PROGRESS REPORT-1

Sentiment Analysis of Twitter Data Using Python

SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE AWARD OF

Degree of Bachelor of Technology in Computer Science & Engineering

SUBMITTED BY

Name: Anmol Singh Rajput
UID:- 17BCS2898
Name:- Arjun Shrivastva
UID:- 17BCS2910
Name:- Satyam Priyadarshi
UID:- 17BCS2903

SUBMITTED TO:

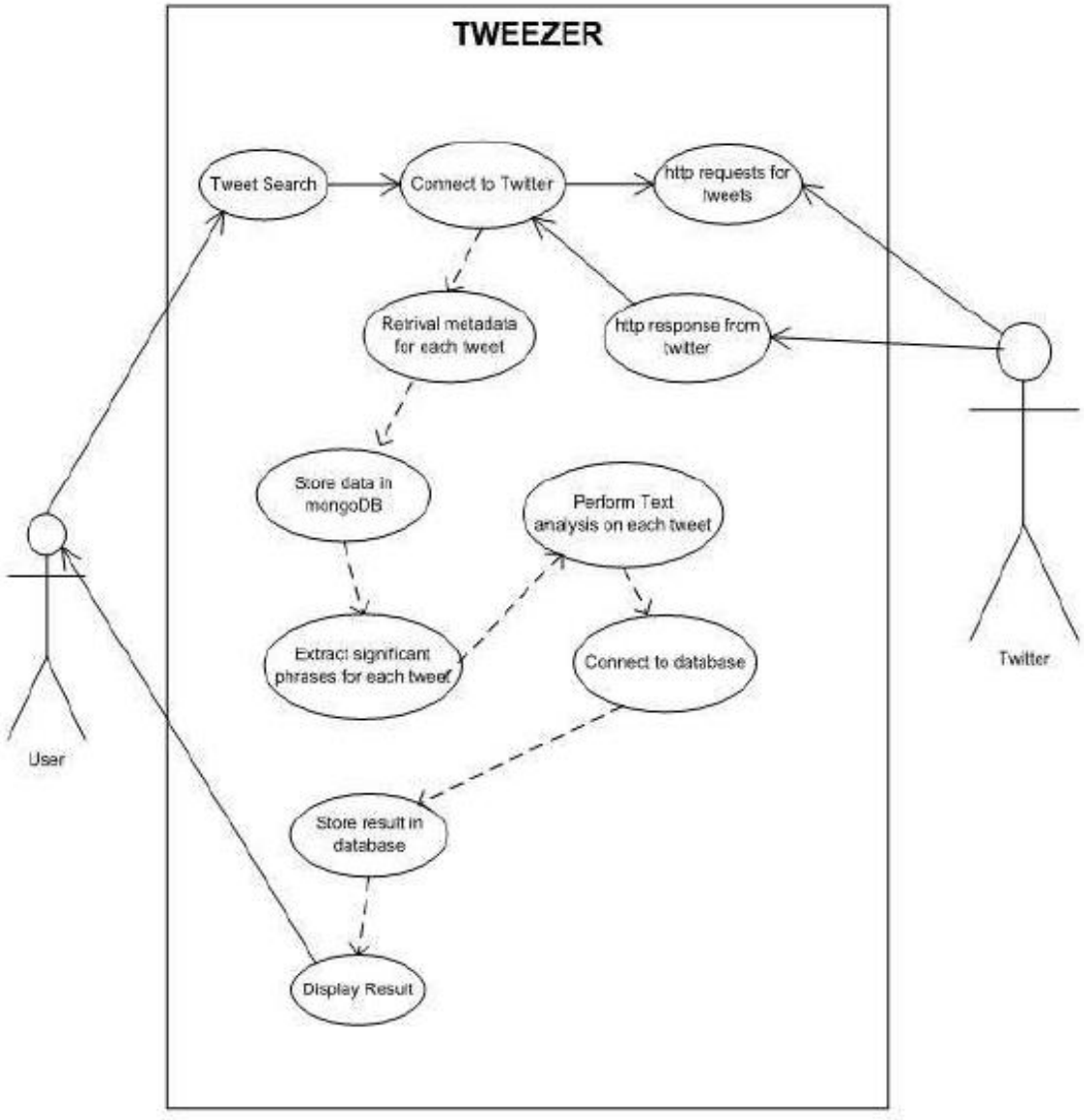
RAJANI MAM

Department of Computer Science & Engineering CHANDIGARH UNIVERSITY
Gharuan, Mohali

1. Project Goal

Social networks are the main resources to gather information about people's opinion and sentiments towards different topics as they spend hours daily on social medias and share their opinion. In this project, we show the application of sentimental analysis and how to connect to Twitter and run sentimental analysis queries. We run experiments on different queries from politics to humanity and show the interesting results. We realized that the neutral sentiment for tweets are significantly high which clearly shows the limitations of the current works. It will be helpful to political party for reviewing about the program that they are going to do or the program that they have performed. Similarly, companies also can get review about their new product on newly released hardware's or software's. Also, the movie maker can take review on the currently running movie. By analysing the tweets analyser can get result on how positive or negative or neutral are peoples about it.

2.Project Design: -



3. Project Work done till date: -

Till now we have done two things, we have work on implement an algorithm for automatic classification of text into positive and negative and Sentiment Analysis to determine the attitude of the mass is positive, negative or neural towards the subject of interest.

4.Total numbers of modules/phases in the project: -

1. Analysis tweets post by different peoples on twitter.
2. Date collection regarding tweets
3. Classification of data: -
    - Native Bayes Classifier (NB) Algorithm
4. A training set for each class of data Phase
5. Tree Diagram analysis of Native Bayes Classifier
6. Graph Isolation for Native Bayes Classifier
7. Analysis tweets segments as (positive / Negative /neural) using ML(Python) Phase
8. Documentation
9. IEEE Paper

5. Number of modules covered till date: -

1. Analysis tweets post by different peoples on twitter.
2. Date collection regarding tweets
3. Classification of data: -
    - Native Bayes Classifier (NB) Algorithm

6. Description of covered modules: -

1.Analysis tweets post by different peoples on twitter: - Measuring your Twitter data can be done through different avenues. Depending how robust you want your analytics, there are diverse options to give you oversight or in-depth analysis.

Twitter provides a robust monthly review for users to see the performance of their content. Some of the data you can see from the home section includes:

- **Tweets:** Total number of Tweets you sent.
- **Tweet impressions:** Total number of times a user was served your Tweet in their timeline or search results (including whether it was seen or not).
- **Profile visits:** Total number of times your profile was clicked on from your Tweets or through search.
- **Mentions:** Total number of times your Twitter handle has been used in other users' Tweets.
- **Followers:** Total number of followers, plus how many new followers you've gained since the previous period.

2. Native Bayes Classifier (NB) Algorithm: -

The Naïve Bayes classifier is the simplest and most commonly used classifier. Naïve Bayes classification model computes the posterior probability of a class, based on the distribution of the words in the document. The model works with the BOWs feature extraction which ignores the position of the word in the document.

$$P(\text{label}|\text{features}) = \frac{P(\text{label}) * P(\text{features}|\text{label})}{P(\text{features})}$$

P(*label*) is the prior probability of a label or the likelihood that a random feature set the label. *P*(*features*|*label*) is the prior probability that a given feature set is being classified as a label. *P*(*features*) is the prior probability that a given feature set is occurred. Given the Naïve assumption which states that all features are independent, the equation could be rewritten as follows:

$$P(\text{label}|\text{features}) = \frac{P(\text{label}) * P(f1|\text{label}) * \ldots\ldots\ldots * P(\text{fn}|\text{label})}{P(\text{features})}$$

Accuracy – around 75%

Algorithm:

- **Dictionary generation: -**Count occurrence of all word in our whole data set and make a dictionary of some most frequent words.

- **Feature set generation:** -All document is represented as a feature vector over the space of dictionary words. For each document, keep track of dictionary words along with their number of occurrences in that document.

Formula used for algorithms:

$$\phi_{k|label=y} = P(x_j = k \mid label = y)$$

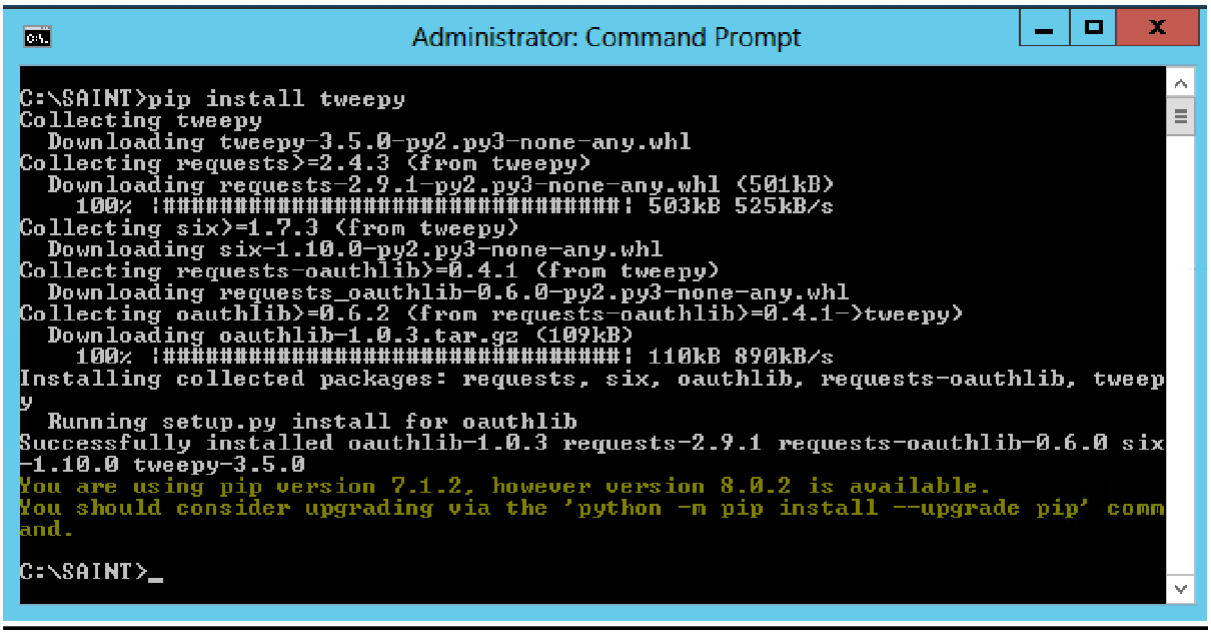
$$\phi_{k|label=y} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n_i} 1\{x_j^{(i)} = \text{k and } label^{(i)} = y\} + 1}{(\sum_{i=1}^{m} 1\{label^{(i)} = y\} n_i) + |V|}$$

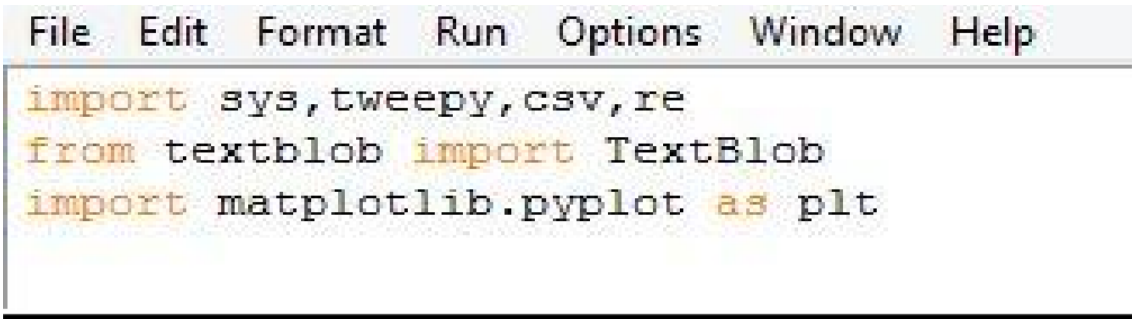
$\phi_{k|label=y}$ = probability that a particular word in document of label(neg/pos) = y will be the k^{th} word in the dictionary.

m = Number of words in i^{th} document.

n_i = Total Number of documents.

7. Screenshots of running project till date





8. Further Directions (Pending work details)

Our projects success depends when we integrate these 3 modules out of which 2 have been already made and 1 is remaining and then most tough part of us project that is combining these 3 is also remaining and after combining them only Real time detection and Documents will be left.