Intern. J. Fuzzy Mathematical Archive
Vol. 6, No. 2, 2015,149-159
ISSN: 2320 –3242 (P), 2320 –3250 (online)
Published on 22 January 2015
www.researchmathsci.org

International Journal of
Fuzzy Mathematical
Archive

Parallel Implementation of Big Data Pre-Processing Algorithms for Sentiment Analysis of Social Networking Data

***V.Jude Nirmal*[1] and *D.I. George Amalarethinam*[2]**

[1]Department of IT, St. Joseph's College (Autonomous), Tiruchirappalli, Tamil Nadu India. e-mail: nirmaljude6@gmail.com
Corresponding Author

[2]Department of Computer Science, Jamal Mohamed College, Tiruchirappalli Tamil Nadu, India. e-mail: di_george@ymail.com

Received 10 November 2014; accepted 30 November 2014

Abstract. Sentiment or Emotion analysis of social networking data involves a lot of data pre-processing. Due to the huge volume, the highly unstructured nature of the data and the enormous rate at which it is being generated, parallel implementations of pre-processing algorithms becomes important. Pre-processing includes string to vector conversion, elimination of trivial words and symbols if any, frequency mapping etc. In this work we implement and compare parallel algorithms for pre-processing of social networking data (Twitter feeds) based on GPGPU (NVidia CUDA C) and Hadoop Map Reduce Architectures. The effectiveness of various tools from the Unix Command Line to the GPGPUs and Hadoop Map-Reduce has been discussed in detail. This work compare and contrast the benefits and drawbacks of GPGPU and Hadoop Map-Reduce and conclude by projecting the effectiveness and drawbacks of using parallel algorithms for effective and timely pre-processing of data.

Keywords: Sentiment analysis, opinion mining, text preprocessing, twitter dataset, map-reduce, GPGPU

1. Introduction

With the increasing importance of social media information in every domain of today's digital age from algorithmic trading and product recommendations to politics, there is a tremendous amount of research work going on in the field of sentiment analysis and opinion mining which is taking us leaps and bounds with the advent of Big Data platforms and tools. The amount of data that could be gathered, processed and stored cheaply and effectively is increasing at an exponential rate with the advent of Hadoop and other related massively parallel platforms and tools. Our work aims at studying the importance of pre-processing in the age of big data where storage and processing of unstructured data is as simple as processing structured data. So why do we have to pre-process the data if Hadoop and other big data tools support handling unstructured data effectively? If required what kind of pre-processing are we talking about and how different it is from the pre-processing that we do in a regular KDD process? What kind of

tools work well in such a scenario and how it is done effectively on such a huge volume of data? Since most of the tools in the Hadoop Ecosystem fundamentally works on the basis of the Map Reduce paradigm (which is a batch processing model), how well do they handle the pre-processing of data that is arriving at a faster rate, like Twitter feeds or posts from Facebook users? How do we handle the velocity part of the Big Data problem? Are the tools of the past no longer valid for these purposes due to the huge volume, variety and velocity of data? These are some of the questions that we are trying to answer. The objective of this work is to identify the best framework or set of tools to pre-process the data from a social networking site like twitter. Even though most of the algorithms for mining big data are found to be supporting unstructured data and also found to be robust to variations in data formats and structure, we stress the importance of pre-processing data as its advantages are found to be many-folded. First it lets us understand the unstructured data that we are dealing with in a better way. Second it helps in dimensionality reduction thereby eliminating a lot of unnecessary features from being handled and in some cases making In-Memory processing of data possible resulting in a huge reduction of I/O overhead. Third it allows to answer the Value and Veracity part of the Big Data paradigm. Finally it allows us to fine tune the data model according to the processing requirements making it much more reliable and accurate. The research paper is organized as follows. The second section (Section 2) deals with the review of literature analyzing the existing techniques for pre-processing of social networking data especially, twitter feeds. The next section (Section 3) discusses the various pre-processing techniques and the requirements for processing social media data and what kind of output is being expected from such a pre-processing framework and also discusses the importance of stop words and emoticons as a special case with respect to twitter data. Section 4 discusses about the Stanford Twitter Dataset, the Twitter API and provides an overview of obtaining feeds using the Twitter API. Section 5 explains in detail the various parts of the pre-processing framework architecture and about the tools and techniques that are found to be suitable for the same. Section 6 deals with the various platforms and tools that could be employed to handle the pre-processing of twitter data and a discussion about their possible effectiveness from a theoretical standpoint. It also deals about the experimental setup and the various parameters considered during the pre-processing phase. Section 7 deals with results and discussion. Section 8 concludes the paper providing a summary of the work and highlighting some future research directions.

2. Literature review

Three common approaches to sentiment classification exists in literature, namely, machine learning, lexicon based methods and linguistic analysis. Each method has its own pros and cons. This section discusses literature that discusses each of these methods and some combination methods that claim better efficiency. Further, here we also discuss methods that provide efficiency using hardware architectures rather than the software counterparts. A GPU (Graphics Processing Unit) based string matching mechanism is presented by Kouzinopoulos et al [1]. This method exploits the processing power and memory bandwidth of GPUs that have recently emerged into the processing market. It uses CUDA (Compute Unified Device Architecture) to perform the computations. A comparative analysis of Naive, Knuth-Morris-Pratt, Boyer-Moore-Horspool and Quick-

Search string matching algorithms were performed on Y.pestis, B.anthracis and BAC reference sequences for different pattern sizes. A stream based active learning method is proposed by Jasmina et al [2], which utilizes sentiment analysis for the financial domain. This method has its basis on the Twitter feeds. It uses these feeds for financial forecasting by performing sentiment analysis. The best text pre-processing setting for training the SVM (Support Vector Machine) is proposed here. This labels entities, by replacing the actual words with certain labels as the pre-processing phase. Other pre-processing techniques such as tokenization, N-gram construction and TF-IDF were used for the construction of the feature vector. F-Measure is used to determine the best pre-processing setting. A similar method analyzing the political scenario of the region analyzing the tweets generated from the region is presented by Rill et al. [22]. Data collection was performed in Germany during the period of parliamentary election 2013. This method works on the basis of twitter hashtags also known as sentiment hashtags. An N-gram based feature selection mechanism, that uses emoticons for lexical analysis is proposed by Read et al. [3]. Similar methods have been found throughout various literatures and uses SVM as its base Classifier [4-11]. Speriosu et al. [8] uses the maximum entropy as its classification technique. Ensemble based techniques [12-16] are also prevalent. These methods use a combination technique to determine the feature vector. An active learning method for sentiment analysis on data streams is presented by Janez, et al. [17]. The workflow implementation for this method was performed in Cloud Flows platform. In general, the sentiment analysis methods do not work on streams, they usually perform batch processing. But in some instances, processing data on the go might be necessary. Cloud Flows is one such platform that supports streaming. Janez, et al. [17] used SVM for the process of pre-processing. The active learning here is enforced by the introduction of a labelling strategy. The initial step is to manually request for the Oracle (administrator/user) to provide labels to a set of samples. These samples are then used for performing the future decisions. Every time the system encounters a new set of data; the Oracle is requested for labelling it. The model is then updated and performs accordingly.
The process of sentiment analysis is not only carried out with English as the language. It has also been experimented with other languages. The variation here is to use a different ontology for creating the model. The sentiment mining model that uses Czech language for performing sentiment analysis on Czech social media is presented by Ivan et al[18]. A similar method that uses the Facebook data of the Spanish to determine the sentiment polarity is presented in Alvaro et al, [19]. A similar research for classification of text according to polarity was performed in Japanese language. Due to the unavailability of a standard corpus in Japanese, they have used the largest available text corpus. Testing was carried out using text from random participants and the results were evaluated. A mechanism that classifies tweets according to the emoticons in them is presented by Arturo et al. [20]. It classifies the tweets according to the polarity of the emoticons. Moreover, the polarity is based on real values from -1 to 1 rather than classifying them into two or three broad categories. A positive value represents a positive polarity, while negative values indicate negative polarity, and 0 indicates that the tweet is neutral.

3. Pre-processing twitter data

Pre-processing of Twitter data is completely different from the pre-processing done for KDD process in regular text datasets. Tweets are very small by themselves (atmost 140

characters) and because of this users tend to use a different slang or acronyms for almost every common word in English. Usually we eliminate words that are less than 3 characters and greater than 16 characters, but that will not be applicable to tweets. Also most words that are available as part of stop words list may have more importance in a tweet than in a standard text. Say for example the word "but" is very common in English and could be removed as a stop word but for a product review the appearance of this word differentiates between a 4 star and 5 star rating (Say I am satisfied with the product but …). So we will have to determine the uncommonly common words and then only the list of stop words could be decided.

Positive Emoticons	Negative Emoticons
o n n *-* *O* * *	
:P :D :d :p	:(;(:'(;'(
;P ;D ;d ;p	=(={):);
:-) ;-) :-) ;-)	)':)';)= }=
:<) :>) ;>) =)	;-{{ ;-{ :-{{ :-{
=} :) (: ;)	:-(;-(:-(;-(
(; :} {: ;}	:,(:'{
{; :]	[: ;]
[; :') ;') :-3	
;-3 :-x ;-x :-X	
;-X :-} ;-} :-]	
;-] :-.)	
^_^ ^-^	

Table 1: List of common positive and negative emoticons used in Twitter

Regular text mining approaches eliminates punctuations and symbols but in the case of tweets we have to consider Emoticons, which express sentiment polarity more effectively than words. So this approach cannot remove all the symbols instead it must search for an emoticon based on an In-Memory Dictionary or Hash Table and then decide about removal of symbols if they are not a part of an Emoticon. Emoticons may contain 2 characters, 3 characters or 4 characters as shown in Table 1. So this approach had to concentrate on writing regular expressions to filter them out from a bunch of regular symbols. It also embeds the Word Net corpus to find the Synonyms and Antonyms to effectively identify the words with positive and negative polarity.

3.1. The stanford twitter dataset and twitter api

The dataset that we have used is the Stanford Twitter Corpus which contains 1.6 million tweets. It is part of the Sentiment 140 project [23]. The dataset contains tweets that are classified as positive, negative and neutral based on the presence of emoticons rather than doing it manually. The format of the dataset is as follows [23]: It contains 6 fields as follows.

0 - the polarity of the tweet
1 - the id of the tweet
2 - the date of the tweet
3 - the query. If there is no query, then this value is NO_QUERY.

4 - the user that tweeted
5 - the text of the tweet

The dataset was collected using the previous generation Twitter Search API based on keyword search. The dataset, though classifies tweets based on emoticons, is clear of all the emoticons and unwanted text and is hence preprocessed to a certain extent. So in order to get raw Twitter data we have to depend on the Twitter API.

Twitter provides access to its data by means of three different services. GNIP [24] which provides access to the full archive and also the current generation Twitter data for a cost. The REST API provides a low latency access to current Twitter data but with severe limitations on the data access controlled by the Twitter authentication process. Twitter Streaming API is another option to access the data. Though it is slow, it doesn't have the limitations of the REST API [24]. Streaming APIs are available in three different type as follows: Public Streams that provides access to the public data flowing through Twitter. Suitable for following specific users or topics, and data mining. User Streams or Single-user streams, containing roughly all of the data corresponding with a single user's view of Twitter. Site Streams or The multi-user version of user streams. Site streams are intended for servers which must connect to twitter on behalf of many users. This approach uses the Streaming APIs Public Streams for accessing the tweets. [24].

3.2. Architecture of the proposed pre-processing framework

The proposed a framework for effective pre-processing of Twitter Feeds is as shown in Figure 1. The framework consists of four major subdivisions from Acquisition of data, followed by two phases of Pre-processing and then Evaluation.

3.2.1. Obtaining twitter data

We have used the Streaming Twitter API and the public Streams version of it. All the public data flowing through Twitter currently could be collected using this Public Streams API. Though it is a low latency process it provides considerable amount of data to work on when run continuously for several hours. Twitter Feeds are accessed using the Streaming API and then tweets related to the query passed is obtained and the content is piped in to the next phase of the pre-processing framework.

3.2.2. Pre-processing Phase 1

In this phase, the tweets are available as text data and each line contains a tweet. Initially we clean up or remove retweets as that will induce a bias in the classification process. Also we remove the links followed by Hashtags which represents the users. We then process tweets that contains positive emoticons and those that contain negative emoticons. Also there is a possibility of a tweet being neutral. We need to remove the punctuations and other symbols that doesn't make any sense as it may result in inefficiencies and may affect the accuracy of the overall process. Also we need to remove tweets that contain both a positive and negative emoticons as it may cause a lot of confusion as far as automatic classification is considered. This results in a clean Twitter Text Corpus which is then sent to the second phase of pre-processing.

3.2.3. Pre-processing Phase 2

In this phase, we tokenize the tweets (split them into individual words of importance). Once tokenized, we perform one of the most important part of the text pre-processing process, which is normalization. Here we eliminate repeat letters from words that are put in to provide stress or a better context to the tweet. Also we normalize the expressions usually

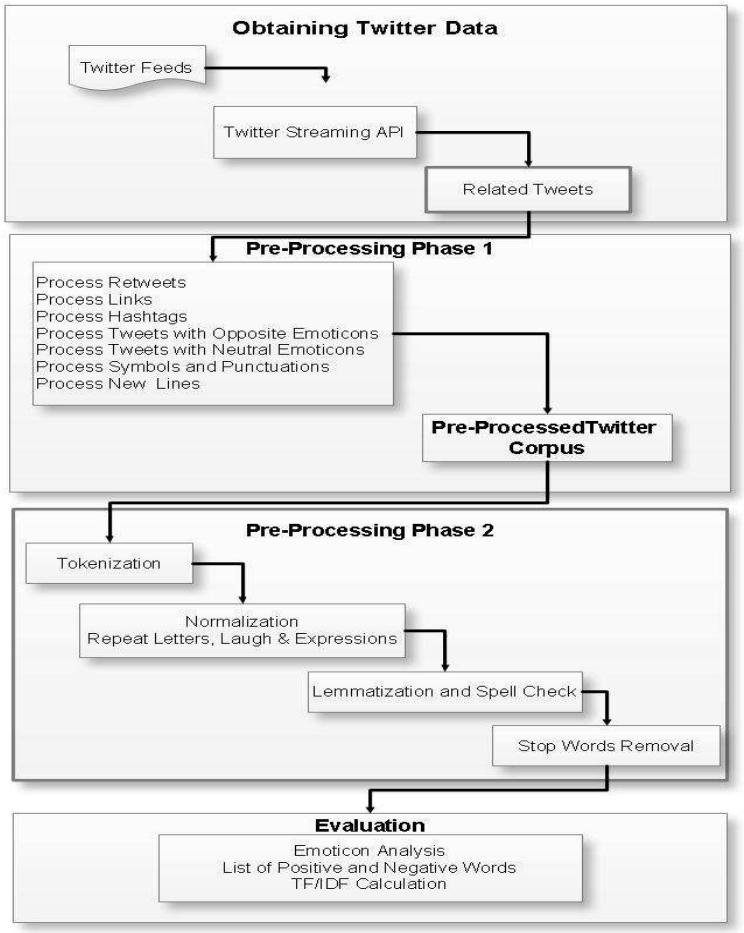

Figure 1: Architecture of the proposed pre-processing framework

used to express laugh, sorrow etc. which may contain a series of repetitive characters. Then comes the process called lemmatization where the core part of the word alone is considered. It is different from stemming where stemming is just structural, lemmatization is contextual and also takes into account the synonyms and antonyms of a given word. After this process, we remove the stop words as provided by standard stop word corpus. Now the data is ready for evaluation and is sent to the next phase.

3.2.4. Evaluation

In this phase we perform the word count followed by calculation of TF/IDF. This phase involves too much of computation and could be considered as a number crunching phase in addition to being Memory or I/O bound like the previous phases. Here we analyze the list of Emoticons and their polarity, TF/IDF and also the list of positive and negative words.

3.3. Experimental setup

Experimental setup includes three totally different configurations to test the suitability of various tools ranging from Linux/Unix command line tools to GPGPUs and Hadoop Map-Reduce based tools. A High End Apple MacBook Pro running Mac OSX Mavericks with a Quad-Core i7 Processor running at 2.5 GHz, 16 GB DDR3 RAM @ 1600MHz and 6 MB L3 Cache followed by a 128 MB L4 Cache and a 512 GB SSD Drive. A High End Dell Precision T7600 Workstation with dual Intel Xeon Octa-Core Processors each running at 2.7GHz, 32GB DDR3 RAM, 20MB L3 Cache Memory with 4 * 600GB 7200RPM HDD. Finally a Hadoop cluster containing 10 High End PCs and a two High End Servers acting as Master Nodes. Master Nodes are Dell R910 Servers with Four Intel Xeon Octa-Core Processors running at 3.5 GHz, 64GB DDR3 RAM, 30 MB L3 Cache Memory per Processor and 6 * 600GB SAS 10K RPM Drives in RAID0 configuration. Data Nodes are Dell OptiPlex 9020 PCs with Quad-Core i7 processors running at 2.7 GHz, 8 GB DDR3 RAM and 1TB 7200 RPM HDD. It is obvious that for almost 90% of the first two phases and for a considerable part of the third phase, using a high end laptop is found to be much more flexible and effective than going in for a workstation or a cluster. This is due to the fact that the data handled during these phases is found to fit In-Memory in such a laptop and hence very effective and fast compared to the GPGPU based approach where the data has to be transferred from the CPU HDD to the CPU memory and then from the CPU memory to the Device memory and the results must be copied back to the CPU memory from the Device memory before being persisted to the hard disk. The cluster on the other hand is basically used for Map-Reduce based batch processing and the overhead with the cluster is that all the intermediate data needs to be persisted to the HDD and then retrieved back for the next phase. Since the first two phases involves a lot of process being performed on the same data again and again just doing a small amount of processing each time, it is very difficult to run the jobs on a GPU or on a cluster as it will involve a lot of Memory or Disk I/O overhead.

Laptops running on Unix/Linux systems provides the flexibility to have the data In-Memory as opposed to persisting them to the HDD each time. We can use the pipe option and pass the data through various text and regular expression processing tools available natively as part of the operating system making it simple and effective. With GNU parallel, we can make use of the inherent parallelism to the maximum extent possible. The pipes and regular expression tools allows us to manage the workflow effectively and is possible to automate the tasks with simple scripts rather than worrying about learning and writing complex Map-Reduce jobs using Java or one of the Hadoop Ecosystem tools.

But as we can clearly see a part of the third phase and the complete final phase involves a lot of number crunching work and is found to be suitable to make use of the CUDA GPGPU programming model. The calculation of TF/IDF based on word

frequencies could be done in the massively parallel architecture with much ease in a fraction of the overall time taken by a single CPU based system. The importance of Hadoop Map-Reduce based cluster and the Hadoop Ecosystem tools could not be undermined at these phases. They prove to be effective and is also massively parallel in comparison to the simple CPU level multi-threading which we obtain from an individual laptop.

3.4. Results and discussion

It could be clearly seen from Table 2 and Figure 2 that the importance of the Hadoop Map-Reduce framework and the need for a distributed cluster could be appreciated only when huge amount of data is handled. As far as a million tweets are concerned, the performance of a Laptop is considerably acceptable considering the cost difference and the overhead of maintaining a cluster. Also in the final phase involving TF/IDF calculations we found that the Memory and I/O overhead is still huge on a GPU as we need to handle the words as such or we need to encode them using an id increasing the overhead. So the actual time taken is too high and is not comparable to what we obtain from the command line tools. Also we have limitations on memory size on the GPU in comparison to the CPU memory. Even high end GPUs have a DRAM size limit of 5 GBs and along with this limitation we have one more problem that, no data-structure like a dictionary or list is available on the GPU programming model and hence limits us in the TF/IDF calculation requiring too much of data transfer overhead.

Dataset	Time (Laptop) in Seconds	Time (Hadoop) in Seconds
1 Book	0.101	11
10 Books	0.723	12
25 Books	4.486	12
50 Books	9.685	13
100 Books	21.95	15
200 Books	52.334	24
400 Books	106.756	36
1.6 Million Tweets	61.792	29

Table 2: Performance of Word Count

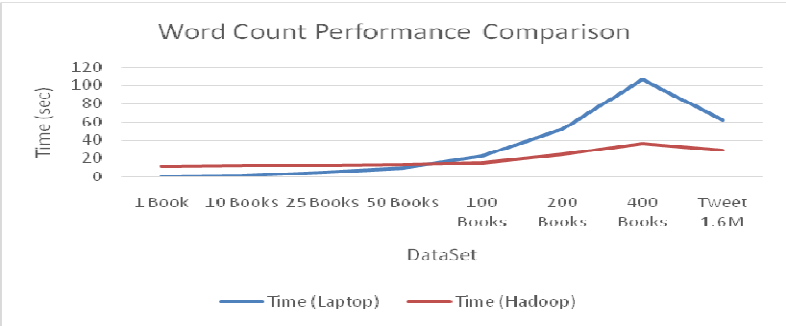

Figure 2: Word count performance comparison

It is found that the problem of TF/IDF calculation is not so much of a divide and conquer nature and hence not suitable for the Map-Reduce Paradigm. Since we need the total number of documents and the number of documents in which a word is occurring in order to calculate the IDF and the word count done on each document separately for the term frequency part, we find that the GPGPU and the Hadoop Map-Reduce cluster not so very suitable as they both lack in memory data structures and due to that it requires several runs of Map-Reduce to calculate all of those elements individually and then running them on a single reducer to calculate the final output. We performed the TF/IDF calculations on the command line using a Laptop in 58.056 Seconds.

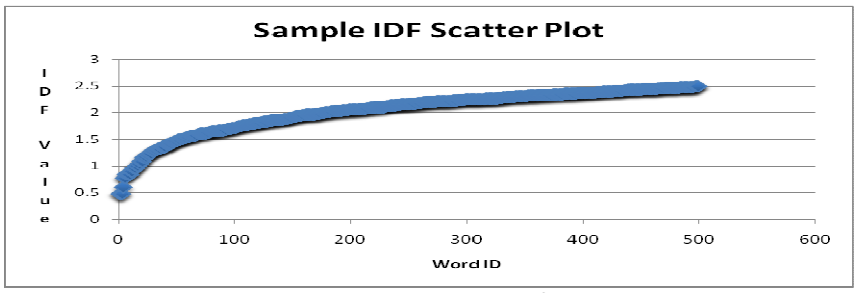

Figure 3: Sample scatter plot for IDF values

Figure 3 shows the sample scatter plot for IDF values of the first 500 words (sorted in ascending order) out of the 785687 words obtained from the 1.6 million tweets in our dataset. The scatter plot shows TF/IDF values for words that occur frequently and those that occur in-frequently and how TF/IDF eliminates the bias.

8. Conclusion and future research directions

Considering the data volume and velocity that we handled including the data feeds from the Twitter API, the Stanford Twitter Dataset and the Books from Gutenberg project, we find that Linux/Unix command line processing tools (usually ignored due to their simplicity) are much suitable for the first two phases of the pre-processing process as opposed to GPUs or Hadoop Big Data environment and that is attributed to the fact that the data fits perfectly In-Memory (mostly fits in the RAM of the System 16GB to 32GB) and due to the flexibility of the tools in handling data streams. We need to verify the same on huge text corpus including Project Gutenberg (containing 45000 Books, we used only 400 books) and Wikipedia text corpus (35 GB+) and New York Times Articles through their API in which case the command line utilities may fail requiring GPUs/Hadoop Map-Reduce architecture due to the sheer volume of data that wouldn't fit In-Memory. For the third and final phases which involves a lot of number crunching in terms of TF/IDF calculations we still do find that a GPGPU based massively parallel approach or the Hadoop Map-Reduce approach not very suitable as we need in memory data-structures to carry out the process effectively. Also the Hadoop Map-Reduce based approach, though not suitable for streaming due to its batch processing nature, is found to be delivering stable performance when huge data is involved. Also we find that using a search engine based approach would be much suitable as it would provide us with the necessary tools to identify the importance of emoticons and certain stop words in the Twitter Text Corpus. With the Apache Lucene Project and the release of Apache Solr

Enterprise Search Engine, it is possible for a common man to utilize the effectiveness of a search engine which is previously too costly involving millions of dollars. We are also considering the ELK (Elasticsearch based on Lucene, LogStash and Kibanna) stack for search and for visualization of the results during the Sentiment Analysis process.

REFERENCES

1. C.S.Kouzinopoulos and K.G.Margaritis, String Matching on a multicore GPU using CUDA, Informatics, 2009. PCI '09. 13th Panhellenic Conference, 2009, pp. 14 – 18.
2. Jasmina Smailović́a, Miha Grčara, Nada Lavrača and Martin Žnidaršiča, Stream-based active learning for sentiment analysis in the financial domain, *Information Sciences*, 285 (2014) 181-203.
3. Jonathon Read, Using emoticons to reduce dependency in machine learning techniques for sentiment classification, Proceedings of the ACL Student Research Workshop, Association for Computational Linguistics, 2005. pp. 43-48
4. Alec Go, Richa Bhayani, and Lei Huang., Twitter sentiment classification using distant supervision, CS224N Project Report, Stanford 2009, 1-12.
5. Dmitry Davidov, Oren Tsur, and Ari Rappoport., Enhanced sentiment learning using twitter hashtags and smileys, Proceedings of the 23rd International Conference on Computational Linguistics: Posters, Association for Computational Linguistics, 2010.
6. Lei Zhang, Riddhiman Ghosh, Mohamed Dekhil, Meichun Hsu and Bing Liu, Combining lexicon-based and learning-based methods for Twitter sentiment analysis, HP Laboratories, Technical Report HPL-2011, 89.
7. Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca Passonneau, Sentiment analysis of twitter data, Proceedings of the Workshop on Languages in Social Media. Association for Computational Linguistics, 2011.
8. Michael Speriosu, Nikita Sudan, Sid Upadhyay and Jason Baldridge, Twitter polarity classification with label propagation over lexical links and the follower graph, Proceedings of the First workshop on Unsupervised Learning in NLP, Association for Computational Linguistics, 2011.
9. Hassan Saif, Yulan He and Harith Alani, Semantic sentiment analysis of twitter, The Semantic Web–ISWC, Springer Berlin Heidelberg, 2012, pp. 508-524.
10. Xia Hu, Lei Tang, Jiliang Tang and Huan Liu, Exploiting social relations for sentiment analysis in microblogging, Proceedings of the sixth ACM international conference on Web search and data mining, ACM, 2013.
11. S.M.Mohammad, S.Kiritchenko and X.Zhu., NRC-Canada: Building the state-of-the-art in sentiment analysis of tweets, arXiv preprint, 2013.
12. J.Lin and A.Kolcz., Large-scale machine learning at twitter, Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, ACM, 2012.
13. Sam Clark and Richard Wicentowski, SwatCS: Combining simple classifiers with estimated accuracy, Second Joint Conference on Lexical and Computational Semantics, Vol. 2, Seventh International Workshop on Semantic Evaluation (SemEval), 2013, pp. 425–429.
14. Carlos Rodr´ıguez-Penagos, Jordi Atserias, Joan Codina-Filba, David Garc´ıa-Narbona, Jens Grivolla, Patrik Lambert, Roser Saur´, and FBM: Combining lexicon-based ML and heuristics for Social Media Polarities, Second Joint Conference on

Lexical and Computational Semantics, Seventh International Workshop on Semantic Evaluation (SemEval 2013), Atlanta, Georgia, June 14-15, 2013, pp. 483–489.
15. A.Hassan, Ahmed Abbasi and D.Zeng., Twitter Sentiment Analysis: A Bootstrap Ensemble Framework, Social Computing (SocialCom), 2013 International Conference on. IEEE, 2013.
16. da Silva, F.F.Nádia, E.R.Hruschka and E.R.Hruschka, Tweet sentiment analysis with classifier ensembles, Decision Support Systems, 2014.
17. Janez Kranjca, Jasmina Smailovića, Vid Podpečana, Miha Grčara, Martin Žnidaršiča and Nada Lavrača, Active learning for sentiment analysis on data streams: Methodology and workflow implementation in the ClowdFlows platform, Information Processing & Management, 2014.
18. I.Habernal, Tomáš Ptáček and J.Steinberger, Supervised sentiment analysis in Czech social media, *Information Processing & Management*, 50 (2014) 693-707.
19. A.Ortigosa, José M.Martín and R.M.Carro, Sentiment analysis in Facebook and its application to e-learning, *Computers in Human Behavior*, 31 (2014) 527-541.
20. Arturo Montejo-Ráez, Eugenio Martínez-Cámara, M. Teresa Martín-Valdivia and L. Alfonso Ureña-López, Ranked WordNet graph for Sentiment Polarity Classification in Twitter, *Computer Speech & Language*, 28 (2014) 93-107.
21. Michal Ptaszynskia, Rafal Rzepkab, Kenji Arakib and Yoshio Momouchi, Automatically is annotating a five-billion-word corpus of Japanese blogs for sentiment and affect analysis, *Computer Speech & Language*, 28 (2014) 38-55.
22. Sven Rilla, Dirk Reinela, Jörg Scheidta, and Roberto V. Zicarib, PoliTwi: Early detection of emerging political topics on twitter and the impact on concept-level sentiment analysis, *Knowledge-Based Systems*, 69 (2014) 24-33.
23. www.sentiment140.com, Accessed on 15th October, 2014.
24. https://dev.twitter.com, Accessed on 15th October, 2014.