Abstract

A sentiment can be defined as a personal positive or negative feeling. Opinion mining is the computational technique for extracting data, classifying it then understanding, and assessing the opinions expressed in various contents. Current era is of social networking sites; petabytes of data is generated daily on them. Millions of people are posting their [1] likes, dislikes, comments daily on social networking sites. In this system, we are proposing a model that will extract the sentiment from a famous micro blogging site, Twitter, where users post their opinions for everything. Proposed model uses modified version of Naïve Bayes machine learning algorithm. Our modifications introduce neutral class by considering probability intersection between positive and negative classes. Algorithm results are improved by reducing words in tweet to their root form through mechanism of pre-processing before passing them to sentiment analyzer. Hence, proposed system classifies tweets as positive, negative or neutral with respect to a query term. This is very useful for the companies who want to know the feedback about their product brands or customers who want to search the opinion from others about product before purchase or also for election exit polls.

Contents

List of Figures

List of Tables

List of Nomenclatures

SENTAL : Application for Machine Learning Approach for Sentimental Analysis of Twitter Feeds using Hadoop Framework.

Chapter 1

Project Overview

Opinion mining is the computational technique for extracting data, classifying it then understanding, and assessing the opinions expressed in various contents. Current era is of social networking sites. There is around petabyte of data generated daily on these sites. Millions of people are posting their likes, dislikes, comments daily on social networking sites. Twitter is one of the social networking giants. [2] In this system, we are proposing a model that will extract the sentiment from Twitter- a famous microblogging site- where users post their opinions for everything. Proposed model uses modified version of Naïve Bayes machine learning algorithm.

Our modifications introduce Neutral class by considering probability intersection between positive and negative classes. Algorithm results are improved by reducing words in tweet to their root form through mechanism of pre-processing before passing them to sentiment analyzer. Hence, proposed system classifies tweets as positive, negative or neutral with respect to a query term. This is very useful for the companies who want to know the feedback about their product brands or customers who want to search the opinion from others about product before purchase or also for election exit polls.

Chapter 2

Introduction and Motivation

2.1 Introduction

In the past decade, new forms of communication, such as micro-blogging and text messaging have emerged and become more popular. While there is no limit to the range of information conveyed by tweets and texts, often these short messages are used to share opinions and sentiments that people have about what is going on in the world around them. Sentiment analysis is a procedure where the dataset consists of emotions, attitudes or assessment which takes into account the way a human thinks.

Sentiment analysis is extremely useful in social media monitoring as it allows us to gain an overview of the wider public opinion behind certain topics.[3] The applications of sentiment analysis are broad and powerful. The ability to extract insights from social data is a practice that is being widely adopted by organizations across the world. Shifts in sentiment on social media have been shown to correlate with shifts in the stock market. The Obama administration used sentiment analysis to gauge public opinion to policy announcements and campaign messages

ahead of 2012 presidential election. Sentiment analysis conducted by the brand revealed that the music played on the commercial had become incredibly irritating after multiple airings, and consumers were flocking to social media to vent their frustrations. A couple of weeks after the advert first aired, over half of online conversation about the campaign was negative.

2.2 Aim and Objective

2.2.1 Aim

- To mine people opinions, by using tweets posted by them on Twitter.
- To modify Naïve Bayes machine learning algorithm to add neutral class

2.2.2 Objective

- To improves accuracy of Naïve Bayes by adding neutral class by probability intersection of positive and negative classes.
- To accurately classifies tweets as positive, negative or neutral with respect to a query term.

2.3 Motivation

The applications of sentiment analysis are tremendous. Many well-known political and administration firms started using the power of opinion mining to improve their productivity. The Obama administration used sentiment analysis to gauge public opinion to policy announcements and campaign messages ahead of 2012 presidential election. Many social networking giants are at front-foot providing vast amount of information. The rise in use of social networking and trending field of data mining, we are encouraged to undertake the analysis of Twitter feeds based on sentiments.

Chapter 3

Problem Statement

3.1 Problem Statement:

The proposed system models machine learning approach for sentimental analysis of Twitter feeds using Hadoop software framework, which improves accuracy of Naïve Bayes by adding neutral class and by elimination of class independence through probability intersection of positive and negative classes.

Chapter 4

Requirement Analysis

4.1 Hardware Requirements

- RAM: Minimum 4GB
- Processor: 2 GHz
- Network: 1 Gigabit Ethernet

4.2 Software Requirements

- O.S.: Ubuntu Precise (12.04) and later
- Network Protocol: IPv4
- CDH 5: Cloudera Hadoop Software Framework
- Oracle JDK 1.7, Python 2.6 or later

- Web Browser
  - Mozilla Firefox 24 or higher
  - Google Chrome
  - IE 9 or higher
  - Safari 5 or higher

Chapter 5

Project Design

5.1 Block Diagram

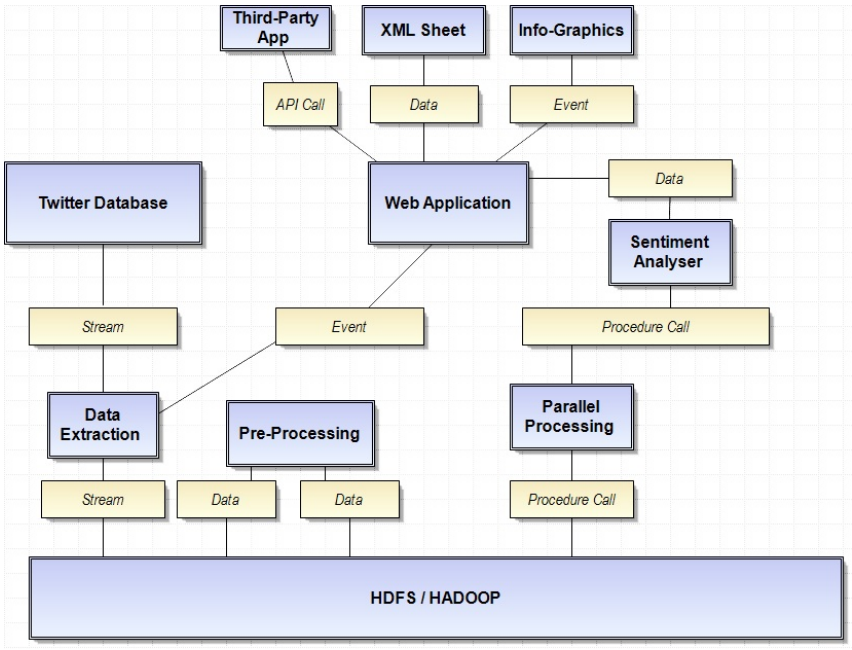


Figure 5.1: SENTAL Architecture

5.2 Data Flow Diagram

The data flow in the proposed system is shown below with the help of data flow diagram consisting of three levels.

User and Sentiment Analyzer are two actors and SENTAL system is the main process. The user passes a desired keyword, of which he/she wants the result of, through the interface. This keyword goes to the process SENTAL system, where it gets processed through other actor- Sentiment Analyzer. The sentiment analyzer processes the keyword returned from SENTAL system to produce the result. The analysis result is passed through SENTAL system and returned to the user as a form of infographics or xml sheet.

5.2.1 Level 0

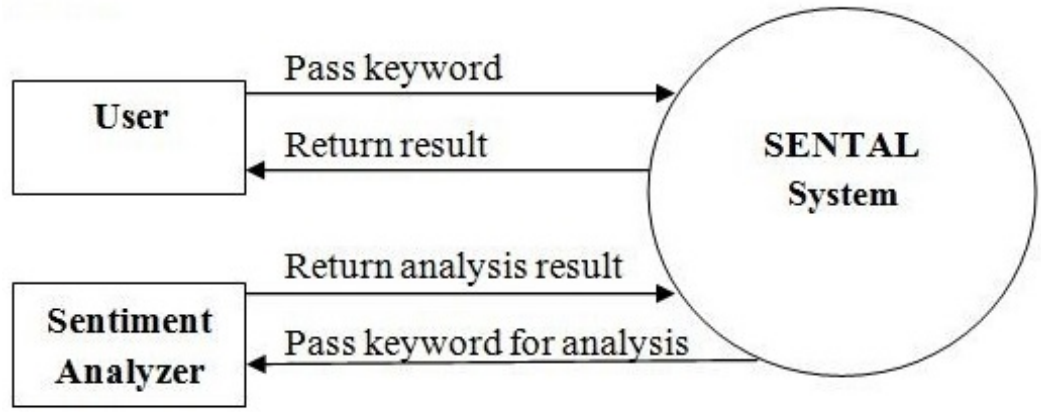


Figure 5.2: DFD Level 0

The actors in the system are User and Sentiment analyzer. SENTAL System is the process. User interacts with the SENTAL system through the Pass keyword and Return result dataflow. Similarly, Sentiment Analyzer interacts with data source through Return result and Pass keyword.

5.2.2 Level 1

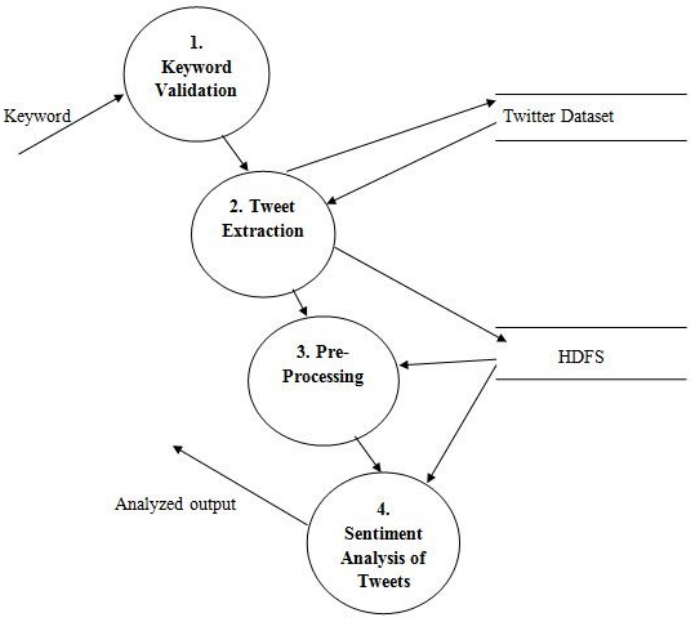


Figure 5.3: DFD Level 1

Entered keyword is validated by Keyword Validation process which is then passed to the Tweet Extraction process. It has access to Twitter dataset data source and HDFS data source. Further pre-processing process gets the raw tweets and gives output to the process Sentiment Analysis as shown in figure 5.2. Analyzed output is given out by the same process.

5.2.3 Level 2

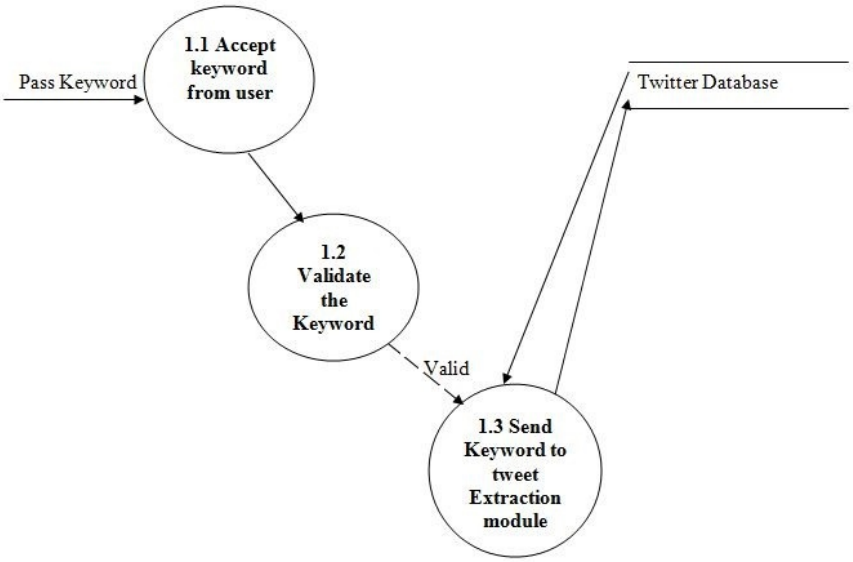


Figure 5.4: DFD Level 2 for tweets extraction

Process 1.1 accepts the keyword from the user and is passed to the process 1.2 to validate the entered keyword. The keyword marked as validated is then passed to process 1.3 where it sends the valid keyword to Tweet Extraction Module. This process has access to the data store Twitter database.

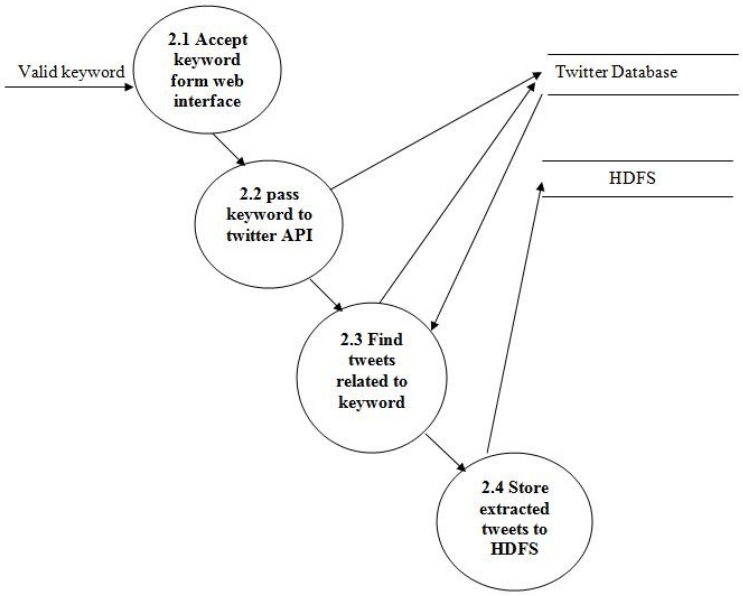


Figure 5.5: DFD Level 2 for storing to HDFS

Valid keyword is passed to web interface through process 2.1, and it is transferred to Twitter API through process 2.2. It accesses Twitter Database to find related tweets using the process 2.3. The output of process 2.3 is passed to process 2.4 to store the extracted tweets to HDFS. It uses the data store HDFS as shown in figure 5.

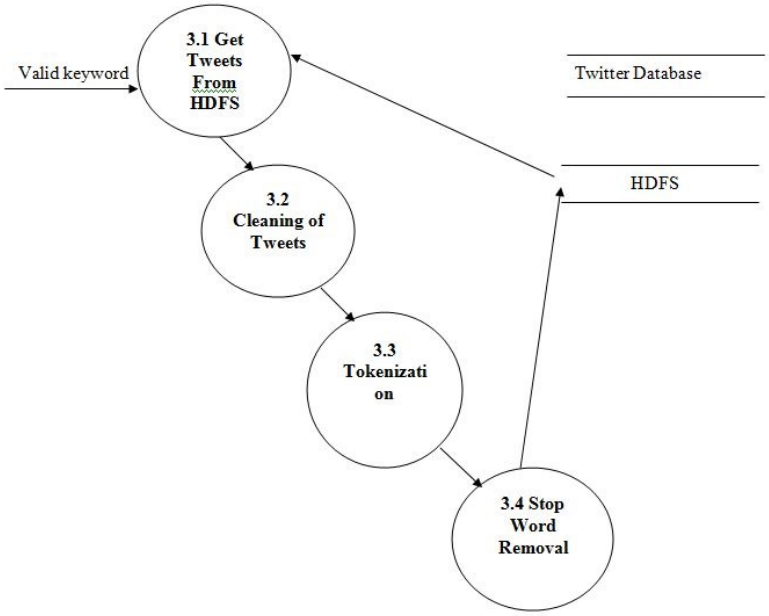


Figure 5.6: DFD Level 2 for Preprocessing

Process 3.1 gets the tweets that are stored in HDFS. It is then passed to the process 3.2 for the cleaning of tweets, followed by the process 3.3, where the tweets are tokenized. The tokenized tweets are passed to the process 3.4 for the Stop-word removal process. The data store accessed is the HDFS.

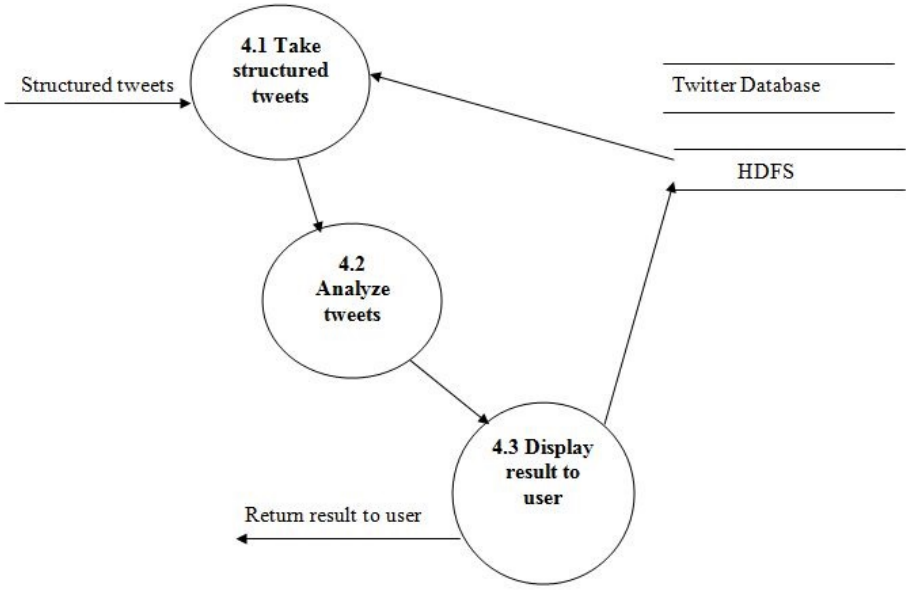


Figure 5.7: DFD Level 2 for Sentiment Analyzer

Input to the process 4.1 is the structured tweets, which are sent to the process 4.2, which analyzes the tweets. Output of process 4.2 is sent to process 4.3, which displays the result to the user. The data source used here is HDFS.

5.3 UML Diagrams

5.3.1 Use case Diagram

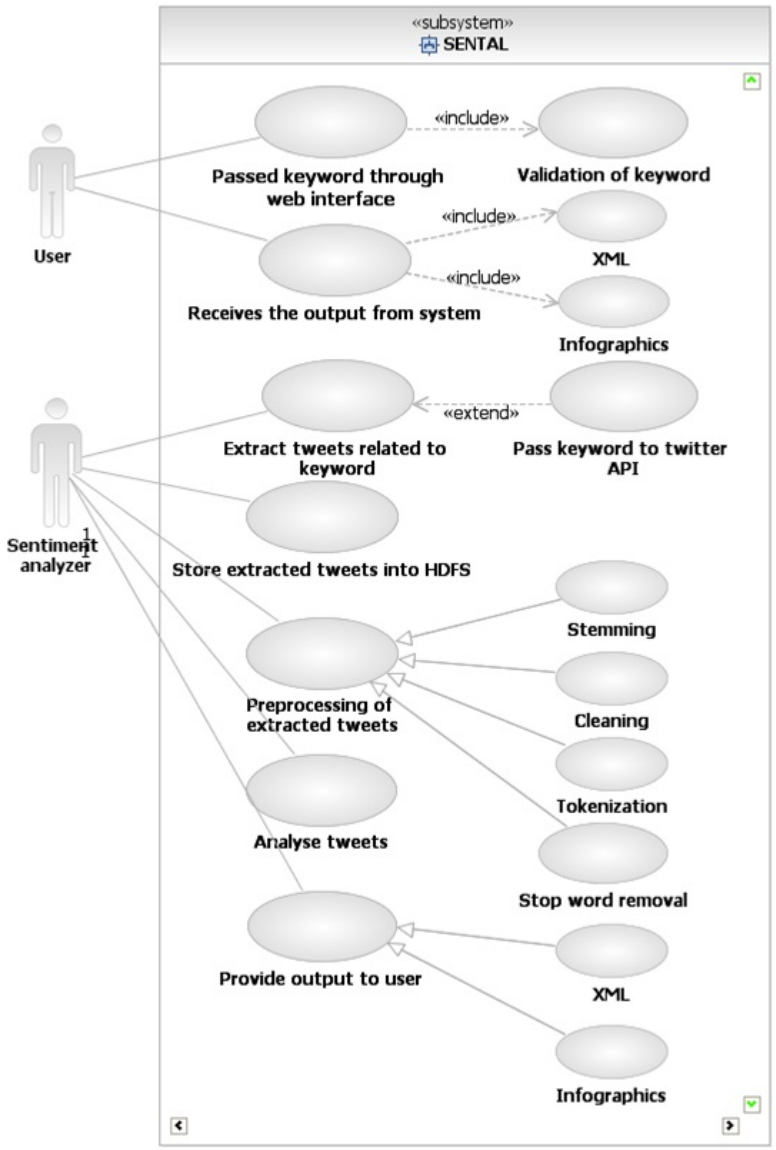


Figure 5.8: Use Case Diagram

The Use case diagram for the proposed system is shown above

The system has two main actors:

1. User 2. Sentiment Analyzer

The User and Sentiment Analyzer interact through various use cases. Some of the cases undergo generalization and specialization, and are shown by keywords include and extend respectively. User passes keyword through the designed web interface. The keyword is validated

and result is extracted through sentimental analyzer and result is passed to the user. The sentiment analyzer receives the keyword from the interface and it extracts the various tweets related to the keyword. Part of the sentiment analyzer, processes the extracted tweets, using various processes like stemming, cleaning, tokenization and stop word removal. The output is displayed in the web page in the form of info graphics or xml sheet.

5.3.2 Activity Diagram

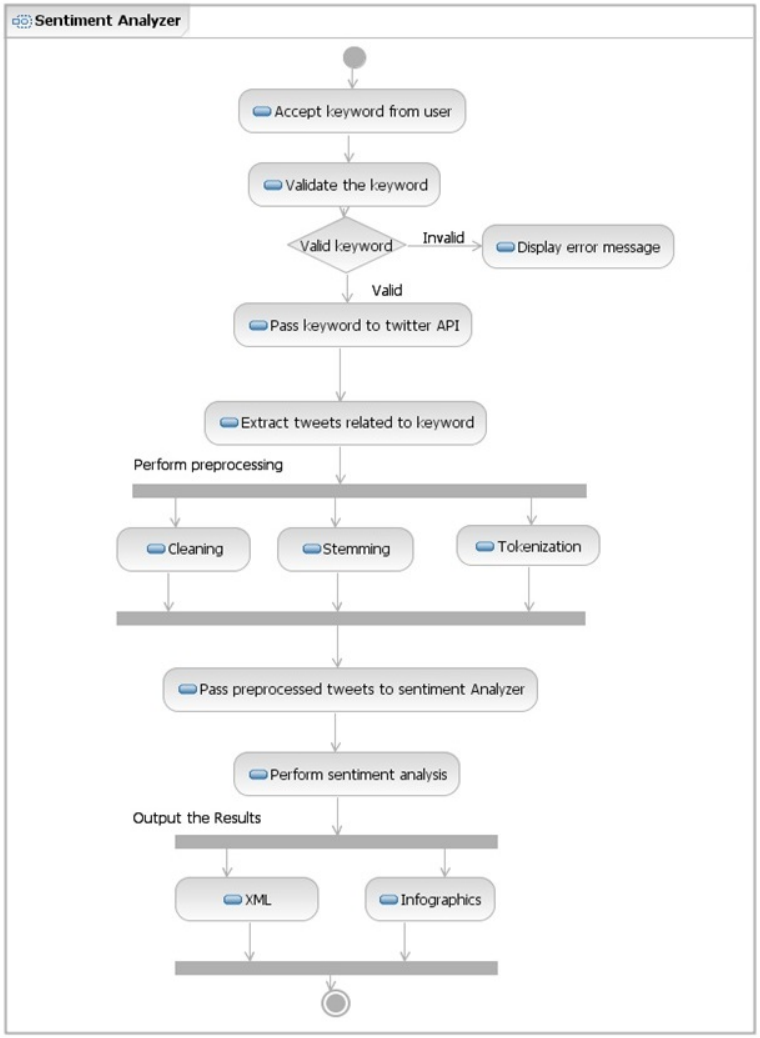


Figure 5.9: Activity Diagram

The Activity diagram for the proposed system is shown in above diagram.

The actor passes a keyword to the web application. The keyword is validated for correct and approved format and the errors are shown, if present. The web application passes the keyword to the twitter database through the sentiment analyzer. The twitter API returns the matched terms for the keyword and returns the raw data to sentiment analyzer. The sentiment analyzer

analyses the raw tweets and processes them to produce the result in form of infographics or xml sheet.

5.3.3 Sequence Diagram

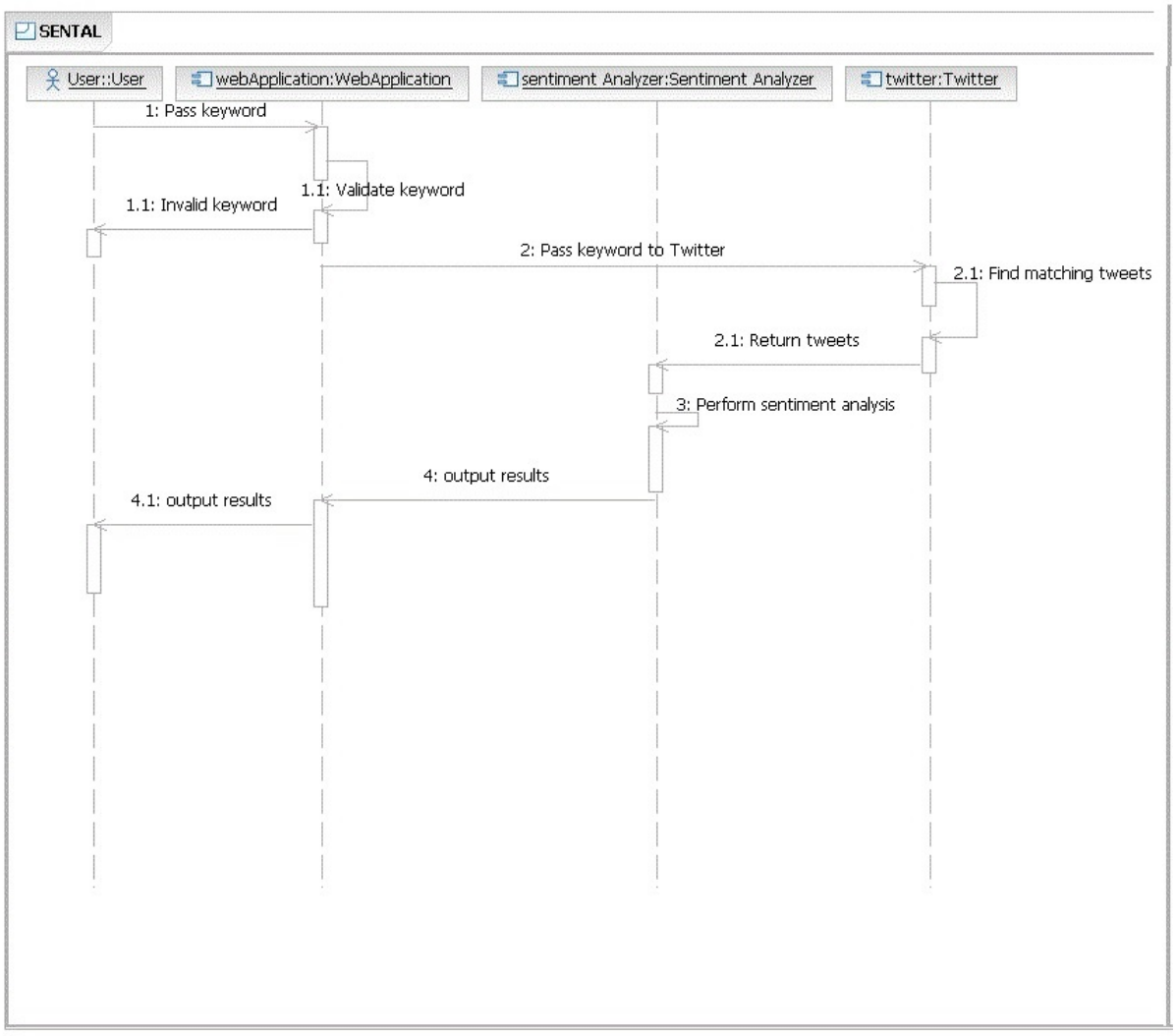


Figure 5.10: Sequence Diagram

The sequence diagram for the proposed system is as shown in figure 6 diagram.

The actor here is User. The actor passes a keyword to the web application. The keyword is validated for correct and approved format and the errors are shown, if present.

The web application passes the keyword to the twitter database through the sentiment analyzer. The twitter API returns the matched terms for the keyword and returns the raw data to sentiment analyzer. The sentiment analyzer analyses the raw tweets and processes them to produce the result in form of infographics or xml sheet.

Chapter 6

Implementation Details

6.1 Installation of Cloudera's Distributed Hadoop

Installation starts with VMWare and booted it with Cent OS, and then we installed CDH through command line.

Further installation included following softwares:

- MySQL Server
- Oracle JDK
- Flume, Hive, Oozie

After installation, we tested CDH ecosystem as given in figure below:

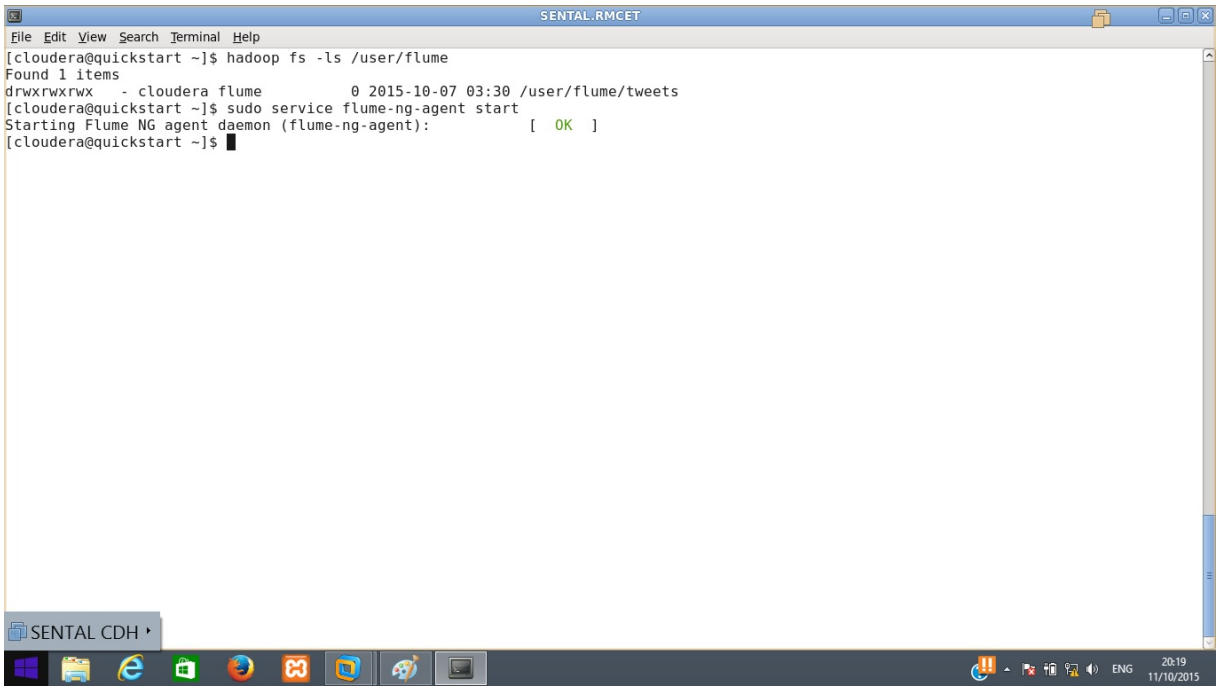


Figure 6.1: Installation of Cloudera's Distributed Hadoop

6.2 Generation of Twitter Stream API Keys

Created an application named SENTAL.rmcet in https://dev.Twitter.com/apps/ and then generated the corresponding keys.

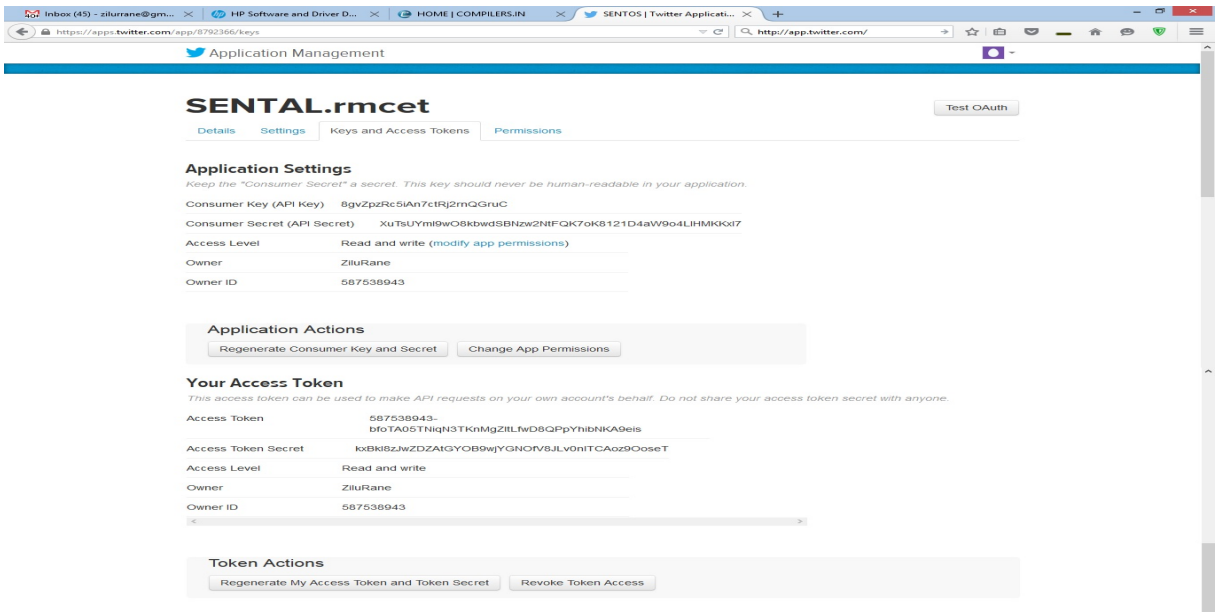


Figure 6.2: Generation of Twitter Stream API Keys

6.3 Flume Configuration

Now we edited /etc/flume.conf file which was previously empty and mentioned keys generated in above step.

Detailed script is shown in figure.

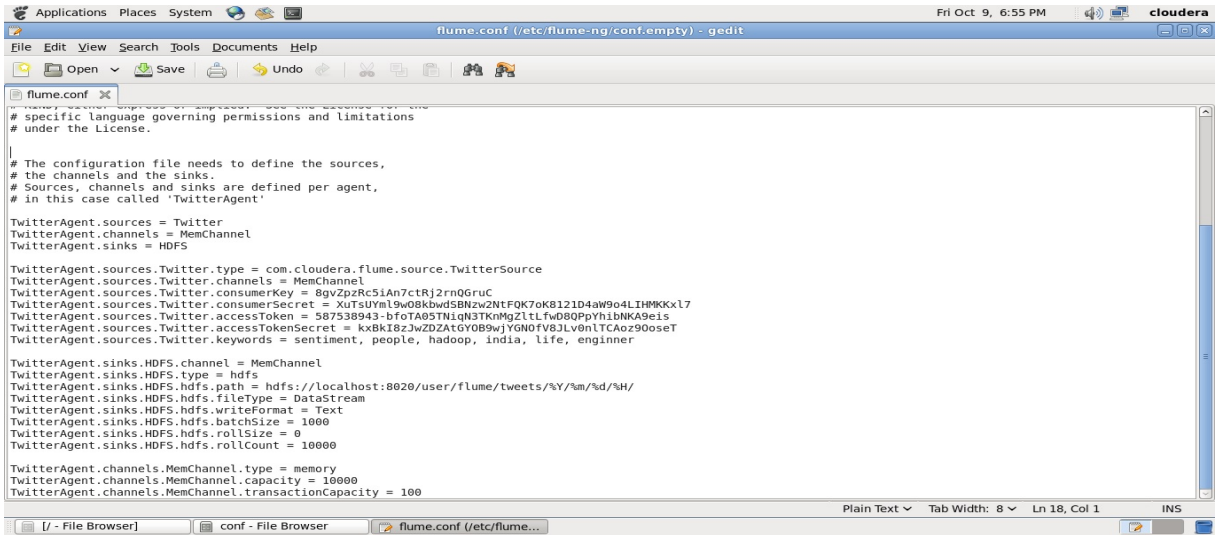


Figure 6.3: Flume Configuration

6.4 Design of SENTAL GUI

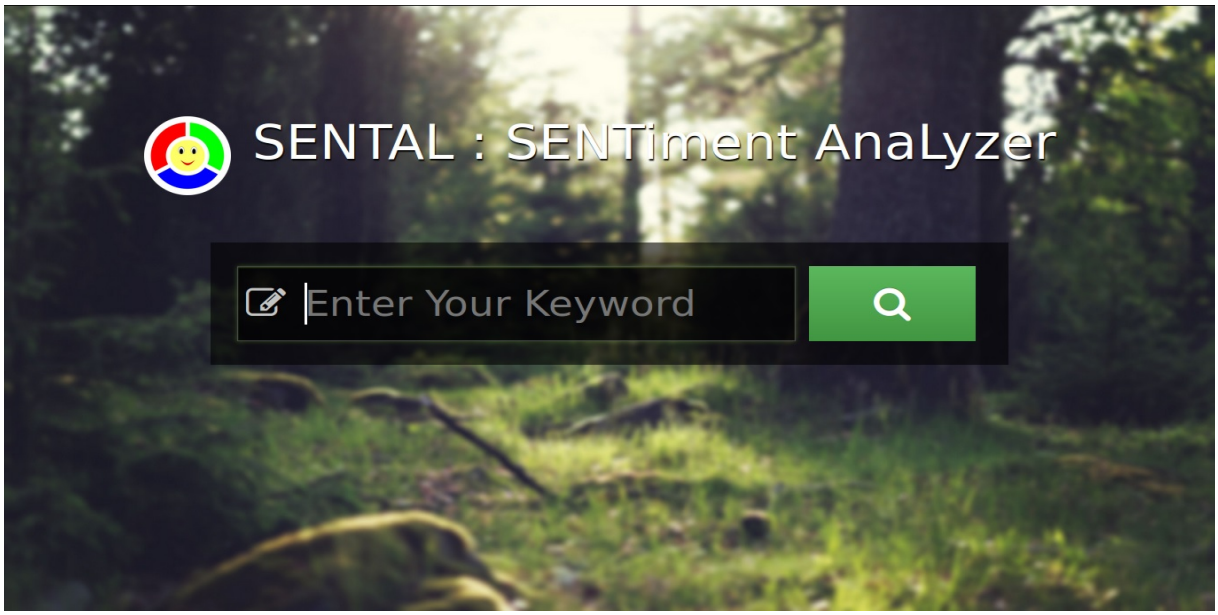


Figure 6.4: SENTAL Homepage

This is the web interface for user for entering the desired keyword for which he/she is expected to find out the sentiments of.

SENTAL output

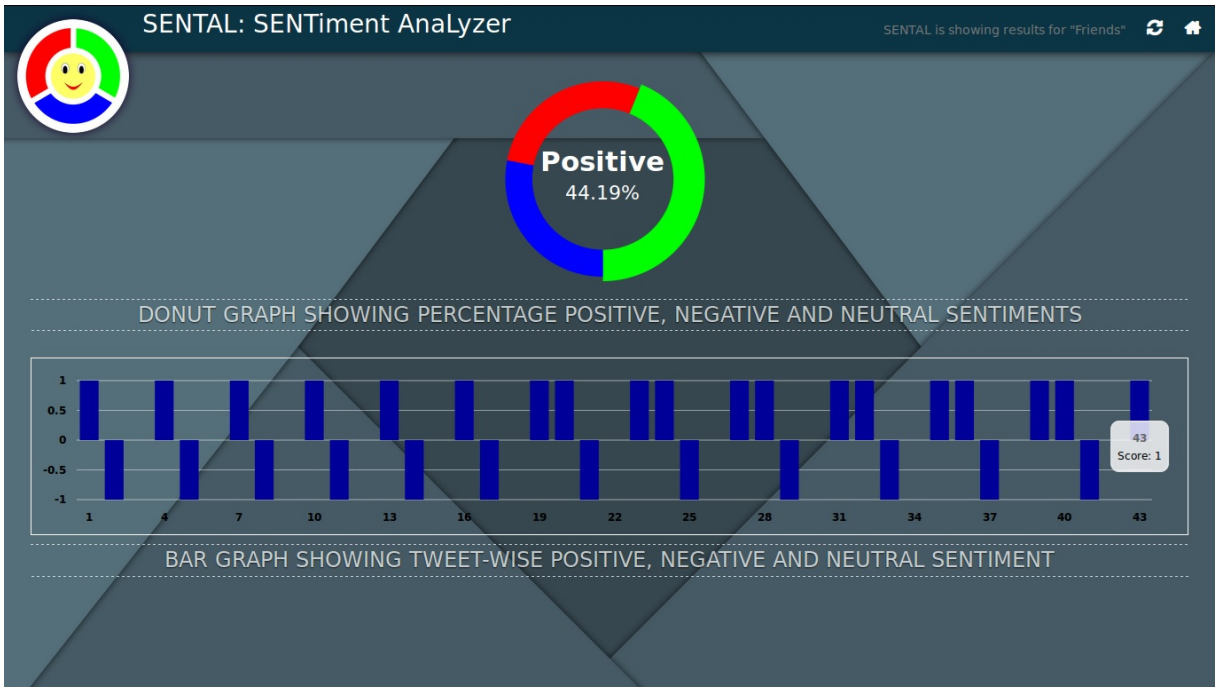


Figure 6.5: SENTAL output

This interface displays the sentiments analysed based on the keyword entered by user. The results are displayed in the form of infographics - bar charts and a pie-chart

6.5 Design of Algorithm

6.5.1 DATA

Raw Tweets

[4]This dataset is the collection of tweets which are extracted using flume. These tweets are in the unstructured data format. The dataset is full of many non-English words along with technical substitutions. This dataset needs to be prepossessed for the effective analysis. This part is done in Preprocessing module.

Dictionaries

The algorithm maintains dictionaries which are effectively the training sets for the classification purpose. The training set is backbone of the algorithm- more strong the training set, more accurate is the analysis of sentiments.

The dictionaries we've implemented are basically yaml files. It is a markup format used for building up dictionary. There are six main dictionaries built for algorithm. They are: positive.yml, negative.yml, neutral.yml, inv.yml, inc.yml and dec.yml.

6.5.2 PRE-PROCESSING

Due to the varying and unpredictable nature of language used in tweets, it is likely that preprocessing mechanism [5] could be used to standardize certain tokens of tweets. It is highly likely that most tweets contain some form of acronym, grammatical or spelling mistakes, colloquialisms and slangs; incorporated into due to the 10,000-character limit imposed by Twitter on tweets.

The quality of the data affects the results and therefore in order to improve the result, the raw data is pre-processed. It deals with the preparation that removes the repeated words and punctuations and improves the efficiency of analysis algorithms.

The pre-processing module extracts the relevant content from the tweets while leaving out the irrelevant ones. The techniques applied in this paper are used commonly in information retrieval applications specifically in sentiment analysis in micro-blogging. The collected data is passed through a series of pre-processors.

Some of the pre-processing steps that have been carried out are explained below.:

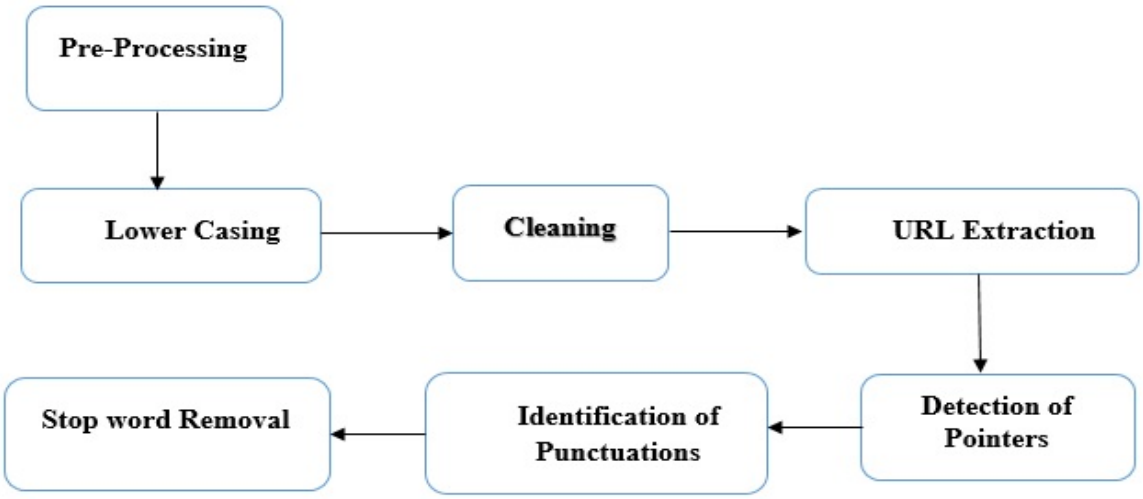


Figure 6.6: Pre-processing Module

Lower Casing

It is important to have the entire word in a consistent case when classifying texts in order

to guarantee that all tokens map to the corresponding feature irrespective of casing.[6] This is extremely important for this research work as it is very common to find irregular casing(such as ”TwITteRseNtlMeNtanAIYsiS”) in micro-blogs.

Cleaning

By using a list of cut off patterns, we omit contact addresses and formatting in order to extract only the[6] textual components, smileys,

URL Extraction

Many tweets contain URLs in order to share more content than what can be given in the limited-character post. The content in the URL might provide supplementary knowledge regarding the emotion a user trying to express, however it would be far too expensive to crawl URLs for their content. In order to trim down the feature size during training, all URLs in the training tweets have been replaced with an equivalence class URL. This could considerably reduce feature size.

Detection of Pointers

In Twitter, posts can point to other users with the use of and @ token in front of a username. And users tag tweets pertaining to a category in twitter, using #. Again, to avoid explosion of features, we abstract it to a constant symbol USER and HASHTAG. This replacement of usemames and hashtags reduce the feature size by a large margin.

Identification of Punctuations

In micro-blogging space, it is common to use excessive punctuation in order to stay away from proper grammar and to communicate emotion more easily. The punctuations can also give insight to the polarity of the message. For example, exclamation marks are used to express powerful emphasis which are usually polar messages [13]. In this step, irrelevant punctuations marks were removed by replacing PUNCT to avoid redundant feature in the training set.

Stop word Removal

Words without a deeper meaning, such as the, is, of, are named [7]stop words and can thus be removed. We use a list of stop words.

```
def cleanTweet(tweet):
      tweet = tweet.lower()
      tweet = re.sub('((www\.[^\s]+)|(https?://[^\s]+))','URL',tweet)
      tweet = re.sub('@[^\s]+',' ',tweet)
      tweet = re.sub('[\s]+', ' ', tweet)
      tweet = re.sub(r'#([^\s]+)', r'\1', tweet)
      tweet = tweet.strip('\'"')
      tweet = re.sub('[|]','',tweet)
      return tweet
```

PRE-PROCESSING MODULE

6.5.3 TOKENIZATION MODULE

We segment tweets by splitting it by spaces and punctuation marks, and form a bag of words. [7]That is each tweet is split into sentences and single words named tokens.

```
class Splitter(object):
    def __init__(self):
        self.nltk_splitter = nltk.data.load('tokenizers/punkt/english.pickle')
        self.nltk_tokenizer = nltk.tokenize.TreebankWordTokenizer()
    def split(self, text):
        sentences = self.nltk_splitter.tokenize(text)
        tokenized_sentences = [self.nltk_tokenizer.tokenize(sent) for sent in sentences]
        return tokenized_sentences
```

TOKENIZATION MODULE

6.5.4 PART-OF-SPEECH TAGGING

One of the earliest steps in the processing of natural language text is part of speech (POS) tagging. Usually this is a sentence-based process and given a sentence formed of a sequence of words, part-of-speech tagging tries to label (tag) each word with its correct part of speech.

The mechanism of classifying words into their parts of speech and labeling them accordingly is known as part-of-speech tagging, POS-tagging, or simply tagging. Parts of speech are also known as word classes or lexical categories. The collection of tags used for a particular task is known as a tag set.

A part-of-speech tagger, or POS-tagger, processes a sequence of words, and attaches a part of speech tag to each word:

```
class POSTagger(object):

    def __init__(self):
      pass

    def pos_tag(self, sentences):
      pos = [nltk.pos_tag(sentence) for sentence in
      sentences]
      #adapt format
      pos = [[(word, word, [postag]) for (word, postag) in
      sentence] for sentence in pos]
      return pos
```

PART-OF-SPEECH TAGGING MODULE

6.5.5 DICTIONARY TAGGING

Dictionary tagging is implemented by dict tagger module. The sole purpose of dictionary tagging is to lookup for the arrival of tokenised word in the predefined training sets.

Initially, it opens all the dictionaries and keep them ready for mapping. The lookup of the extracted token is then followed in every opened dictionary. As soon as the token is found in either of the dictionaries, the tagger notifies the presence of the token in corresponding dictionary. After successfully tagging, it returns tagged sentences.

Lemmatization

In computational linguistics, stemming refers to the process that reduces inflected words to their stem. This process is carried out by the lemmatiser() module. The lemmatisation results in the drawing out a derived word to it' s root form. This is an important step in NLP.

```
wnl = WordNetLemmatizer()

downcase=[i.replace(i,wnl.lemmatize(i,'a')) for i in nouns]
```

Calculation of Sentence Score

In this module, following dictionaries serve the most important role of calculating sentiment score:- inc.yml, dec.ml and inv.yml.

If a tagged token is found to be of positive sentiment, the score is added multiplicatively. On contrast, if the token is found as of negative sentiment, the score is subtracted divisionally. If word which changes the polarity of a sentence are encountered, then the token is looked up in inv.yml dictionary and the score is inverted multiplicatively.

```
def sentence_score(sentence_tokens, previous_token, acum_score):
    if not sentence_tokens:
        return acum_score
    else:
        current_token = sentence_tokens[0]
        tags = current_token[2]
        token_score = sum([value_of(tag) for tag in tags])
        if previous_token is not None:
            previous_tags = previous_token[2]
            if 'inc' in previous_tags:
                token_score *= 2.0
            elif 'dec' in previous_tags:
                token_score /= 2.0
            elif 'inv' in previous_tags:
                token_score *= -1.0
        return sentence_score(sentence_tokens[1:], current_token, acum_score + token_score)
```

SCORE CALCULETION MODULE

Storing of Result

Finally, overall sentiment score of a tweet is calculated and is stored in a separate output file. This file is an input for the Graph Rendering module, which renders an infographic format of sentiment analysis.

Consider a sample tweet to be analyzed

```
This project is just an biggest piece of work!!
```

The first step carried out is the tokenisation of the sentence.

```
[['This', 'project', 'is', 'just', 'an', 'biggest', 'piece',
'of', 'work', '!'], ['!']]
```

The tokenized words are stored in a list in python.

The second step is to convert the tokens to possible root word by lemmatizer() function:

```
[['This', 'project', 'is', 'just', 'an', u'big', 'piece', 'of',
'work', '!'], ['!']]
```

In next step, every token undergoes part of Speech tagging by POSTagger module:

```
[('This', 'This', ['DT']),
('project','project', ['NN']),
('is', 'is', ['VBZ']),
('just', 'just', ['RB']),
('an', 'an', ['DT']),
('biggest', u'big', ['JJ']),
('piece', 'piece', ['NN']),
('of', 'of', ['IN']),
('work', 'work', ['NN']),
('!', '!', ['.'])],
[('!', '!', ['.'])]]
```

Then the POSTagged tokens are looked up in dictionaries or training set to find out it's sentiment.

```
[('This', 'This', ['DT']),
('project', 'project', ['NN']),
('is', 'is', ['VBZ']),
('just', 'just', ['RB']),
('an', 'an', ['DT']),
(u'big', u'big', ['positive', 'JJ']),
('piece', 'piece', ['NN']),
('of', 'of', ['IN']),
('work', 'work', ['NN']),
('!', '!', ['.'])],
[('!', '!', ['.'])]]
```

Once the tokens are successfully classified by the algorithm, a sentiment score is calculated for whole sentence:

```
score = sentiment_score(dict_tagged_sentences)
```

The value of a score is either positive or negative or 0 Accordingly the newly arrived token is added to the corresponding dictionary.

Finally, the overall result for the score is stored to a text file name Result-Tweets.txt. This file serves as an input to the graph rendering module.

Chapter 7

Technology Used

7.1 Python

Python is a multi-paradigm programming language: object-oriented programming and structured programming are fully supported, and there are a number of language features which support functional programming and aspect-oriented programming (including by metaprogramming and by magic methods) Many other paradigms are supported using extensions, including design by contract and logic programming.

Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. An important feature of Python is dynamic name resolution (late binding), which binds method and variable names during program execution. The design of Python offers some support for functional programming in the Lisp tradition. The language has map(), reduce() and filter() functions; comprehensions for lists, dictionaries, and sets; and generator expressions. The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML

Python has a large standard library, commonly cited as one of Python's greatest strengths, providing tools suited to many tasks. This is deliberate and has been described as a "batteries included" Python philosophy. For Internet-facing applications, a large number of standard formats and protocols (such as MIME and HTTP) are supported. Modules for creating graphical user interfaces, connecting to relational databases, pseudorandom number generators, arithmetic with arbitrary precision decimals, manipulating regular expressions, and doing unit testing are also included.

Most Python implementations (including CPython) can function as a command line interpreter, for which the user enters statements sequentially and receives the results immediately (REPL). In short, Python acts as a shell. Other shells add capabilities beyond those in the basic interpreter, including IDLE and IPython. While generally following the visual style of the Python shell, they implement features like auto-completion, retention of session state, and syntax highlighting.

7.2 Hadoop

HADOOP ARCHITECTURE

Hadoop make use of HDFS for data storage purpose. Each cluster of Hadoop contains variety of nodes hence HDFS architecture is broadly divided into following three nodes,

- Name Node.
- Data Node.
- HDFS Clients(Edge Node).

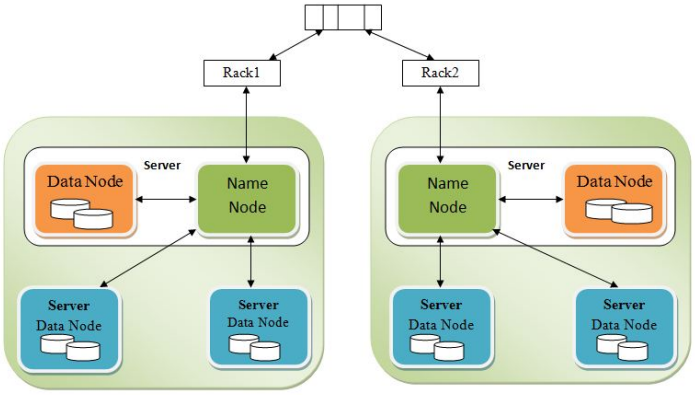


Figure 7.1: Hadoop Architecture

Name Node:

Name Node is also known as master node, which contains the information about or we can say that meta data about the all data node and there address(use to talk), free space, data they store, active data node , passive data node, task tracker, job tracker and many other configuration like replication of data.

Data Node:

In simple words, Data Node is one type of slave node in the Hadoop, which is used to save the data and there is task tracker in data node which is use to track on the ongoing job on the data node and the jobs which coming from name node.

HDFS Clients:

Hadoop Architecture, is based on HDFS, which is hadoop distributed file system. In which data is equally (ideally) distributed on each node in the hadoop system. When client want to fetch or add or modify or delete some data from hadoop, then hadoop system collect the data from each node and do the meaningful actions as per requirement.

Chapter 8

Test Cases

Table 8.1: Test Cases

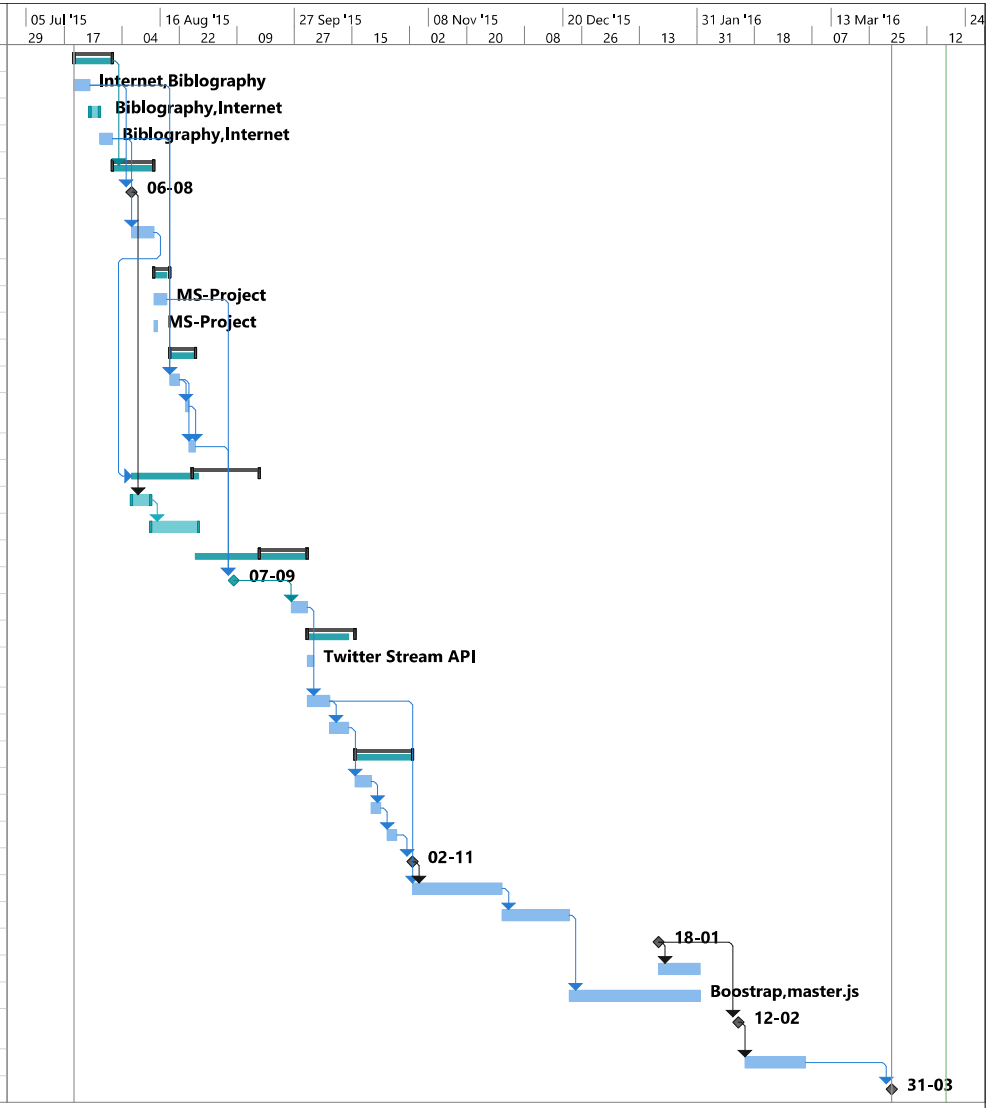
Test Cases				Test Steps			
Sr. No.	**ID**	**Test Case Name**	**Test Case Description**	**Step**	**Expected Result**	**Actual Result**	**Status**
1	TC001	Preprocessing	Passing raw unstructured tweets to Cleaning module and getting cleaned tweets	#1	Preprocessed tweets	As expected	Pass
2	TC002	Tokenization	Splitting individual tweets to tokenss	#2	Tokenized tweets	As expected	Pass
3	TC003	POStagging	Getting part of speech of each tokens	#3	POSTagged tweet	As expected	Pass
4	TC004	Dictionary Tagging	Looking up the training set	#4	Dictionary tagged tokens	As expected	Pass
5	TC005	Score Calculation	Calculate score of sentence, to decide whether it is positive, ngative or neutral	#5	Accumulated score	As expected	Pass
6	TC006	Main Result storing	Storing result of tweet in Result-tweet.txt	#6	Score get stored in file	As expected	Pass
7	TC007	Dictionary Updating	Update dictionary with new training data	#7	Updated dictionary	As expected	Pass

Note: # - Refer appendix

Chapter 9

Project Timeline

ID	Task Mode	Task Name	Duration	Start	Finish	Predecessors	Resource Names
1		**Information Gathering**	**10 days**	**Mon 20-07-15**	**Fri 31-07-15**		
2		Review of Existing Systems	5 days	Mon 20-07-15	Fri 24-07-15		Internet,Biblography
3		Gathering Documents Related to Proposed System	2 days	Sat 25-07-15	Mon 27-07-15		Biblography,Internet
4		Review of Machine Learning Algorithms	4 days	Tue 28-07-15	Fri 31-07-15		Biblography,Internet
5		**Requirement/Information Analysis**	**11 days**	**Sat 01-08-15**	**Thu 13-08-15**	**1**	
6		Comparison of Reviewed Systems and Determining Drawbacks and Advantages	5 days	Sat 01-08-15	Thu 06-08-15	2	
7		Comparison of Reviewed Machine Learning Algorithms and Selecting Best Algorithm	6 days	Fri 07-08-15	Thu 13-08-15	4	
8		**Plan Schedule**	**3 days**	**Fri 14-08-15**	**Tue 18-08-15**		
9		Project scheduling and Creating WBS	2 days	Fri 14-08-15	Mon 17-08-15		MS-Project
10		Creating Gantt Chart	1 day	Fri 14-08-15	Fri 14-08-15		MS-Project
11		**Scope Definition and Cost Estimation**	**6 days**	**Wed 19-08-15**	**Wed 26-08-15**		
12		Defining Assumptions and Limitations	3 days	Wed 19-08-15	Fri 21-08-15	2,4	
13		Defining Non-functional Properties and Listing Advantages	1 day	Mon 24-08-15	Mon 24-08-15	12	
14		Determining Financial Requirements	2 days	Tue 25-08-15	Wed 26-08-15	12,13	
15		**Analysis and Modification Algorithm**	**15 days**	**Wed 26-08-15**	**Tue 15-09-15**	**7**	
16		Studying Selected Algorithm	5 days	Fri 07-08-15	Wed 12-08-15	6	
17		Modifying According to Proposed System	11 days	Thu 13-08-15	Thu 27-08-15	16	
18		**Configuration of Tools**	**12 days**	**Wed 16-09-15**	**Wed 30-09-15**		
19		Installation of HADOOP	8 days	Thu 27-08-15	Mon 07-09-15	14,9	CDH
20		Configuration of FLUME and HIVE	4 days	Sat 26-09-15	Wed 30-09-15	19	
21		**Coding of Tweet Extraction Module**	**11 days**	**Thu 01-10-15**	**Thu 15-10-15**		FLUME
22		Creation of an Application in Twitter's developer site to generate key	2 days	Thu 01-10-15	Fri 02-10-15		Twitter Stream API
23		Configuration of FLUME	5 days	Thu 01-10-15	Wed 07-10-15	20	
24		HDFS Configuration	4 days	Thu 08-10-15	Tue 13-10-15	23	
25		**Coding of Preprocessing module**	**12 days**	**Fri 16-10-15**	**Mon 02-11-15**		**Hive,Oozie**
26		Cleaning and Tokenization	3 days	Fri 16-10-15	Tue 20-10-15	24	
27		Stop word removal	3 days	Wed 21-10-15	Fri 23-10-15	26	
28		Part-of-Speech tagging	3 days	Mon 26-10-15	Wed 28-10-15	27	
29		Stemming	3 days	Thu 29-10-15	Mon 02-11-15	28	
30		Testing of Data Extraction & Pre-Processing Module	20 days	Tue 03-11-15	Mon 30-11-15	23,29	
31		Designing GUI & Structure of Infografics	15 days	Tue 01-12-15	Mon 21-12-15	30	
32		Coding Sentiment Analyzer	20 days	Tue 22-12-15	Mon 18-01-16		Python
33		Connection of HDFS & Sentiment Analyzer	10 days	Tue 19-01-16	Sun 31-01-16	32	
34		Coding GUI & Infographics	30 days	Tue 22-12-15	Sun 31-01-16	31	Boostrap,master.js
35		Training and Improving Sentiment Analyzer Algorithm Result	10 days	Mon 01-02-16	Fri 12-02-16	32	
36		Testing of Whole System	15 days	Mon 15-02-16	Fri 04-03-16	35	
37		Documentation and Report Generation	13 days	Tue 15-03-16	Thu 31-03-16	36	MS Office



Project: sental
Date: Mon 17-08-15

Task	Project Summary	Manual Task	Start-only	Deadline			
Split	Inactive Task	Duration-only	Finish-only	Progress			
Milestone	Inactive Milestone	Manual Summary Rollup	External Tasks	Manual Progress			
Summary	Inactive Summary	Manual Summary	External Milestone				

Chapter 10

Task Distribution

Table 10.1: Task Distribution

Sr. No.	Task		Project Member		
			Zilu	Mayur	Akshay
1	Requirement Gathering	Literature Survey	30%	40%	30%
		Study of Existing Systems	30%	30%	40%
2	Requirement Analysis	Distribution of Workload	40%	30%	30%
		Defining Scope of the Project	40%	30%	30%
3	Design		40%	30%	30%
4	Coding and Implementation		30%	40%	30%
5	System Maintenance and Documentation		30%	30%	40%

Chapter 11

Conclusion and Future Work

11.1 Conclusion

Thus, in proposed system we are using a set of techniques of machine learning for sentimental analysis based on Twitter data. We are gathering the Twitter dataset through Hadoop and to analyze content using byes technique to support neutral class.

11.2 Future Work

We are going to implement following improvements in future:

1. Provision of API extension in format of XML.
2. Providing real time simulation of tweet analysis.
3. Current system is only restricted to Web platform, In future we can extent its scope to mobile platforms like Android apps.

Bibliography

[1] P.D. Turney *Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews*. [*Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*]. Philadelphia, pp. 417-424, July 2002.

[2] A. Go, R. Bhayani, and L. Huang *Twitter Sentiment Classification using Distant Supervision*. [*Stanford*]. Technical, 2009.

[3] Balakrishnan Gokulakrishnan *Opinion Mining and Sentiment Analysis on a Twitter DataStream*. [*The International Conference on Advances in ICT for Emerging Regions ICTer 2012: 182-188*].

[4] Ana C.E.S.Lima *Automatic Sentiment Analysis of Twitter Message*. [*2012 Fourth International Conference on Computational Aspects of Social Networks (CASoN)*]. 2012.

[5] K.P.Murphy *Naive Bayes classifiers*. [*Stanford*]. Technical, 2009.

[6] A. Go, R. Bhayani, and L. Huang *Twitter Sentiment Classification using Distant Supervision*. [*http://www.cs.ubc.cal-murphyk/TeachinglCS340-Fall06/readingiNB.pdf.*].

[7] A. Pak and P.Paroubek *Twitter as a Corpus for Sentiment Analysisand Opinion Mining*. [*Conference on International Language Resources and Evaluation LREC'lO*]. May 2010

Appendix

#1 : Processing Raw Tweets

```
    def cleanTweet(tweet):
        tweet = tweet.lower()
        tweet = re.sub('((www\.[^\s]+)|(https?://[^\s]+))','URL',tweet)
        tweet = re.sub('@[^\s]+',' ',tweet)
        tweet = re.sub('[\s]+', ' ', tweet)
        tweet = re.sub(r'#([^\s]+)', r'\1', tweet)
        tweet = tweet.strip('\'"')
        tweet = re.sub('[|]','',tweet)
        return tweet
```

#2 : Tokenization

```
class Splitter(object):
    def __init__(self):
        self.nltk_splitter = nltk.data.load('tokenizers/punkt/english.pickle')
        self.nltk_tokenizer = nltk.tokenize.TreebankWordTokenizer()
    def split(self, text):
        sentences = self.nltk_splitter.tokenize(text)
        tokenized_sentences = [self.nltk_tokenizer.tokenize(sent) for sent in sentences]
        return tokenized_sentences
```

#3 : Part of speech tagging

```
class POSTagger(object):
    def __init__(self):
        pass
    def pos_tag(self, sentences):
        pos = [nltk.pos_tag(sentence) for sentence in sentences]
        #adapt format
        pos = [[(word, word, [postag]) for (word, postag) in sentence] for sentence in pos]
        return pos
```

#4 : Calculation of sentence score

```
def sentence_score(sentence_tokens, previous_token, acum_score):
    if not sentence_tokens:
        return acum_score
    else:
        current_token = sentence_tokens[0]
        tags = current_token[2]
        token_score = sum([value_of(tag) for tag in tags])
        if previous_token is not None:
            previous_tags = previous_token[2]
            if 'inc' in previous_tags:
                token_score *= 2.0
            elif 'dec' in previous_tags:
                token_score /= 2.0
            elif 'inv' in previous_tags:
                token_score *= -1.0
        return sentence_score(sentence_tokens[1:], current_token, acum_score + token_score)
```

#5 : Dictionary tagging

```
class DictionaryTagger(object):
    def __init__(self, dictionary_paths):
        files = [open(path, 'r') for path in dictionary_paths]
        dictionaries = [yaml.load(dict_file) for dict_file in files]
        map(lambda x: x.close(), files)
        self.dictionary = {}
        self.max_key_size = 0
        for curr_dict in dictionaries:
            for key in curr_dict:
                if key in self.dictionary:
                    self.dictionary[key].extend(curr_dict[key])
                    print key
                else:
                    self.dictionary[key] = curr_dict[key]
                    self.max_key_size = max(self.max_key_size, len(key))
    def tag(self, postagged_sentences):
        return [self.tag_sentence(sentence) for sentence in postagged_sentences]
    def tag_sentence(self, sentence, tag_with_lemmas=False):
        tag_sentence = []
        N = len(sentence)
        if self.max_key_size == 0:
            self.max_key_size = N
        i = 0



        while (i < N):
            j = min(i + self.max_key_size, N) #avoid overflow
            tagged = False
            while (j > i):
                expression_form = ' '.join([word[0] for word in sentence[i:j]]).lower()
                expression_lemma = ' '.join([word[1] for word in sentence[i:j]]).lower()
                if tag_with_lemmas:
                    literal = expression_lemma
                else:
                    literal = expression_form
                if literal in self.dictionary:
                    #self.logger.debug("found: %s" % literal)
                    is_single_token = j - i == 1
                    original_position = i
                    i = j
                    taggings = [tag for tag in self.dictionary[literal]]
                    tagged_expression = (expression_form, expression_lemma, taggings)
                    if is_single_token:
                        original_token_tagging = sentence[original_position][2]
                        tagged_expression[2].extend(original_token_tagging)
                    tag_sentence.append(tagged_expression)
                    tagged = True
                else:
                    j = j - 1
            if not tagged:
                tag_sentence.append(sentence[i])
                i += 1
        return tag_sentence
```

#6 : Main result storing

```
def sentiment_score(review):
    return sum([sentence_score(sentence, None, 0.0) for sentence in review])

def addResult(score):
    #record the score and save it to Result file...
    x = "Tweets"
    open('Result-'+x+'.txt','a').write(str(score)+'\n')
```

#7 : Dictionary updating

```
def findPositiveTag(text):
    #print(pos_tagged_sentences)
    tokens = nltk.word_tokenize(text)
    tagged = nltk.pos_tag(tokens)
    nouns = [word for word,pos in tagged if(pos =='JJ' or pos == 'JJR' or pos == 'RB')]
    downcase=[i.replace(i,wnl.lemmatize(i,'a')) for i in nouns]
    for i in range(0,len(downcase)):
        print(downcase[i])
        if(downcase[i] in open("dicts/positive.yml","r").read()):
            continue
        if(downcase[i] in open("dicts/negative.yml","r").read()):
            continue
        if(downcase[i] in open("dicts/neutral.yml","r").read()):
            continue
        else:
            open("dicts/positive.yml","a").write(str('\n'+downcase[i])+": [positive]")
    #store NN and JJ part
```

Use of Hadoop Framework for Web Based Sentiment Analysis

Akshay R. Kalambate[1] Mayur R. Mane[2] Zilu Rane[3] Prof. Pralhad S. Gamare[4]
[1,2,3,4]Department of Computer Engineering
[1,2,3,4]Mumbai University, RMCET Ambav, Maharashtra 415804, India

Abstract—Current era is of social networking sites, petabytes of data is generated daily on web. Millions of people are posting their likes, dislikes, comments daily on social networking sites. In this system, we are proposing a model that will extract the sentiment from a famous micro blogging site, Twitter, where users post their opinions for everything. Twitter is an online web application which contains lots of data that can be a structured or semi-structured or un-structured data format. We can collect the data from the twitter by using Apache Hadoop(BIG DATA) eco-system using online streaming tool Flume. There are different types of analysis that can be done on the collected data. So here we are taking sentiment analysis, for this we are choose to use Hive and its queries to give the sentiment data based up on the groups that we have defined in the HQL (Hive Query Language).Proposed model uses modified version of Naïve Bayes machine learning algorithm. Our modifications introduce neutral class by eliminating class conditional independence assumption of Naïve Bayes classifier by considering probability intersection between positive and negative classes. Algorithm results are improved by reducing words in tweet to their root form through mechanism of pre-processing before passing them to sentiment analyser. Hence, proposed system classifies tweets as positive, negative or neutral with respect to a query term. This system may prove useful for the enterprises who want to know the feedback about their product brands or the customers who want to improve their productivity and this system may also can be beneficial for election exit polls.

Key words: Hadoop Framework, Web Based Sentiment Analysis

I. INTRODUCTION

The current trend of social networking in the World Wide Web is observed greatly; especially of micro-blogging. Since there is no limit to the range of information carried by tweets and texts, often these short messages are used to share opinions that people use, about what is going on in the world around them.

In social media monitoring, Sentiment analysis is extremely useful, as it allows us to gain an overview of the wider public opinion. The different applications of sentiment analysis are wider and powerful. The ability to extract valuable information from social data is a practice that is being widely adopted by organisations, across the world. The Obama administration used sentiment analysis, to study public opinion to policy announcements and campaign messages ahead of 2012 presidential election.

In short, at Present situation peoples are expressing their thoughts through some online applications like Facebook, Twitter, WhatsApp, etc. As we concerned with Twitter, in Twitter more than 1TB of text data is generating every week in the form of tweets posted by all world peoples. But, analysing all this Tweets is very difficult as these huge data that are going to be generated day by day. This problem is taking now and can be solved by using concept of BIG DATA[1], that is Apache Hadoop ecosystem.

II. HADOOP ARCHITECTURE

Hadoop make use of HDFS for data storage purpose. Each cluster of Hadoop consists of different nodes. Hence, HDFS architecture is broadly divided into following three nodes,

- Name Node.
- Data Node.
- HDFS Clients(Edge Node).

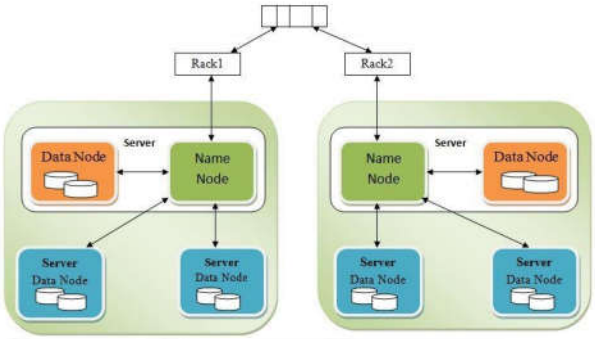


Fig. 1: Architecture of Hadoop

A. Name Node:

Name Node is also known as master node, which contains the information about or we can say that meta data about the all data node and there address(use to talk) and many other configuration like replication of data.

B. Data Node:

In simple words, Data Node is one type of slave node in the Hadoop, which is used to save the data. There is a task tracker in data node. It is used to track the ongoing jobs on the data node and the jobs which coming from name node.

C. HDFS Clients:

Hadoop Architecture is based on HDFS- Hadoop Distributed File System. The data is equally (ideally) distributed on each node in the Hadoop system. When client want to fetch or add or modify or delete some data from Hadoop, then Hadoop system collect the data from each node and do the meaningful actions as per requirement.

1) Advantages:

- Distributed data and computations, and HDFS store large amount of information.
- Simple programming model. So, sentiment analyser can be implemented easily.
- Quick recovery from system failures.
- Once data written in HDFS can be read several times, increases redundant tweets extraction.

III. LITERATURE REVIEW

In recent years a lot of work has been done in the field of 'Sentiment Analysis' by number of researchers. Work in this field started since the beginning of current century. In its

early stage, it was intended for binary classification, which assigns opinions or reviews to bipolar classes such as positive or negative.

Go et al[2] started one of the early researches in this area, where the authors try a novel approach to automatically classify sentiments in tweets. They have used distant learning methods. They had classified the tweets ending with positive emoticons, one such like :-), as positive; and tweets ending with negative emoticons such as :-(, as negative. But the system uses unigram approach which fails during determination of neutral sentiments.

There are various approaches to retrieve data from Twitter dataset. Traditional approach includes writing a program in suitable languages, to get data from Twitter database. The obtained unstructured data is to be filtered to get desired structured data, again by using programs. This is tedious way and mostly time and resource inefficient. The data to be processed is then stored in RDBMS, which has limitations for creating tables and accessing it.

Paper[3] mentions the future work of using Oozie to perform sentiment analysis task with certain time span and result visualisation task.

IV. PROPOSED SYSTEM

The proposed system models the machine learning approach for sentimental analysis of Twitter feeds using Hadoop software framework. It improves accuracy of Naïve Bayes by adding Neutral class and by probability intersection of positive and negative classes.

A. System Architecture

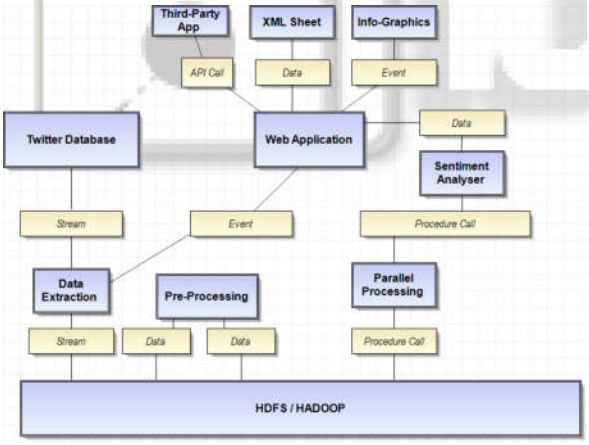


Fig. 2: Architecture of proposed system

B. Outline of System

1) Validate the Keyword

System uses Web interface to get keyword from user and validate the keyword so, only english keyword can be passed to tweet extraction module.

2) Tweets Extraction

Pass given keyword to Twitter API that find tweets related to keyword and return back to extraction module in JSON format. All above task will be done through flume that will store this Tweet Set into HDFS.

3) Pre-processing

The quality of the data affects the results and therefore in order to improve the quality, the raw data is pre-processed. It deals with the preparation that removes the repeated words and punctuations and improves the efficiency of analysis algorithms.

- Cleaning: By using a list of cut off patterns, we omit contact addresses and formatting in order to extract only the textual components, smileys.
- Tokenisation: Each tweet is split into sentences and single words named tokens.
- Stop word removal: Words without a deeper meaning, such as the, is, of, are named stop words and can thus be removed. We use a list of stop words.
- Part-of-Speech Tagging: It involves identification of verb, adverb, nouns from the sentence.
- Stemming: In computational linguistics, stemming refers to the process that reduces inflected words to their stem.

4) Parallel Processing

Structured tweets are now passed to sentiment analyser module. Oozie will take care that sufficient tweets are ready to continue task with sentiment analyser.

5) Sentiment Analyser

Now, sentiment analyser module can analyse that tweets by using modified Naïve-Bayes classifier. We will use special scoring models for smileys and normal text. Hadoop supports HiveQL to query preprocessed tweets.

C. Naïve Bayes Classifier

The Naive Bayes classifier is based on the Bayes' theorem. It is a probabilistic model. It calculates the probability of a tweet belonging to a specific class such as positive or negative. This assumes that all the features are conditionally independent. Probabilities were calculated using formula as,

$$P_{NB}(c|d) = \frac{(P(c)\sum_{i=1}^{m} P(f|c)^{n_i(d)})}{P(d)}$$

Here, class c is assigned to tweet *d*, where, *f* represents a feature and $n_i(d)$ represents the count of feature f_i found in tweet *d*. There are a total of m features. Parameters P(c) and P (f|c) are obtained through maximum likelihood estimates which are incremented by one for smoothing. Pre-processed data along with extracted feature is provided as input for training the classifier using naïve Bayes. Once the training is complete, during classification it provides the polarity of the sentiments.

1) Output

Analysed output will be in the form of XML sheet. Proposed system also makes use of infographics to visualise the output.

V. CONCLUSION

Data and networks are becoming trends. This project shows how we can use Petabytes of data that generated daily to improve our life standard. The proposed system designs an API that can be used in Prediction of Market Trends, Infotainment, Stock markets and lots more.

REFERENCES

[1] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, and A. H. Byers, "Big Data: The Next Frontier For Innovation, Competition, And Productivity", May 2011.

[2] A. Go, R. Bhayani, and L. Huang, -Twitter Sentiment Classification using Distant Supervision, Stanford, Technical, 2009.
[3] "Effective Sentiment Analysis on Twitter Data using: Apache Flume and Hive"
[4] International Journal of Innovative Science, Engineering & Technology, Vol. 1 Issue 8, October 2014.By, Penchalaiah.C, Murali.G,

International Journal of Emerging Trends & Technology in Computer Science

print this page

A Motivation for Recent Innovation & Research

ISSN 2278-6856

www.ijettcs.org

ACCEPTANCE LETTER

Date:11-Apr-2016

To.

Author(s) Name:Mr. Mane Mayur R., Mr. Kalambate Akshay R., Mr. Rane Zilu Ramkrishna, Prof. Gamare P. S.
Corresponding Author E-Mail Id :akshaykalambate15@gmail.com

Subject: Acceptance Notification for your paper (Paper id: IJETTCS-2016-04-07-50)

Dear Author(s),
We are pleased to inform you that your paper entitled :Machine Learning Algorithm For Sentimental Analysis of Twitter Feeds and Paper Id :IJETTCS-2016-04-07-50 has been ACCEPTED for publication in International Journal of Emerging Trends & Technology in Computer Science (IJETTCS), Volume 5, Issue 2, March - April 2016 .

You are requested to send soft copy of the final Camera Ready paper in MS-WORD format, scan copy of filled the copyright form in own handwriting,scan copy of filled the Publication Fee form in own handwriting along with slip of payment details send to editor@ijettcs.org.

You have to complete all formalities on or before 16-April-2016 failing to which your paper may be subjected to rejection. Your prompt response will be appreciated. Thank you again for your submission to IJETTCS.

Editor in Chief,
International Journal of Emerging Trends & Technology in Computer Science
ISSN 2278-6856
Website: www.ijettcs.org
Email: editor@ijettcs.org

ACKNOWLEDGEMENT

We would like to express our sincere gratitude towards our guide, Prof. P. S. Gamare, for the help, guidance and encouragement, he provided during the B. E. Project. This work would have not been possible without his valuable time, patience and motivation. We would like to thank him for making our stint thoroughly pleasant and enriching. It was great learning and an honor being his students. We are deeply indebted to Prof. L. S. Naik (Head of Department) and Prof. P. S. Gamare (Project Coordinator) and the entire team in the Computer Department. They supported us with scientific guidance, advice and encouragement, they were always helpful and enthusiastic and this inspired us in our work. We take the privilege to express our sincere thanks to Dr. M. M. Bhagwat, our Principal for providing the encouragement and much support throughout our work.

Mr. Rane Zilu Ramkrishna

Mr. Mane Mayur Ravindra

Mr. Kalambate Akshay Rajendra

AUTHORS

Authors from left Mr. Zilu Rane, Prof. Pralhad Gamare, Mr. Akshay Kalambate and Mr. Mayur Mane

1. **Mr. Zilu Ramkrishna Rane (zilurrane@gmail.com)**
2. **Mr. Mayur Ravindra Mane (mayoorm909@gmail.com)**
3. **Mr. Akshay Rajendra Kalambate (akshaykalambate15@gmail.com)**
4. **Prof. Pralhad S. Gamare (pralhad.gamare81@gmail.com)**