A Distributed Framework for Real-Time Twitter Sentiment Analysis and Visualization

Jamuna S. Murthy, G. M. Siddesh and K. G. Srinivasa

Abstract Social networking site such as Twitter contributes to the huge amount of text data every day and hence provides the best opportunity for sentiment analysis (SA). Existing systems for SA used Hadoop and MySQL and hence lacked real-time analysis. Thus, this research work aims at introducing a distributed framework that processes and analyzes tweets in real time using apache storm and Redis database. Proposed system focuses on key aspects needed for SA in terms of data collection, data parsing, and data visualization. A novel algorithm called Emoticon-Polarity-SentiWordNet (EPS) is used for Twitter sentiment analysis and the classification results are visualized on a real-time web application. Evaluation results proved that our system outperforms the existing system.

Keywords Real-time data analytics · Hadoop · Apache storm
Redis · D3

1 Introduction

Nowadays, people use Twitter for posting short messages called tweets to express their thoughts and opinions on different aspects ranging from simple ones such as "Hey @xyz!!Gunnyt☺sleep" to themes such as "#IPL-2017". Tweets are undoubtedly rich with user information and thus performing SA on the tweets offers companies and organizations a fast and effective way of planning marketing strategies and helps in quick decision-making [1]. But due to the tweet length

J. S. Murthy · G. M. Siddesh
Department of ISE, Ramaiah Institute of Technology, Bengaluru, India
e-mail: jamunamurthy.s@gmail.com

G. M. Siddesh
e-mail: siddeshgm@gmail.com

K. G. Srinivasa (✉)
Ch. Brahm Prakash Government Engineering College, New Delhi, India
e-mail: kgsrinivasa@gmail.com

P. K. Sa et al. (eds.), *Recent Findings in Intelligent Computing Techniques*,
Advances in Intelligent Systems and Computing 709,
https://doi.org/10.1007/978-981-10-8633-5_6

restriction of 140 characters, many people use irregular expressions, emoticons, smiles, and abbreviations for saving the room of space for their tweets. This has increased tweet sentiment analysis problems to greater extent showing data sparsity and sarcasm. Over the years, many classifiers were trained by the researchers for sentiment classification of tweets but most of the algorithms used the traditional approaches such as bag of words model, unigrams–bigrams model, POS Tagging, etc., which lead to less classification accuracy. Also, the existing framework for SA used Hadoop for processing a massive set of tweets and used MySQL database for querying. But Hadoop and MySQL are traditional approaches and do not support analysis over real-time. Hence, the proposed system aims at implementing a distributed real-time framework for Twitter sentiment analysis with a novel classification algorithm called EPS. The keys aspects of the framework lie in implementing three major modules called the data collection, data parsing, and data visualization.

2 Literature Review

Sentiment analysis deals with analyzing the user-generated data and hence many researchers have worked on the same and proposed different techniques which are discussed in this section. Name entity recognition (NER) was implemented by Alan Ritter et al. [2] which tried to rebuild natural language processing (NLP) pipeline using POS tagging and polarity-based sentiment classification with an accuracy of 72%. Vinh et al. [3] used a combination of lexicon-based approach and naïve Bayes classifier for sentiment classification using MapReduce and increased the classification accuracy up to 73.7%. Pang et al. [4] reported an accuracy of movie review tweets with 81%, 80.4%, and 82.9% for naive Bayes, MaxEnt, and SVM-based classifier, respectively. Balamurali et al. [5] proved that unigram and trigram model outperforms the trigram model when used with naive Bayes classification for tweet sentiment analysis. On the other hand, the reverse was true in case of SVM and MaxEnt which was proved by Chang et al. [6]. Although the above methods tried to increase the classification accuracy to some extent, the proposed EPS algorithm outperforms all the existing classifiers by increasing the classification accuracy to 85%.

3 Proposed Work

The proposed framework for real-time SA consists of three major modules called data collection, data parsing, and data visualization as shown in Fig. 1 The main objective of "Data Collection" module is to capture the sparse tweets continuously over real time. The tweets are polled using Twitter streaming API with the help of java library Twitter4j and are consumed using Kafka [7], a distributed message queuing system. Thus, tweets collected using kafka are injected to Apache Storm [8] for data parsing. The "Data Parsing" module includes two components data

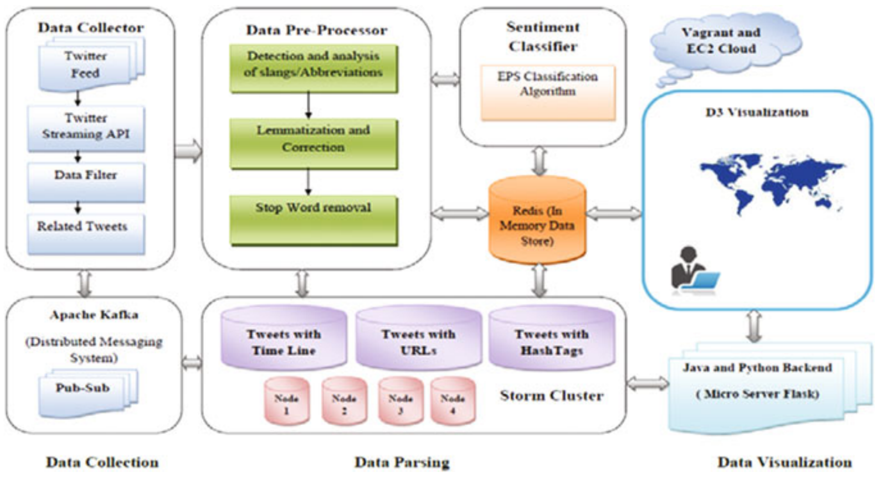

Fig. 1 Proposed architecture

preprocessor and sentiment classifier. Data preprocessor acts as a natural language processing pipeline to extract the important features from the tweet using dictionary based approach. For each tweet, at first, detection and analysis of slangs and abbreviations are done to correct the irregular spells. Later, the longer words are stemmed to short words and are corrected. Next, the skip words such as "http", "to", etc., are removed. Finally, the preprocessed tweet is sent to sentiment classifier for further classification. The sentiment classifier implements proposed EPS algorithm in terms of three classifiers called improved emoticon classifier (IEC), enhanced polarity classifier (EPC) and SentiWordNet Classifier (SNC) and the algorithm description is given below.

The set of tweets T is defined as

$$T = \{t_1, t_2, \ldots, t_n\}$$

If each tweet t contains w words then set of words W is defined as

$$W = \{w_1, w_2, \ldots, w_m\}$$

Then the sentiment score S, calculated for each word is given as

$$\text{Score} = \sum_{i=1}^{n} \sum_{j=1}^{m} S_{t_i w_j}$$

Step 1: IEC does classification of emoticons. First, the regular expression is used to detect the presence of emoticons in tweets. A manually tagged rich list of emoticons with positive and negative is initialized. The emoticon in the tweet is matched against the defined positive and negative list to obtain matched positive and negative emoticon scores. Later, the two scores are added to get aggregate score. Finally, the tweet is given a value 1 if the aggregate score is greater than zero, it is assigned −1 if less than zero and the score 0 indicated that the calculated sum is zero. The tweets which cannot be classified using emoticon analysis are considered to be neutral and classified using other two classifiers. Let P_E = {positive emoticons list} and N_E = {negative emoticons list} be two lists tagged as input to IPC then step 1 can be represented as:

$$\text{Score}(e) = \begin{cases} 1, (w_x \in W) \wedge (t \in T) \wedge (w_x \in \text{PE}) \\ -1, (w_y \in W) \wedge (t \in T) \wedge (w_y \in \text{NE}) \\ 0, (w_2 \in W) \wedge (t \in T) \wedge (w_2 \notin \text{PE}) \wedge (w_2 \notin \text{NE}) \end{cases}$$

where Score(e) is emoticon score and w_x, w_y, and w_z are the words from W and t is a tweet from T.

Step 2: EPC takes the input as two lists which are positive words list and negative words list this is known as “bag of words tagging”. Generally, words are domain independent. IPC works only with words with correct spelling and if there is a combination of positive or negative in tweet it is classified as neutral and addressed using SNC. Words are identified based on splitting the words in tweets with delimiter. The words list used are collected from [9] and both the lists are combined to obtain roughly around 9500 words. EPC is an enhancement from [10] and hence the name. It works similar to IEC except that their emoticons are used and here words are used. Let P_W = {positive words list} and N_W = {negative words list} then Step 2 is simply represented:

$$\text{Score}(w) = \begin{cases} 1, (w_x \in W) \wedge (t \in T) \wedge (w_x \in \text{PW}) \\ -1, (w_y \in W) \wedge (t \in T) \wedge (w_y \in \text{NW}) \\ 0, (w_2 \in W) \wedge (t \in T) \wedge (w_2 \notin \text{PW}) \wedge (w_2 \notin \text{NW}) \end{cases}$$

where Score(w) word score of IPC, w_x, w_y, and w_z are the words from W and t is a tweet from T.

Step 3: SNC is based on the SentiWordNet dictionary and the tweets are classified based on sentiment identified from the dictionary and it assigns different weights for the words based on the type of sentiment using parts of speech. Similar to the previous step in IPC, here too the words are delimited and the sentiment value is calculated using SentiWordNet library. The aggregate sentiment score is calculated by adding the weight of each word which was assigned using SentiWordNet. Finally, the tweet is given the value 1 if the aggregate score is greater than zero, it is assigned −1 if less than zero and the score 0 indicated that the calculated sum is zero. Step 3 can be represented in a simple way as:

$$\text{Score}(s) = \begin{cases} 1, (w_x \in W) \wedge (t \in T) \wedge (\text{weight}(w_x) > 0) \\ -1, (w_y \in W) \wedge (t \in T) \wedge (\text{weight}(w_y) < 0) \\ 0, (w_2 \in W) \wedge (t \in T) \wedge (\text{weight}(w_2) < 0) \end{cases}$$

where Score(s) is sentiment score of SNC and weight (w_x), weight (w_y), weight (w_z) are the words from W and t is a tweet from T.

Step 4: For the refined tweets from the preprocessor first IEC in Step 1 is applied, next EPC and finally SNC. If IPC classifies the tweet as neutral, it goes to Step 2 and if the tweet is classified as neutral at this step too, it moves to Step 3. The tweets which are not classified using any classifier are considered to be neutral. Final classification step can be expressed as

$$\text{Class} = \begin{cases} \text{Positive}, (S_e > 0) \vee (S_e = 0 \wedge S_w > 0) \vee (S_e = 0 \wedge S_w = 0 \wedge S_s > 0) \\ \text{Negative}, (S_e < 0) \vee (S_e = 0 \wedge S_w < 0) \vee (S_e = 0 \wedge S_w = 0 \wedge S_s < 0) \\ \text{Neutral}, (S_e = 0) \wedge (S_w = 0) \wedge S_w = 0 \end{cases}$$

The tweets parsed during the data parsing module are stored in the Redis database and are visualized using “Data Visualization” module. The visualization module is built with the help of javascript library D3 and using web application library Flask.

4 Evaluation Results

The results are evaluated as throughput and scalability to check the performance of data collection and data parsing modules. Throughput is evaluated as the number of messages collected using the data collection module. Series of three tests were conducted on three different days and the average is taken to discuss the results. From Fig. 2, we observe that the average rate of input messages increases when the

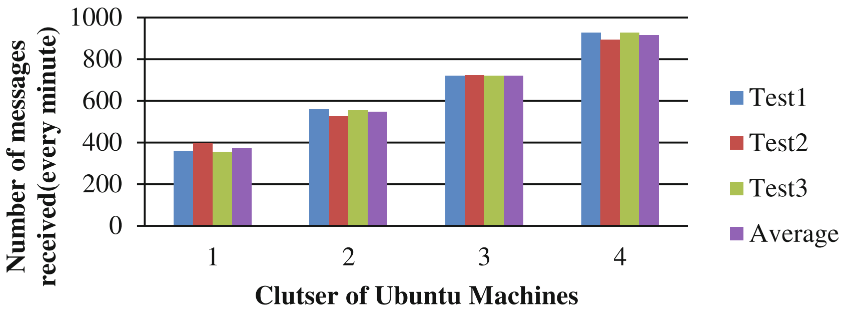

Fig. 2 Evaluation of throughput

number of worker nodes increase. Thus, by implementing a minimal of 4 node cluster for our framework we can collect a variety of tweets from each node and reach up to the benchmark of collecting millions of tweets per day.

Here, the scalability of the framework is evaluated by altering the number of nodes and vCPUs. Series of five tests were conducted and the average is taken to discuss results. As the results in Fig. 3 show raising the number of workers increases the performance of sentiment component dramatically. When the number of virtual CPUs cannot satisfy the computing requirements of the component, increasing number of workers cannot enhance the performance of the component, it may even decrease the number of output messages.

For example, comparing the first two rows of Fig. 3, the number of workers is added from 1 to 2, the average outputs decrease from 589 to 381 messages per minute. By comparing the first row to the third row, which keeps the same workers and increases the number of virtual CPUs, the average message output increases from 589 to 748. We can conclude that one virtual CPU cannot cover the computing resources of one Storm sentiment worker. In the third and fourth row, we increase the number of workers, with 4 virtual CPUs, the average output increases from 748 to 1184 per minute. Likewise, in rows five and six, with six virtual CPUs, increasing worker number raises the output from 1454 to 1654 per minute. When computing resources are adequate, increasing the number of workers leads to increased output as well. As the system applies Twitter 4j to collect data, whose average data fetching rate is 1071 messages per minute, our system can completely process all collected messages, with more than four virtual CPUs and two worker threads.

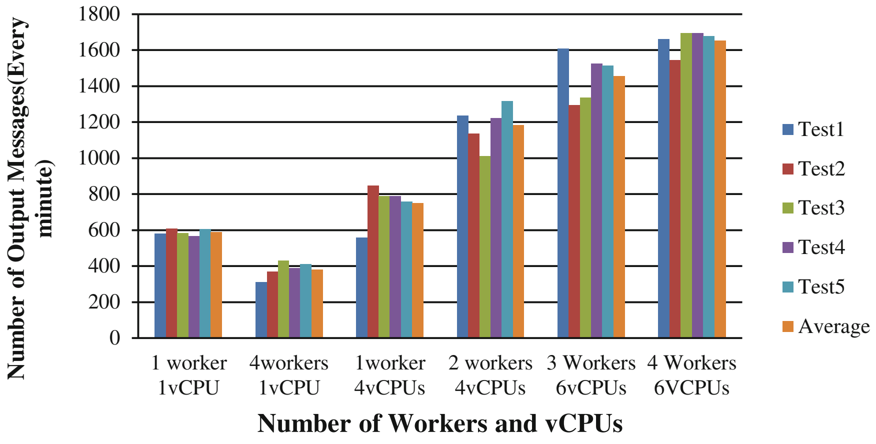

Fig. 3 Evaluation of scalability

5 Conclusion

This research work has proposed a distributed real-time Twitter sentiment analysis and visualization framework by implementing a novel algorithm for SA called EPS. The whole framework is implemented in the form of three major modules which are easily reusable. Apache Storm is a core part of the framework which makes our system distributed and provides real-time stream processing capabilities. Evaluation results prove that our framework provides best results of throughput and scalability. This proves that our system can scale well with real-time big data analytics. Future research directions include experimenting with other data resources like Facebook, Instagram, etc.

References

1. Agarwal, A., et al.: Sentiment analysis of twitter data. In: Proceedings of the Workshop on Languages in Social Media. Association for Computational Linguistics (2011)
2. Ritter, A., Clark, S., Etzioni, O.: Named entity recognition in tweets: an experimental study. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics (2011)
3. Khuc, V.N., et al.: Towards building large-scale distributed systems for twitter sentiment analysis. In: Proceedings of the 27th Annual ACM Symposium on Applied Computing. ACM (2012)
4. Pang, B., Lee, L., Vaithyanathan, S.: Thumbs up?: sentiment classification using machine learning techniques. In: Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing, vol. 10. Association for Computational Linguistics (2012)
5. Balamurali, A.R., Joshi, A., Bhattacharyya, P.: Harnessing wordnet senses for supervised sentiment classification. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics (2011)
6. Chang, C.C., Lin, C.J.: LibSVM: a library for support vector machines. ACM Trans. Intell. Syst. Technol. (TIST) **2**(3), 27 (2011)
7. Garg, N.: Apache Kafka. Packt Publishing Ltd (2013)
8. Jain, A., Nalya, A.: Learning Storm. Packt Publishing (2014)
9. http://www3.nd.edu/mcdonald/Word_Lists.html
10. http://www.cs.uic.edu/