Comput Syst Sci & Eng (2018) 3: 187–202

International Journal of
Computer Systems
Science & Engineering

Sentiment Analysis System in Big Data Environment

Wint Nyein Chan[1] **and Thandar Thein**[2]

[1] *University of Computer Studies, Yangon*
[2] *University of Computer Studies, Maubin*
E-mail: wintnyeinchan2012@gmail.com; thandartheinn@gmail.com

Nowadays, Big Data, a large volume of both structured and unstructured data, is generated from Social Media. Social Media are powerful marketing tools and social big data can offer the business insights. The major challenge facing social big data is attaining efficient techniques to collect a large volume of social data and extract insights from the huge amount of collected data. Sentiment Analysis of social big data can provide business insights by extracting the public opinions. The traditional analytic platforms need to be scaled up for analyzing a large volume of social big data. Social data are by nature shorter and generally not constructed with proper grammatical rules and hence difficult to achieve high reliable result in Sentiment Analysis. Acquiring effective training data is a challenge, although learning based approaches are good for sentiment classification. Manual Labeling for training data is time and labor consuming. In this paper, Sentiment Analysis system on Big Data Analytics platform is proposed to provide valuable information by analyzing large scale social data in an efficient and timely manner since they have been implemented using a MapReduce framework and a Hadoop distributed storage (HDFS). The proposed Sentiment Analysis system consists of four modules: data collection, data cleaning and preprocessing, class labeling and sentiment classification. The system enables high-level performance of sentiment classification while taking advantage of combining lexicon-based classifier's effortless setup process and learning based classifier. Twitter stream data is used for system evaluation as the Twitter is widespread Social Media and a good source of information in the sense of snapshots of moods and feelings as well as up-to-date events. The evaluation results show that this system achieve a promising accuracy by 84.2%. Moreover, this system is able to scale up to analyze the large scale data by decreasing the processing time when adding more nodes in the cluster

Keywords: Hadoop, MapReduce, Big data, Sentiment Analysis, Lexicon, Machine learning

1. INTRODUCTION

As the rapid growth of the Internet and online activity, many services such as blogging, podcasting, social networking and bookmarking are popular. These services allow users to create and share information within open and closed communities and contribute to the volumes of the data. According to IBM reports everyday "2.5 quintillion bytes of data" is created [19] and data are increasing each year. In Social Networking, Twitter has 320 million monthly active users and they posts 500 million tweets every day; Facebook has 936 million daily and 1,440 million monthly active users as of December, 2015. These factors are reasons of a rise of Big Data [9]. Big data is characterized by the volume, velocity, veracity, variety, value and volatility of data.

At the age of Big Data, data is captured from different sources, such as mobile devices and web browsers, and it is stored in various data formats. The traditional storage and analytics platform cannot handle the different sources and different formats of the structured and unstructured data. Big Data Analytics has become popular for analyzing and managing large volume of structured and unstructured data [15]. Hadoop is a good platform for Big Data Analytics as it provides scalability, cost-effective, flexible, fast, secure and authentication, parallel processing, Availability and resilient nature. It is an open-source software framework comprises of two parts: storage part and processing part. The storage part is called the Hadoop Distributed File System (HDFS) and the processing part is called MapReduce.

Sentiment Analysis is one of the main agenda in big data that focus on various ways to analyze big data to identify patterns and relationships, make informed predictions, deliver actionable intelligence and gain business insight from this steady influx of information [18]. Sentiment Analysis is the process of using text analytics to mine various sources of data for opinions. There are two main approaches to the problem of Sentiment Analysis:

lexical approach and machine learning approach. In the lexical approach, the definition of sentiment is based on the analysis of individual words and/or phrases; emotional dictionaries are often used: emotional lexical items from the dictionary are searched in the text, their sentiment weights are calculated, and some aggregated weight function is applied. This technique is governed by the use of a dictionary consisting pretagged lexicons. The classification of a text depends on the total score it achieves. In the machine learning approach the task of Sentiment Analysis is regarded as a common problem of text classification [11] and it can be solved by training the classifier on a labeled text collection. The machine learning approach applicable to Sentiment Analysis mostly belongs to supervised classification. A number of machine learning techniques have been adopted to classify the reviews.

In recent years, Social Media have become very popular and many researchers have increasingly interested in studies of Sentiment Analysis for social data. It can help the organization to determine marketing strategies by providing public opinions. Twitter is one of the popular social media data platforms, which combines features of blogs and social network services. Twitter was established in 2006 and experienced rapid growth of users in the first years [22]. Twitter is a good source of information in the sense of snapshots of moods and feelings as well as up-to-date events and current situation commenting. In addition, Twitter provides easy access of data using search API, stream API and Firehose [6].

In this paper, Sentiment Analysis system on Big Data Analytics platform is proposed to analyze large scale social data. Data collection, data cleaning and preprocessing, sentiment classification are executed at four layers. These four layers are Data Ingestion Layer, Storage Layer, Processing Layer, and Analytics Layer. In Data Ingestion Layer, Apache Flume is used to collect tweet stream data and the collected data is ingested to HDFS through the memory channel. HDFS is located in Storage Layer. In Processing Layer, MapReduce is used for parallel processing. Data cleaning and preprocessing, class labeling and sentiment classification are implemented in Analytics Layer. All of the processes from Analytics Layer are executed in distributed manner by using HDFS and MapReduce. The sentiment classification is implemented by combining lexicon and machine learning based classifier. Instead of manually labeling the class, lexicon based method (SentiStrength) is used for labeling the class. For machine learning based approach, Mahout naïve Bayes classifier is used for providing scalable sentiment classification.

2. RELATED WORK

Due to the recent growth of data available in the World Wide Web, especially of those that reflect people's opinions, experiences and feelings, there had been increasingly interest in studies of Sentiment Analysis for large datasets.

Hadoop MapReduce framework has been used to perform the huge set of data in an efficient and timely manner. The authors [23] proposed distributed system for Sentiment Analysis of large scale data by using a MapReduce framework and a distributed database model. The system involved two components: lexicon builder [13] and sentiment classifier. They used the sentiment lexicon together with machine learning technique in order to solve the misclassification error of lexicon based classifier. Online Logistic Regression from Mahout machine learning library was used for learning based approach. The system was implemented by using existing twitter datasets. The experiment results showed that their lexicon and learning based classifier could obtain higher accuracy than the lexicon-based classifier which only relies on searching for sentiment words/phrases. For scalability, the evaluation results showed that the running time of the proposed system with different volumes of data decreases when adding more machines into the cluster. The authors [5] performed sentiment mining using a Naive Bayes classifier (NBC) to evaluate the scalability of NBC in large data sets. To achieve fine-grain control of the analysis procedure, they implemented the NBC on top of Hadoop framework with additional four modules: the work flow controller (WFC), the data parser, the user terminal and the result collector. The two datasets, which had already been labeled the class, were used for their experiments. They evaluated the scalability of NBC in a large data set and the result was encouraging in that the accuracy of NBC was improved and approaches 82% when the dataset size increases. They have demonstrated that NBC was able to scale up to analyze the sentiment of millions movie reviews with increasing throughput. N. Nodarakis et al. [17] developed a novel method to harvest the sentiment knowledge from large scale data sets by using Hadoop MapReduce framework. K Nearest Neighbour (KNN) classification algorithm was used to perform the classification procedure of diverse sentiment types in a parallel and distributed manner. Bloom filters were utilized to compact the storage size of intermediate data and boost the performance of the proposed algorithm. The Twitter Search API was used to collect the data. Since the hashtags and emoticons had been defined as the sentiment label, no more methods were required for labeling the class. To evaluate the performance of the proposed system, two experimental settings: the multi-class classification and the binary classification settings were utilized. The results showed that binary classification with Bloom filters confirmed the usefulness of their approach and larger values of k provided a great impulse in the performance of the algorithm when not using Bloom filters. They provided the scalability of the algorithm for their system almost linearly as the data size increases in all cases.

Apache Spark, one of the large scale data processing frameworks, can offer real time monitoring and analysis. A. Assiri et al. [1] described a distributed solution using big data techniques (Spark, Flume) to process the real-time Sentiment Analysis using only lexicon based approach. The flume was used for listening Twitter stream data with certain hashtags. When the new batch of tweet arrives at Flume sink, spark streaming consumes the data and loads into memory as RDD objects spread across multiple nodes based on the cluster size. Two types of real data set [2, 20] are used to implement the proposed solution. In order to test the performance of the lexicon-based algorithm, they implemented it in two ways: One way in Java and the second way in Spark. The results showed that the two implementing ways of lexicon-based algorithm achieved the same accuracy. But significant performance improvements in running time of lexicon based algorithm in Spark compared to the implementation of the lexicon based algorithm in Java. A. Baltas et al.

[3] introduced Sentiment Analysis tool that analyzes microblogging messages regarding their sentiment and it implemented on Apache Spark, an open-source framework for programming with Big Data. Some pre-processing steps were performed in achieving better results in Sentiment Analysis. The Sentiment Analysis tool is based on machine learning methodologies alongside with Natural Language Processing techniques and utilizes Apache Spark's machine learning library, MLlib. The three algorithms in MLlib utilized are Naive Bayes, Logistic Regression and Decision Trees. Binary and Ternary Classification was experimented with different pre-classified tweets data sets. On the Binary Classification case, they focused on the way that the dataset size affects the results, while on the Ternary Classification case, the focus is given to the impact of the different features of the feature vector given as an input to the classifier. For the binary classification problem, they observed that naïve Bayes classifier performs better than Logistic Regression and Decision Trees. According to the experimental results of binary classification problem, it is also obvious that the dataset size plays a rather significant role for Naive Bayes. The ternary classification results show that Naive Bayes outperforms the other two algorithms. To experiment with different clusters and evaluate Spark's performance in regards to time and scalability are described as the future works.

Researchers had made a few experiments to demonstrate the difference of their techniques and contributions for improving the performance of sentiment classification. In [7], the authors presented the pre-processing techniques can improve sentiment classification effectiveness. They experimentally compared 15 commonly used pre-processing techniques on two Twitter datasets: SS-Twitter dataset and SemEval dataset. Each technique was evaluated in three representative machine learning algorithms, namely, Linear SVC, Bernoulli naïve Bayes, and Logistic Regression. According to the results, techniques like stemming, removing numbers, and replacing elongated words improve accuracy, while others like removing punctuation do not. The paper was only focused on the pre-processing techniques of Twitter Sentiment Analysis and described evaluation results of each technique on accuracy. The authors [14] implemented a new entity level Sentiment Analysis method to effectively extract the public opinion from social data by combining lexicon-based and learning based approach. Only Lexicon based approach for entity-level Sentiment Analysis method gave high precision, but low Recall. To improve recall, the results of lexicon based approach were used as the training examples of the learning based approach. Five diverse of Twitter data sets obtained from the Twitter API were used for evaluating the method. Support Vector Machines (SVM) was used for learning based approach. Experimental results showed that the proposed method was highly effective and promising by improving the recall and the F-score compared with the state-of-the-art baselines.

In this work, Hadoop MapReduce is used for being able to manage large scale social data in a distributed manner. A large volume of Twitter stream data is used for execution and the data are collected by Apache Flume. As the Flume ingest the raw Twitter stream data, data cleaning need to be performed. Preprocessing is executed to improve the classification accuracy. Lexicon and learning based classifier are combined to achieve high level performance of sentiment classification. To acquire effective training data for learning based classification, SentiStrength lexicon based classifier is used for class labeling. Mahout naïve Bayes classifier is used to provide scalable learning based sentiment classification.

3. IMPLEMENTING BIG DATA ANALYTICS PLATFORM

In the proposed Sentiment Analysis system, Big Data Analytics Platform is developed to scale up the traditional analytics platform for analyzing large scale social data by using Apache Flume, HDFS, MapReduce and Mahout machine learning library [4]. Sentiment Analysis is implemented on Big Data Analytics platform and high level architecture of the proposed Sentiment Analysis system is illustrated in Figure 1. The proposed system consists of four layers: Storage Layer, Processing Layer, Data Ingestion Layer and Analytics Layer. The detail descriptions of each layer are presented the following subsections.

3.1 Data Ingestion Layer

In this layer, tweet stream data are collected and the collected data are ingested to HDFS through the memory channel by using Apache Flume. Apache Flume is a distributed, reliable, and available service for efficiently capturing, aggregating, and moving large amounts of log data. It has a simple and flexible architecture based on streaming data flows. It is robust and fault tolerant with tunable reliability mechanisms and many failover and recovery mechanisms.

3.2 Storage Layer

In Storage Layer, HDFS is used to provide scalable and reliable data storage. HDFS serves master/slave architecture and single NameNode serve as a master server. Name Node executes file system namespace operations like opening, closing, and renaming files and directories. It also determines the mapping of blocks to DataNodes. DataNodes is used to store the actual data in HDFS. The input file is split into one or more blocks and these blocks are stored in a set of DataNodes. Each block size is 64 MB. DataNodes are responsible for serving read and write requests from the clients. The DataNodes also perform block creation, deletion, and replication upon instruction from the NameNode.

3.3 Processing Layer

In this system, Yarn and MapReduce-2 [8] are located in the processing Layer to process vast amounts of data in-parallel on clusters of commodity hardware in a reliable, fault-tolerant manner. The MapReduce job splits the input data set into independent chunks which are processed by the map tasks in a completely parallel manner. The framework sorts the outputs of the maps, which are then input to the reduce tasks. Both the input and the output of the job are stored in HDFS. The framework takes care of scheduling tasks, monitoring them and re-executes the failed tasks. The MapReduce framework consists of a single

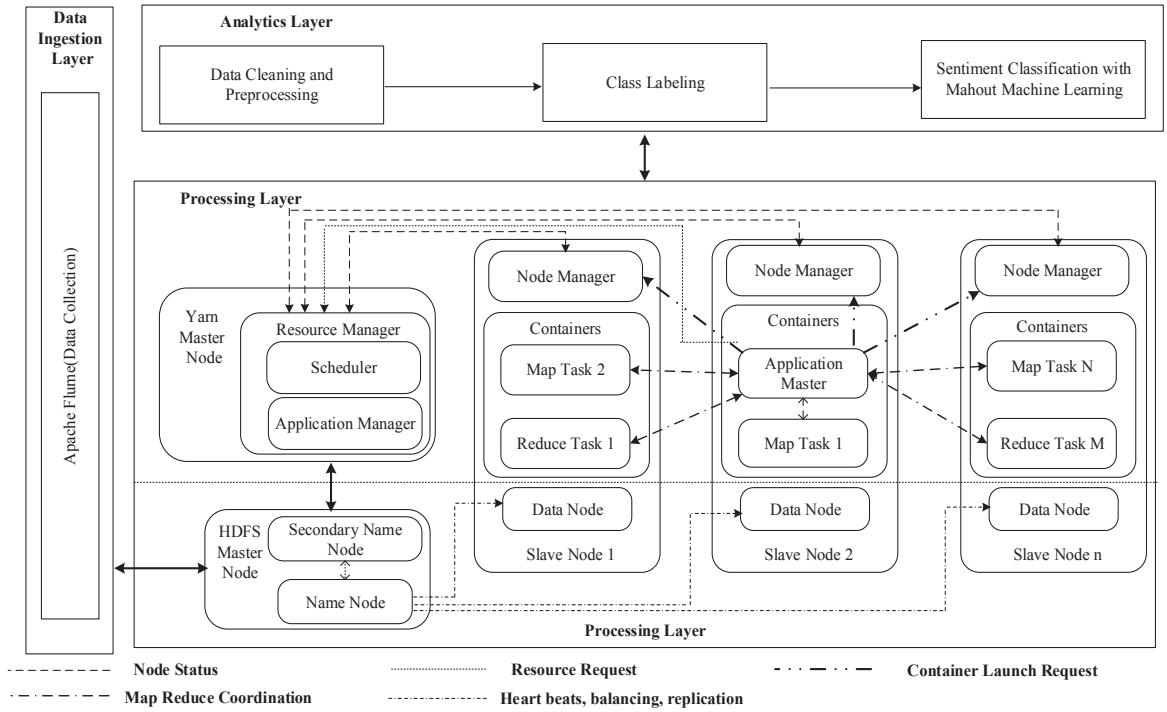

Figure 1 High Level Architecture of Proposed Sentiment Analysis System on Big Data Analytics Platform

master ResourceManager, one slave NodeManager per cluster-node, and MRAppMaster per application. The application specifies the input/output locations and supply map and reduce functions via implementations of interfaces and abstract-classes. The input/output locations and other necessary job parameters comprise the job configuration. The Hadoop job client submits the job and configuration to the ResourceManager which assumes the responsibility of distributing the software and configuration to the slaves, scheduling tasks and monitoring them, providing status and diagnostic information to the job-client.

3.4 Analytics Layer

Data cleaning and preprocessing, class labeling and sentiment classification are performed in the analytics layer. Sentiment classification is implemented by combining lexicon and learning based approaches. To be scalable classification, the lexicon and learning based classifiers are performed in distributed manner. SentiStrength is used for lexicon based classification and Mahout machine learning library is used for learning based classification. The Mahout machine learning library is specifically designed to use Hadoop for enabling scalable processing of huge data sets. Once the data is stored on the HDFS, Mahout provides the data science tools to automatically find meaningful patterns in big data sets.

4. SENTIMENT ANALYSIS SYSTEM FOR SOCIAL BIG DATA

To effectively extract the useful information from large volumes of social big data, the proposed Sentiment Analysis system is implemented with four modules: data collection, data cleaning and preprocessing, class labeling and sentiment classification. The process flow of the proposed Sentiment Analysis system is illustrated in Figure 2. The detail function of each process is explained in the following subsections.

4.1 Data Collection

In the proposed Sentiment Analysis system, a large volume of Twitter stream data that is generated from Twitter services is collected by Apache Flume. Twitter services and Flume are vital role of the data collection process and the brief explanations of Twitter services and Flume are described as follows:

4.1.1 Twitter Service

To access Twitter data, there are three different ways [6]: Twitter's Search API; Twitter's Stream API; Twitter's Firehose. In this work, Twitter's Stream API is used for collecting Twitter stream data. It is a push of data as tweets happen in near real-time. With Twitter's Streaming API, users register a set of criteria (keywords, usernames, location, etc.) and tweets match the criterias are pushed directly to the user. But, it is heavily based on the criteria users request and the current traffic.

4.1.2 Apache Flume

Apache Flume is a distributed, reliable, and available service for efficiently capturing, aggregating, and moving large amounts of log data. In this work, Flume is deployed by Twitter Agent in order to ingest Twitter stream data from the Twitter Service (Twitter Stream API), and forward it to HDFS through MemoryChannel. A Twitter Agent is an independent process that

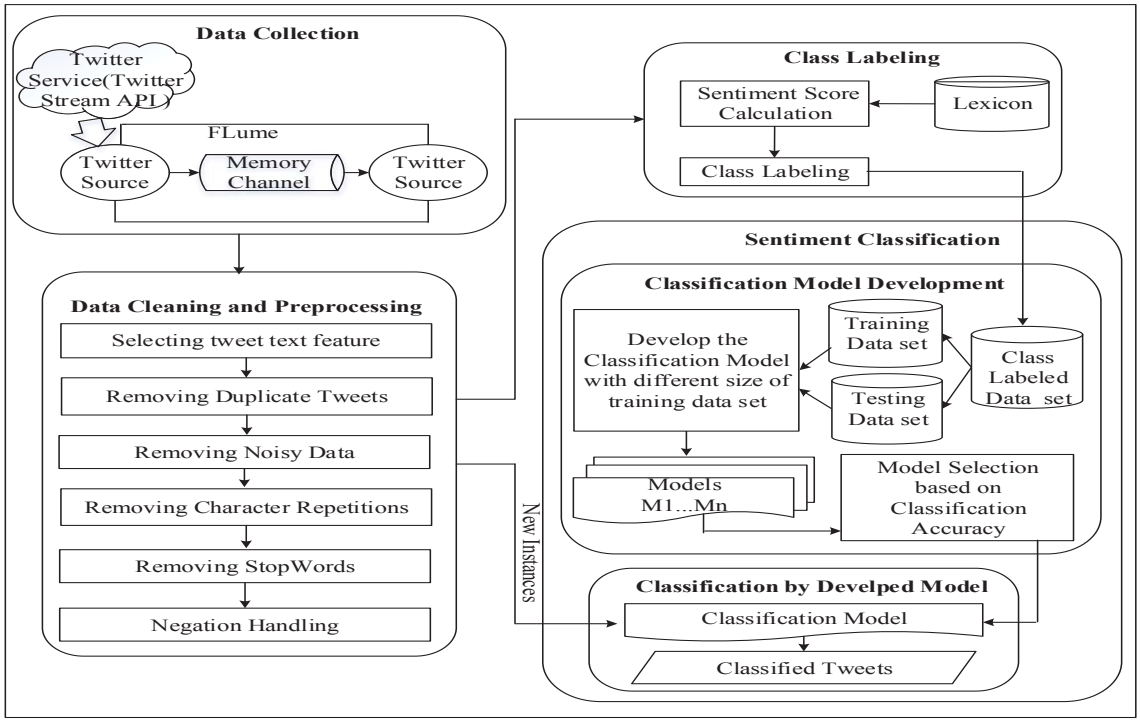

Figure 2 Process flow Diagram of the Proposed Sentiment Analysis System.

receives the data from its source and transports it to its destination. In reality, the process could be more complex where there might be multiple Agents getting data from the same source or an Agent getting data from the output of another Agent. Twitter Agent has three main components – a TwitterSource, a MemoryChannel and a HDFS Sink.

Twitter Source The source processes events and moves them along by sending the stream data into a Memory channel [10]. The sources operate by gathering discrete pieces of data, translating the data into individual events, and then using the channel to process events one at a time, or as a batch. In this system, cloudera Twittersource is used as a data source. Data is limited by key words. The source comes as an event-driven source. Event-driven sources typically receive events through mechanisms like callbacks or RPC calls. In the TwitterSource, the twitter4j library is used to keep access to the Twitter Streaming API. In order to connect to the Twitter APIs, [21] some of the application specific secrets like Consumer Key and Consumer Secret, Access Token, Access Secret are needed to be used. To get the application specific secrets, it is needed to create a Twitter application. The Twitter application is created by the following link https://apps.twitter.com/. It generates Consumer key, Consumer secret, Access token, and Access token secret.

Memory Channel The channel acts as a pathway between the TwitterSource and HDFS Sink. Events are added to the channel by TwitterSource, and later removed from the Channel by HDFS Sink. It uses as an in-memory queue to store events until they're ready to be written to a sink. As the channel holds all events in memory, the channel's capacity and transaction capacity are limited by the "capacity" and "transaction Capacity" parameters in the configuration file. In this work, the channel's capacity is set up to 10000 and the transaction capacity is set up to 100. The "capacity" parameter defines as the as the maximum number of events stored in the channel at any given time. The channel capacity is going to need to be sized such that it is large enough to hold as many events as will be added to it by upstream agents. The "transaction Capacity" parameter is defined as the maximum number of events the channel will take from a source or give to a sink per transaction. This going to vary depending on how many tiers of agents/collectors have been setups. In general though this should probably be equal to whatever you have the batch size set to in the client. This parameter is also a good defense against rogue clients pushing a huge number of events to a source, causing the agent to run out of memory. This parameter forces batches to be of limited size and thus limits the number of events per RPC call, and is a simple defense against denial of service (DoS) attacks.

HDFS Sink HDFS Sink, which writes events to a configured location in HDFS. In the HDFS Sink configuration, defines the size of the files with the roll count parameter and set up to 10,000, so each file will end up containing 10,000 tweets. It also retains the original data format, by setting the file Type to DataStream and setting writing Format to Text. This is done instead of storing the data as a Sequence File or some other format. The file path is defined as that the files will end up in a series of directories for the year, month, day and hour during which the events occur. For example, an event that comes in at 1/2/2017 3:00PM will end up in HDFS at hdfs://hadoop1:8020/user/flume/tweets/2017/1/2/17/. The timestamp is set to true in configuration and which is used by Flume to determine the timestamp of the event, and is used to resolve the full path where the event should end up.

4.2 Data Cleaning and Preprocessing

As flume ingests the raw Twitter stream data in nested JSON format and the raw data may contain irrelevant and duplicated data, data cleaning need to be performed. For cleaning the raw data, selecting tweet text features, removing duplicate tweets, removing noisy data are executed. Removing character repetitions, removing stopwords and negation handling are performed during the preprocessing. The preprocessing process not only simplifies the classification task, but also serves to greatly decrease the processing cost in the training phase.

4.3 Selecting Tweet Text Feature

Many tweet data features are included in one record of Twitter stream data and the proposed Sentiment Analysis system focus on tweet text features with English language. For Sentiment Analysis, tweet text feature is selected among other feature because it expresses twitter users' feeling and opinion.

Tweet id feature is also selected to assign as a key and tweet text feature is assigned as a value in Map Reduce process. To select tweet id and tweet text feature, the collected tweet data in JSON format is fetched as a JSON object using JSON parser. And "tweets_id", "tweets_text" and "tweet_lang" are extracted as a string from JSON object. "tweet_lang" is checked whether English or other. If "tweet_lang" is English, tweets-text and tweets-id will be check duplicate or not. If the duplication is detected in tweets, the system will remove the duplicate tweets. If "tweet_lang" is not English language, the system will move to the next line. Detail procedures of selecting tweet text feature and removing duplicate tweets are presented in Figure 3.

4.3.1 Removing Duplicate Tweets

Twitter Stream data may include many duplicate tweets. To remove duplicate tweets, the extracted "tweets_id" and "tweets_text" is checked whether already stored or not. If the extracted features have not already stored, they are added as the list of tweet data and then analyze the data. The selected sample tweet text and tweet id are shown in Table 1.

4.3.2 Removing Noisy Data

The term noisy data is used to describe any piece of information within the tweet that will not be useful for the machine learning algorithm to assign a class to that tweet. There are included noisy data [19] such as character repetitions, website links with URL, @username, punctuation additional white space Replace hash tags with the same word without the hash tags. For example, #fun is replaced with fun. Replaced website links with URL so links to websites that start with www.* or http. Converted @username to "usermentionsymbol by replacing @username instances found in tweets with "usermentionsymbol" for the classifier to easily identify that a user is being referenced. Non-Alphabets are replaced with space. After removing noisy data, sample tweet texts can be seen in Table 2.

4.3.3 Removing Character Repetitions

Tweets may contain a character that is repeating more than two times, like the word 'greeeeat'. It is important to replace words like this with their source words, so they can be merged. Otherwise, the classifier will treat them as different words, and probably the words will be ignored because of their low frequency of occurrence. Figure 4 shows the procedure which is the main function for removing character repetitions. This procedure called the other procedure i.e. "containsRepetitions" in order to check whether the repetitions contains or not. If the repetitive characters are found and the character count is more than 2, the procedure: "replaceRepetitions" is invoked for replacing the character itself. Finally the result is returned in this procedure. After removing character repetitions, the sample result can be seen in Table 3.

Figure 5 presents the procedure for checking whether the repetitive characters contain or not in the input data. Firstly, the former and pervious characters are compared by indexing word in one record of tweet. If the previous and current characters are the same, the character is counted. If the quantity of character is two or more, the procedure returns as true. Otherwise, it returns false. Detail procedures of replace repetitions are presented in Figure 6. If two or more repetitive characters are found, the procedure replaces the character by deleting the repetitive characters with substring function.

4.3.4 Removing Stopwords

When working with text classification methods, removal of stopwords is a common approach to reduce noise in the data. In this work, not only common stopwords but also stopwords based on classification domain are considered by manually examining the data. For example, domain stopwords contain iphone, apple, mobile, etc.

4.3.5 Negation Handling

Negation handling is one of the factors that significantly affect the accuracy of learning based classifier. For example: the word "good" in the phrase "not good" will be contributing to positive sentiment rather that negative sentiment as the presence of "not" before it is not taken into account. To solve this problem, the simple procedure for handing negations is described in Figure 7. It transforms a word followed by a not or n't into "not_" + word.

4.4 Class Labeling

Instead of manually labeling the class, SentiStrength lexicon based classifier is used for the task of annotating the training data for the learning-based classifier. SentiStrength [16], a lexicon-based classifier, uses additional (non-lexical) linguistic information and rules to detect the sentiment strength in short informal English text. The contextual valence shifter: negation and intensifier are used to evaluate the context sentiment and to solve the context dependent problem by applying NegationWordList and BoosterWordList. There are consists of eight dictionaries: BoosterWordList, EmoticonLookupTable, EmotionLookupTable, EnglishWordList, IdionLookupTable, Negation-

Procedure: SelectingTweetText _And_Remove Duplicate Tweets

Input: rawtweets // rawtweets is JSON file; Output: tweet_text_features

```
1.  Begin
2.    tweetsarray[ ] rawtweets.split(" \\n ");
3.    int i 0
4.    while(i < tweetsarray.length( ))
5.      Create JSON Object " obj" of tweetsarray[i]
6.       tweet_lang String.valueOf (obj.get("lang"))
7.      if (Tweet_lang.equals("en"))
8.        Create JSON Object "JSONobj" of tweetsarray[i]
9.          tweets_id String.valueOf (JSONobj.get("id"))
10.       if(!tweets_idno.contains(tweets_id))
11.           Add tweets_id into tweets_idno
12.             tweets_text String.valueOf(JSONobj.get("text"))
13.           Convert all of the tweets text to lower case
14.           if(!tweets_text_feature.contains(tweets_text))
15.                Add tweets_text into tweet_text_features
16.                  i ← i+1
17.             endif
18.         endif
19.       else i=i+1
20.     endif
21.   endwhile
22. end
```

Figure 3 Selecting Tweets_ text feature and Removing Duplicate Tweets Procedure.

Table 1 Sample tweet_texts and tweets id.

tweet_id	tweets_text
88841	I liked a @YouTube video from @booredatwork https://t.co/lNPwtDWD2z iPhone 7 Review: hmmmmm.....???

Table 2 Sample tweet texts and tweets id that is Removed Noisy Data.

tweet_id	tweets_text
88841	I liked a usermentionsymbol video from usermentionsymbol urlinksymbol iPhone 7 Review hmm

WordList, QuestionWords and slangLookupTable.

EmotionLookupTable consists of 2546 sentiment words with their strength. Some words include Kleene star stemming (e.g., ador*). In this work, other domain specific words (e.g., smartphone) are added with their polarity values. The word "miss" is a special case with a positive and negative strength of 2. It is

Procedure : Removing_Character_Repetitions

Input: rawtweets
Output: tweets which is removed two or more character repetitions

```
1. Begin
2.    current_tweets ← rawtweets
3.    temp ← current_tweets.replaceAll( " ^A-Za-z ", " ")
4.    if(temp.length() > 0 && containsRepetitions(temp))
5.       temp ← replaceRepetitions(temp)
6.      return temp
7.    else return current_tweets
8.    endif
9. end
```

Figure 4 Removing Character Repetitions Procedure.

Table 3 Sample Raw Tweets and Cleaned Tweetby Removing Character Repetitions.

Raw Tweets	**Cleaned Tweets by Removing Character Repetitions**
If I put my iPhone on "Do Not Disturb" WHHHYYY am I still getting phone calls? I've recorded this... https://t.co/mEcIBmjLN2	If I put my iPhone on Do not Disturb WHHYY am I still gting phone calls i have recorded this urlinksymbol
RT @ThvGuySpvzz: iPhone 7 Camera soooooo clear you can really see mf's souls lifting outta their body from receiving that bomb head	RT usermentionsymbol iPhone 7 Camera soo clear you can really see mf s souls lifting outta their body from receiving that bomb head

frequently used to express sadness and loves simultaneously. A spelling correction algorithm deletes repeated letters in a word when the letters are more frequently repeated than normal for English or, if a word is not found in an English dictionary, when deleting repeated letters creates a dictionary word. A booster word list is used to strengthen or weaken the emotion of sentiment words. A booster word list consists of 28 words and their strength of sentiment. Some booster words are "totally, completely and might". An idiom list is used to identify the sentiment of a few common phrases. This overrides individual sentiment word strengths. Negations are used to reverse the semantic polarity of a particular term, and skip any intervening booster words. Default multiplier for negated words is 0.5. In this case, instead use of default multiplier for negated words, the multiplier is set to 1. At least two repeated letters added to words give a strength boost sentiment words by 1. For instance, haaaappy is more positive than happy. Neutral words are given a positive sentiment strength of 2 instead. An emoticon list with polarities is used to identify additional sentiment. The emoticon list consists of 116 common emoticon words and their polarity strength (-1 and 1). Sentences with exclamation marks have a minimum positive strength of 2, unless negative. Negative sentiment is ignored in questions.

The procedure of calculating the sentiment score by applying SentiStrength are presented in Figure 8. The procedure is performed in a distributed manner using Map Reduce function. At the Mapper stage, the collected raw data is parsed with the JSONParser in order to select the tweets and tweets_id. After cleaning the data, the total polarity strength is calculated for each sentence by using SentiStrength_Data. And the total sentiment score is calculated by combining the strength of positive sentiment (1 to 5) and negative sentiment (-1 to -5). For each text, the output score is two integers: 1 to 4 for positive sentiment strength and a separate score of - 1 to - 4 for negative sentiment strength. If the sentiment score is equal and greater than 1, the output label is "positive. If the sentiment score is equal to 0, the output label is "neutral". If the score is equal and less than -1 is negative. As the result combination is not needed, the Reduce stage outputs the results obtained by the Mappers. The Sample class labeled training data is shown in Table 4.

4.5 Classification Model Development

Mahout naïve Bayes classifier [12], scalable machine learning algorithm, is conducted to develop the classification model. Naïve

Procedure: Checking_Repetitions

Input: rawtweets
Output: checking results of whether repetitions contains or not//true or false

```
1.  Begin
2.    tweets_text ← rawtweets
3.    return_tweets ← tweets_text.substring(0,1)
4.    previous_tweets ← tweets_text.charAt(0)
5.    count ← 0
6.    for(index=1; index ← tweets_text.length(); i++)
7.      current_tweets ← tweets_text.charAt(i)
8.      return_tweets ← return_tweets + current_tweets
9.      if(current_tweets == previous_tweets)then
10.       count ← count+1
11.       if (count >= 2)
12.         return true
13.       else count ← 0
14.       previous_tweets ← tweetext.charAt(index)
15.       return false
16.     endif
17.     endif
18.   endfor
19. end
```

Figure 5 Checking Repetitions Procedure.

Table 4 Sample Training Data.

Tweet Text	Class
Woohoo my father is going to gift me an iPhone for the success of my research GreatNewsIsHere	positive
I got the iPhone 7 but I can not connect it because I do not have wifi at home and my stupid carrier stopped letting me use personal hotspot	negative
When your boyfriend buys you an iPhone 7 on NationalBoyfriendDay he is a keeper	neutral

Bayes is a learning algorithm that is frequently employed to tackle text classification problems. It is computationally very efficient and easy to implement. Figure 9 illustrates the procedure of classification model development. The class labeled data is used as the input data and the input data is preprocessed by applying preprocessing steps. The preprocessed class labeled data set is split into training and testing datasets in order to build the classification model. And then the training and testing data are transformed into the sequence file. As this sequence file consists of key values pairs, class category and tweet_id are set to key and tweets text are set to value. Then Feature generation is performed by using the sparse vector function. In feature generation, TFIDF feature vectors are generated for improving the performance of classification model. The TFIDF vectors are used to train the classification model. Different classification models are developed by applying different size of the training data set. For each developed model, classification accuracy is calculated to select suitable model. The suitable model is se-

Procedure: Replace_Repetitions

Input: rawtweets
Output: replaced repetitions tweets

```
1.  Begin
2.    tweets_text ← rawtweets
3.    return_tweets tweets ← text.substring(0,1)
4.    previous_tweets ← tweettext.charAt(0)
5.    found ← false
6.    for(index=1; index<tweettext.length(); i++)
7.      current_tweet ← tweettext.charAt(i)
8.      return_tweets ← return_tweets + current_tweets
9.      if(currenttweet == previous_tweets)
10.       if (found == true)
11.       return_tweets ← return_tweets.substring(0,toreturn.length()-1)
12.     else found ← true
13.     endif
14.   else if (found == true)
15.           found ← false
16.           previous_tweets ← tweettext.charAt(i)
17.       endfor
18.     end
19.   return return_tweets
20. end
```

Figure 6 Replacing Repetitions Procedure.

lected by comparing the accuracy of classification models with different size of training datasets.

4.5.1 Classification by Developed Model

The newly incoming tweets are classified by using selected model. In Figure 10, the procedure of sentiment classification for new instances with distributed manner using MapReduce function is described. At the Mapper stage, data cleaning and preprocessing need to be performed for the new incoming tweets to be effective classification. Word id and tfidf weights are used to create vectors of the new tweets. With the classifier, the vector score is calculated by applying vector and developed model. To calculate the output results (class category), the "bestscore" is set to "-Double. MAX_VALUE", the "bestcategory-Id" is set to "-1" and "category-Id" is set to index of vector score. If the vector score is greater than bestscore, bestcategoryId is replaced with categoryId. If the "bestcategory-Id" is equal with "0", the classifier classify as "positive". If the "bestcategory-Id" is equal with "1", the classifier classify as "neutral". If the "bestcategory-Id" is equal with "2", the classifier classify as "negative". The naïve Bayes algorithm is considered naive because it assumes that the value of a particular feature is independent of the value of any other feature, given the class variable. Due to the fact that the result combination is not needed, the Reduce stage outputs the results obtained by the Mapper function.

5. EXPERIMENTS AND RESULTS

In this section, experiment parameters of the proposed system, data sets and explanations about evaluation result are presented.

Procedure: Negation_Handling

Input: tweets
Output: negation handled tweets

```
1. Begin
2.    negated ←false
3.    if the tweets contains negated words:
4.     negated ← true
5.     for each words in tweets:
6.       Replace word to "not_" + word
7.     endfor
8.    endif
9. end
```

Figure 7 Negation Handling Procedure

Procedure: Class_Labeling_Job

```
1. Input: k: tweets_idno, v: tweet_text_feature //k:key, v:value
2. Output: k:tweets_indo, v: class labeled data
3. Function MAPPER(key, values)
4.    while(value ∈ values):
5.       Perform Data Cleaning
6.       Calculate the polarity strength for each sentence
7.       Calculate the total score by combining the strength of sentiment
8.          if (score >= 1) then print "positive"
9.          else if (score < = -1) then Print" negative"
10.         else if (score == 0) print "neutral"
11.            Emit (twee & tweet_id, class label)
12.         endif
13.      endwhile
14. Function Reducer(key, values)
15.   while(value ∈ values):
16.      Emit(tweet & tweet_id, class label)
17.   endwhile
```

Figure 8 Class Labeling Procedure.

5.1 Experiment Environment

The specifications of devices and necessary software component of the proposed system are described in Table 5.

Procedure : Developing_Classification_Model

Input: class labeled data
Output: classification model

1. Begin
2. Perform data preprocessing
3. Split the input data into training and testing datasets
4. Transform training and testing datasets into sequence file
5. Convert sequence file to TFIDF feature vector
6. Train the classification model using the vector
7. Develop the classification models with different size of training data set
8. Calculate classification accuracy for each developed model
9. Select model based on classification accuracy
10. end

Figure 9 Developing Classification Model Procedure.

Table 5 Experiment Parameters

Parameters	Specification
Server/Client OS	Ubuntu 14.04 LTS
Host Specification	Intel ® Core i7-3770 CPU @ 3.40GHz, 8GB Memory, 1TB Hard Disk
VMs Specification	4GB RAM, 100 GB Hard Disk
Software Component	- Hadoop 2.7.1 - Flume 1.6 -SentiStrength2 - Mahout 0.10.0

5.2 Data Sets

In order to test the functionality of the proposed system and prove the achieved results with promising accuracy, tweets stream data related with iphone product is examined. The data are collected for two months from January to February in 2017. 200,000 tweet data are utilized as the training datasets and 50000 new batch of tweets are applied as the test set for evaluation of the performance of sentiment classification.

5.3 Evaluation Results

In the experiment, the cluster is composed of 4 computing nodes (VMs) with one name node and three data nodes.

To present the evaluation results of the system, beginning from evaluating the performance of lexicon based classifier (SentiStrength). To establish the ground truth, the evaluation result of lexicon based classifier is compared with manual classification. The comparative results for the performance of lexicon based classifier and manual classification are illustrated in Figure 11. In this work, 10,000 tweets are randomly selected from training data for evaluation of lexicon based classifier. Tweets Percentages of classification by SentiStrength for neutral, positive and negative class label are 39, 30 and 31. Manually classified Tweets percentages for neutral, positive and negative class label are 52, 23 and 25. Therefore, error rate for Lexicon based classifier is 13% in neutral class, 7% in Positive class and 6% in negative class label. The overall accuracy rate is 74% and error rate is 26%.

The Twitter stream dataset is preprocessed for improving the performance of the classifier. The classification accuracy of Mahout naïve Bayes classifier using the preprocessing steps is presented in Table 6. The results show that the preprocessing can be able to improve the classifier.

For building the classification model, the preprocessed class

Procedure : Classification_Job

Input: k1: tweet_id, v1: Newly Collected Tweet Stream Data //k, k1, k2 : key, v, v1, v2 : values
Output: k: tweets, tweet_id, v: class category //class category : positive || negative || neutral

```
1. Function Mapper(k1, v1)
2.   while(value G= values)
3.     Performed Data cleaning
4.     Applied the Preprocessing steps for classification
5.     Create vector by using word-id, tfidf value
6.     Calculate vector_score by applying vector and developed model
7.     Assign bestscore to "-Double. MAX_VALUE", "bestcategory-Id" to "-1" and "category-Id" to index of vector_score
8.     if (vector_score > bestscore)
9.       Replace bestcategory-Id to category-Id
10.      if (bestcategory-Id == 0)
11.        Print class category as " positive"
12.      else if (bestcategory-Id == 1)
13.        Print class category as " neutral"
14.      else if (bestcategory-Id == 2)
15.        Print class category as " negative"
16.      end if
17.    end if
18.  Emit(tweets & tweet_id, class category)
19. End function
20. Function Reducer(k2, v2)
21.   while(value G= values)
22.     Emit(tweets & tweet_id, class category)
23. End function
```

Figure 10 Classification by Developed Model Procedure.

Table 6 Classification Accuracy of Mahout Naïve Bayes Classifier.

Feature	Accuracy
Character Repetition Removal	78.16
Character Repetition Removal + Stopwords Removal	79.38
Character Repetition Removal+ Stopwords Removal + Negation Handling	82.56

labeled dataset is divided into two disjoint parts: training and testing dataset. These two datasets is used to generate (or fit) the model. Different training and test dataset can affect the accuracy of classification model. In Figure 12 shows the classification accuracy changes while varying the training and testing dataset sizes. The difference between the minimum and maximum clas-

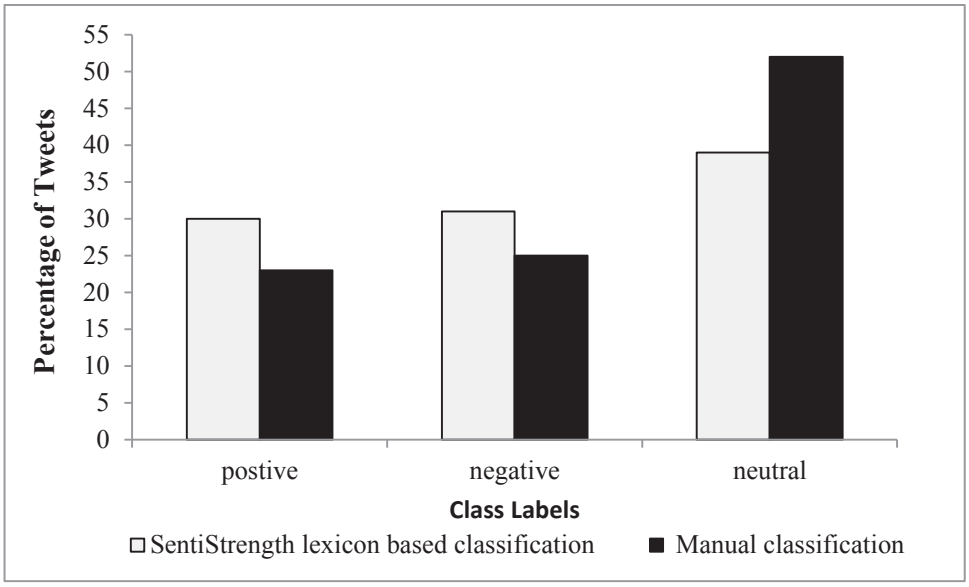

Figure 11 Percentage of Tweets on Lexicon based classification and Manual classification

sification accuracies when varying the training set size is little about 8%. The highest accuracy of the classification model is 82.56 while training set is 80% and the model is selected for classifying new instances.

After selecting the classification model, new test sets are classified by applying the model. Table 7 shows the accuracy and F-Measure of the proposed Sentiment Analysis system and the overall evaluation results show that the proposed system achieves the promising accuracy by 84.2%.

In order to test the scalability of this system, the proposed Sentiment Analysis system runs the job with different number of tweets on different nodes. Each node is developed on each machine. Figure 13 shows the processing time of the proposed Sentiment Analysis system. According to the result, the processing time of the system with different volumes of data decreases when adding more nodes into the cluster.

6. CONCLUSION AND FUTURE WORK

In this paper, Big Data Analytics platform is developed to scale up the traditional analytics platform for analyzing large scale data. SA is implemented on Big Data Analytics platform for extracting useful information from large volumes of social big data. Hadoop is built for big data analytics and it is a good platform for being able to manage large data at scale and which can improve scalability and efficiency by adopting distributed processing environment since they have been implemented using MapReduce and HDFS. The proposed Sentiment Analysis system consists of four modules: data collection, data cleaning, class labeling and preprocessing, sentiment classification. These modules are implemented at four layers: Data Ingestion Layer, Storage Layer, Processing Layer and Application Layer. The sentiment classification is implemented by combining lexicon and supervised machine learning-based approach. Lexicon-based approach is adopted to reduce time and labor consuming for manual labeling while SentiStrength lexicon-based classifier is applied to label the class. The system classifies the polarity (positive, negative and neutral) on the real Twitter stream data. The system enables high-level performance of learning based classification while taking advantage of the lexicon-based classifier's effortless setup process. Evaluation results show that the reliability of the performance of lexicon-based classifier by comparing manual classification results and achieved the accuracy rate is 74%. The class labeled dataset is preprocessed and the evaluation results show that preprocessing can be able to improve the accuracy of classifier. To achieve high-level performance of the classification model, the suitable model is selected by comparing the accuracy of Mahout naïve Bayes classifier with different size of training datasets. According to the evaluation results, the highest accuracy of the classification model is 82.56% while training set is 80%. The overall accuracy and f-measure of the proposed Sentiment Analysis system are 84.2% and 83.0%. The scalability of the system is evaluated with processing time and the evaluation results show that the running time of the system with different volumes of data decreases when adding more nodes into the cluster. In future work, the proposed Sentiment Analysis system will be implemented on Spark for real time analysis of large scale social data.

REFERENCES

1. A. Assiri, A. Emam, H. Al-dossari, "Real-Time Sentiment Analysis of Saudi Dialect Tweets Using SPARK", IEEE International Conference on Big Data, pp. 3947-3950, December 2016.
2. A. Assiri, A.Emam and H. Al-Dossari. "Saudi twitter corpus for Sentiment Aanalysis". International Journal of Computer, Electrical, Automation, Control and Information Engineering, World Academy of Science, Engineering and Technology, Vol 10, pp. 242–245, 2016.
3. A. Baltas, A. Kanavos, A. K. Tsakalidis, "An Apache Spark Implementation for Sentiment Analysis on Twitter Data", Algorithmic Aspects of Cloud Computing, pp. 15-25, April 2017
4. A. C. Oliver. "Machine-learning-with-mahout". Avail-

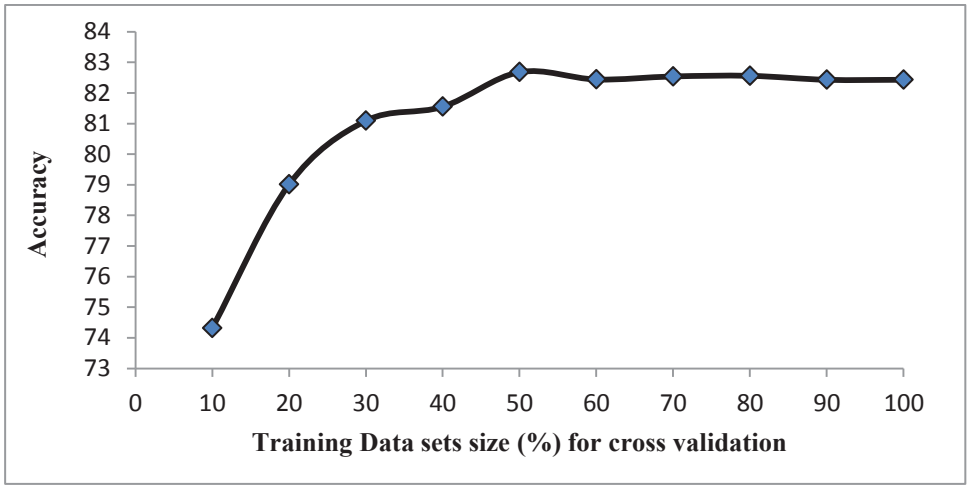

Figure 12 Classification Accuracy for Different size of Training Dataset.

Table 7 Classification Accuracy and F-Measure of Proposed Sentiment Analysis system.

	Accuracy (%)	F-Measure (%)
Positive	78.5	81.5
Negative	85.3	84.2
Neutral	88.7	83.4
Overall	**84.2**	**83.0**

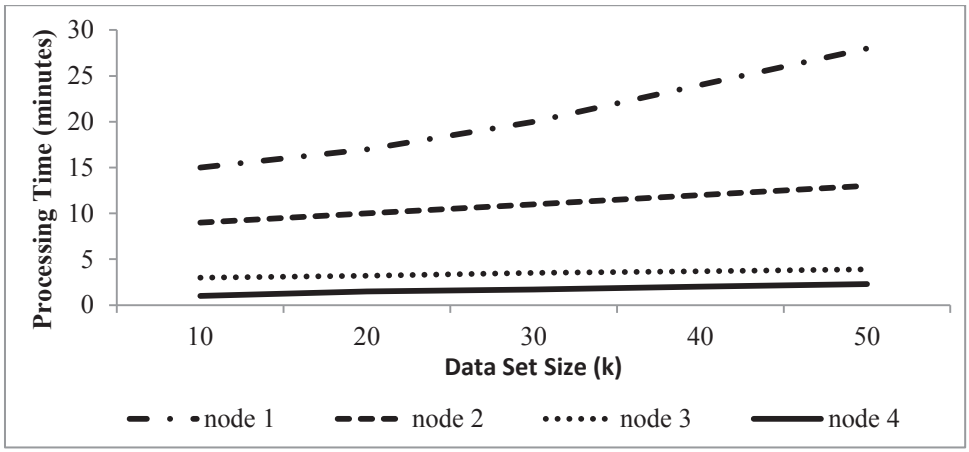

Figure 13 Processing Time of the Proposed Sentiment Analysis System.

able: http://www.infoworld.com/article/2608418/application-development/enjoy-machine-learning-with-mahout-on-hadoop.html, May 2014, [Online; accessed 3-December-2016].

5. B. Liu, E. Blasch, Y. Chen, D. Shen, G. Chen, "Scalable Sentiment Classification for Big Data Analysis Using Naive Bayes Classifier", IEEE International Conference on Big Data, pp. 99-104, 2013
6. B. Milsom, "Twitter API vs Firehose", Available: https://www.echosec.net/twitter-api-vs-firehose/, June 2015 [Online; accessed 13-September-2016]
7. D. Effrosynidis, S. Symeonidis, A. Arampatzis, "A Comparison of Pre-processing Techniques for Twitter Sentiment Analysis", 21st International Conference on Theory and Practice of Digital Libraries, Volume: LNCS 10450, pp. 394-406, September 2017.
8. "Hadoop Yarn", Available at "https://hadoop.apache.org/docs/r2.7.1/hadoop-yarn/hadoop-yarn-site/YARN.html. " [Online; accessed 20-August-2016]
9. H. Thakkar and D. Patel, "Approaches for Sentiment Analysis on Twitter: A State of Art study", CoRR abs/1512.01043, December 2015.
10. Jon Natkins, "Analyzing Twitter data with Hadoop", Available: http://blog.cloudera.com/blog/2012/10/analyzing-twitter-data-with-hadoop-part-2-gathering-data-with-flume/, October, 2012, [Online; accessed 28, August-2016].
11. K. Bannister, "Sentiment Analysis". Available: https://www.brandwatch.com/blog/understanding-sentiment-analysis/ [Online; accessed 15-October-2016].
12. L. Giura. "Sentiment Analysis using Mahout Naïve Bayes", Available: http://technobium.com/sentiment-analysis-using-mahout-Naive -Bayes/, [Online; accessed 20-Oct-2016].

13. L. Velikovich, S. Blair-Goldensohn K. Hannan, R. McDonald, " The viability of web-derived polarity lexicons ", The 2010 Annual Conference of the North American, pp. 777–785, June 2010.
14. L. Zhang, R. Ghosh, M. Dekhil, M. Hsu, and B. Liu, "Combining Lexicon-based and Learning-based Methods for Twitter Sentiment Aanalysis", HP Laboraties, 2011.
15. M. Skuza, A. Romanowski, " Sentiment Analysis of Twitter Data within Big Data Distributed Environment for Stock Prediction", Computer Science and Information Systems (FedCSIS), pp. 13-16, Sept 2015.
16. M. Thelwall, K. Buckley, G. Paltoglou, "Sentiment Strength Detection for the Social Web", Journal of the American Society for Information Science and Technology, pp. 163-173, January 2012.
17. N. Nodarakis, S. Sioutas, A. Tsakalidis, G. Tzimas, " MR-SAT: A MapReduce Algorithm for Big Data Sentiment Analysis on Twitter", 12th International Conference on Web Information Systems and Technologies, pp. 23-25, April 2016.
18. N. M. Sharef, H. M. Zin and S. Nadali, "Overview and Future Opportunities of Sentiment Analysis", Journal of Computer Sciences, January 2016.
19. P. Zikopoulos, C. Eaton, D, DeRoos, T. Deutch and G. Lapis, "Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data", McGralw-Hill Osborne Media, pp. 176, 2011.
20. S. R. El-Beltagy and A. Ali, "Open issues in the Sentiment Analysis of Arabic social media: a case study", 9th International Conference on Innovations in Information Technology, pp. 5–220, 2013.
21. S. Shaikh. "Flume Installation and Streaming twitter data using flume", Available: https://www.eduonix.com/blog/bigdata-and-hadoop/flume-installation-and-streaming-twitter-data-using-flume/ [Online; accessed 20-Oct-2016]
22. "Twitter Statistics", Available at: http://www.statisticbrain.con/twitter-statistics/ [Online; accessed 2-January-2013]
23. V. N. Khuc, C. Shivade, R. Ramnath, J. Ramanathan, "Towards Building Large-Scale Distributed Systems for Twitter Sentiment Analysis", 12 Proceedings of the 27th Annual ACM Symposium on Applied Computing, pp. 459-464, March 2012.