING 2010: Poster Volume, pp. 36-44.
- [11] S. Batra and D. Rao, "Entity Based Sentiment Analysis on Twitter", Stanford University,2010A. Bifet and E. Frank, "Sentiment Knowledge Discovery in Twitter Streaming Data", In Proceedings of the 13th International Conference on Discovery Science, Berlin, Germany: Springer,2010, pp. 1–15.
- [12] A. Agarwal, B. Xie, I. Vovsha, O. Rambow, R. Passonneau, "Sentiment Analysis of Twitter Data", In Proceedings of the ACL 2011 Workshop on Languages in Social Media,2011, pp. 30–38
- [13] A. Kumar. and T. M. Sebastian, "Sentiment Analysis: A Perspective on its Past, Present and Future", International Journal of Intelligent Systems and Applications (IJISA), MECS Publisher, 2012 (Accepted to be published)
- [14] A. Kumar and T. M. Sebastian, "Machine learning assisted Sentiment Analysis". Proceedings of International Conference on Computer Science & Engineering (ICCSE'2012), 2012, pp. 123-130.