Sentiment Analysis on Multilingual Tweets using Big Data Technologies

Rodrigo Martínez-Castaño, Juan C. Pichel and Pablo Gamallo[1]

Abstract— **In this paper a new parallel system for sentiment analysis on multilingual tweets based on Big Data technologies is presented. On the one hand, our sentiment classifier performs as well as other state-of-the-art classifiers considering tweets written in different languages. On the other hand, our system is capable of processing millions of tweets in short times taking advantage of Big Data processing and parallel architectures, showing a good scalability in all the considered scenarios.**

Keywords— **Sentiment analysis, Big Data, Performance, Twitter, Hadoop.**

I. Introduction

Sentiment Analysis consists in finding the opinion (e.g. positive, negative, or neutral) from text documents such as movie reviews or product reviews. Opinions about movies, products, etc. can be found in web blogs, social networks, discussion forums, and so on. Companies can improve their products and services on the basis of the reviews and comments of their costumers. Recently, many works have stressed the microblogging service Twitter. As Twitter can be seen as a large source of short texts (tweets) containing user opinions, most of these works make sentiment analysis by identifying user attitudes and opinions toward a particular topic or product. The task of making sentiment analysis from tweets is a hard challenge. On the one hand, as in any sentiment analysis framework, we have to deal with human subjectivity. Even humans often disagree on the categorization on the positive or negative sentiment that is supposed to be expressed on a given text. On the other hand, tweets are too short text to be linguistically analyzed, and it makes the task of finding relevant information (e.g. opinions) much harder.

Useful conclusions can only be extracted when huge amounts of text or documents are analyzed. However, standard solutions cannot handle gigabytes or terabytes of text data in reasonable time. In this way, professionals demand scalable solutions to boost performance of the sentiment analysis process. To address this challenge we propose to take advantage of parallel architectures using Big Data technologies.

In this paper a new parallel system for sentiment analysis on multilingual tweets based on Big Data technologies is introduced. The goal of our system is twofold. First, the sentiment classifier should perform as well as other state-of-the-art classifiers considering tweets written in different languages. For this reason a thorough evaluation was carried out analyzing Spanish and English tweets. We must highlight that our classifier took part in two different sentiment analysis contests showing a good behavior with respect to other approaches. Second, millions of tweets should be processed in short times. With this objective in mind tweets are analyzed in parallel taking advantage of a Hadoop [1] cluster which decreases the processing times noticeably. Performance results demonstrates the benefits in terms of scalability of our proposal.

II. Background & Related Work

A. Big Data processing

MapReduce [2] is a programming model introduced by Google for processing and generating large data sets on a huge number of computing nodes. A MapReduce program execution is divided into two phases: *map* and *reduce*. The input and output of a MapReduce computation is a list of key-value pairs. Users only need to focus on implementing map and reduce functions. In the map phase, map workers take as input a list of key-value pairs and generate a set of intermediate output key-value pairs, which are stored in the intermediate storage (i.e., files or in-memory buffers). The reduce function processes each intermediate key and its associated list of values to produce a final dataset of key-value pairs. In this way, map workers achieve data parallelism, while reduce workers perform parallel reduction. Note that parallelization, resource management, fault tolerance and other related issues are handled by the MapReduce runtime.

Apache Hadoop [1] is the most successful open-source implementation of the MapReduce programming model. Hadoop consists, basically, of three layers: a data storage layer (HDFS), a resource manager layer (YARN), and a data processing layer (Hadoop MapReduce Framework). HDFS is a block-oriented file system based on the idea that the most efficient data processing pattern is a write-once, read-many-times pattern. For this reason, Hadoop shows good performance with embarrassingly parallel applications requiring a single MapReduce execution (assuming intermediate results between map and reduce phases are not huge), and even for applications requiring a small number of sequential MapReduce executions [3]. However, Hadoop performs poorly when applications do not fit into one of the previous workflows. For example, an iterative algorithm can be expressed as a sequence of multiple MapReduce jobs. Since different MapReduce jobs cannot shared data directly, intermediate results have to be written

[1]Centro de Investigación en Tecnoloxías da Información (CITIUS), Universidade de Santiago de Compostela, Spain, e-mail: `{rodrigo.martinez, juancarlos.pichel, pablo.gamallo}@usc.es`

to disk and read again from HDFS at the beginning of the next iteration, with the consequent reduction in performance.

B. Sentiment analysis

There exist two types of approaches for sentiment analysis: Machine learning classification and lexicon-based strategy. Machine learning methods use several learning algorithms to determine the sentiment by training on a known dataset. Many of them rely on very basic classifiers, e.g. Naive Bayes [4] or Support Vector Machines [5]. They are trained on a particular dataset using features such as bag of words or bigrams, and with or without part-of-speech (PoS) tags. It is admited that very basic language models based just on bag of words perform reasonably well [6]. The lexicon-based technique involves calculating sentiment polarity for a text using dictionaries or lexicons of words. The lexicon entries are annotated with their semantic orientation: polarity (positive, negative, or neutral) and strength [7].

To deal with short messages such as tweets and SMS, the state-of-the-art systems are based on machine learning techniques using as features polarity lexicons [8], [9]. Our strategy also makes use of polarity lexicons to enrich the set of features of the classifier. Most recent approaches to sentiment analysis on short messages and social media are endowed with rich linguistic information such as shallow syntactic structures [10] or syntactic dependency trees [11]. Following the tendency to use knowledge-rich linguistic features, our approach will be provided with shallow syntactic information to detect polarity shifters (e.g., negation markers).

We can find in the literature several interesting works which applied sentiment analysis to different document or text sources using Big Data technologies, for example, movie reviews [12] and hotel reviews [13]. Another authors analyzed tweets in real time [14]. Finally, some researchers have focused on the impact in the performance of different ways of distributing virtual machines (workers) on a cloud environment using sentiment analysis as case study [15].

III. Architecture of the System

The system is composed of three main modules illustrated in Figure 1. The first one is the Tweet Mining Module, which is the responsible of collecting the tweets for the future processing. This module has two components that will be explained in detail in the next section. The second module is, strictly speaking, the MapReduce application. This module is in charge of processing the stored tweets from a HBase[1] table. HBase is a non-relational and distributed column-oriented database built on top of HDFS. The mappers receive the tweets from HBase and process them through the sentiment analysis classifier if they match the query terms. Note that there are two reducers, one is responsible of summarizing the number of successfully processed tweets and the other calculates the average positivity ratio and the number of matches for a particular query. This module is described in Section V.

The application can be executed through a web interface as shown in Figure 2, which corresponds to the third module of the system. The interface has two different views. A simple web form (main view, left image) allows to customize the MapReduce jobs that will be launched using two filters: start and end dates (an interval can be established) and the input terms. Several terms can be introduced separated by spaces. The second view shows the obtained result when the MapReduce job finishes (right image in the figure). The web back-end processes the data received from the web form, launches a Hadoop job and reads the outputs from HDFS, showing the result when it is finished.

IV. Tweet Mining Module

The Tweet Mining Module was developed in Java and consists of two components. The main module is a web scraper which acquires tweets directly from the official user interface for web browsers. The second component uses the Streaming API of Twitter with the help of the Twitter4J[2] library.

Twitter streams an arbitrary selection of tweets through the Streaming API. This represents a small fraction of the total volume of tweets published at each moment. The provided stream is the same for every consumer and it is very limited. It is said to provide about the 1% of the total tweets produced in every moment.

The idea to deal with the aforementioned problems is to retrieve tweets making queries to the Search API with common words of a particular language. Queries should be executed very often for each term when using this method, but this is not feasible due to the API limitations. However, these limitations can be bypassed considering web scraping techniques. This is the job of the second and main component of the Tweet Mining Module, which gathers tweets from the official web client making queries directly to `twitter.com/search`.

The module retrieves the HTML code that the web client returns for each query of every selected term and processes it using the HtmlUnit[3] library. Through the web client, Twitter provides 20 tweets every 20 seconds for each query and for every consumer (at most, if available). Their system makes a selection of tweets as the Streaming API does, but, in this case, if the optimal terms are selected, a significant percentage of the total produced tweets for some language could be collected.

An important drawback of this method arises when lots of tweets contain only very common terms (usually there are more than those 20 tweets per term) forcing the loss of tons of tweets. This method

[1] `https://hbase.apache.org/`

[2] `http://twitter4j.org/`

[3] `http://htmlunit.sourceforge.net/`

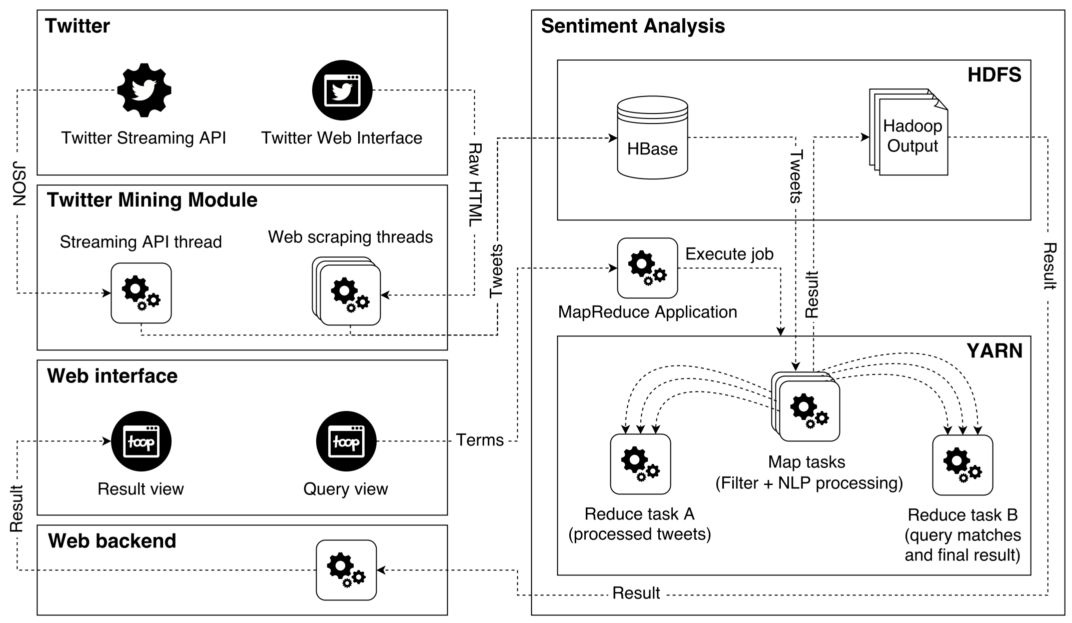

Fig. 1
Architecture diagram of the Sentiment Analysis system.

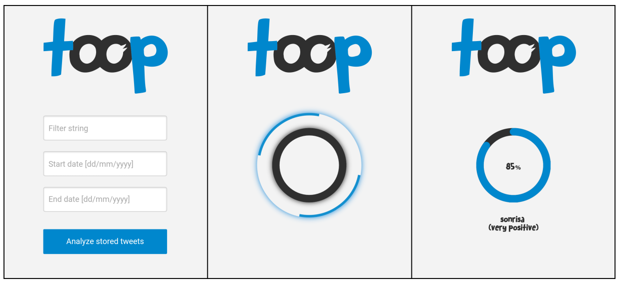

Fig. 2
Web GUI of the Sentiment Analysis system.

gets good results using frequency lists of certain languages and other type of terms like trending ones. This module and the idea behind it pretend to maximize the tweet retrieval at zero-cost.

When this module is executed, one thread is consulting the Streaming API while a configurable number of web scraper threads are retrieving information from the web client with the list of selected terms. In order to launch a mining job, the API tokens of a Twitter application must be set if using the official Streaming API. Other relevant parameters that can be set are the number of scraper threads to launch, the base size of the buffer and the step (in order to distribute the writes into HBase: the buffer of each thread is increased progressively with the step value), the number of laps (times that every web scraper thread consults its assigned terms) and the sleep time (number of seconds that every web scraper thread will pause its execution before starting a new lap).

V. Sentiment Analysis Module

Our approach is based on a Naive Bayes (NB) classifier. NB combines efficiency (optimal time performance) with reasonable accuracy. The main theoretical drawback of NB methods is that it assumes conditional independence among the linguistic features. If the main features are the tokens extracted from texts, it is evident that they cannot be considered as independent, since words co-occuring in a text are somehow linked by different types of syntactic and semantic dependencies. However, even if NB produces an oversimplified model, its classification decisions are surprisingly accurate [16]. To improve the performance of the system, the classifier was enriched with lexicon-based features. Our sentiment analysis system is called *CitiusSentiment*[4], it is multilingual and supports English and Spanish texts. It was implemented in Perl.

A. Strategy and features

Our classifier requires both a simplified annotated corpus of tweets (or short sentences) and a polarity lexicon with both Positive and Negative words. The annotated corpus only contains positive and negative examples of tweets. Neutral tweets are not required. As a result, a basic binary (or boolean) classifier which only identifies both Positive and Negative tweets is trained. In order to detect tweets without polarity (or Neutral), the following basic rule is used: if the tweet contains at least one word that is also found in the polarity lexicon, then the tweet has some degree of polarity. Otherwise, the tweet has no polarity at all and is classified as Neutral. The binary classifier is actually suited to specify the ba-

[4]freely available at `http://gramatica.usc.es/pln/tools/CitiusSentiment.html`

sic polarity between positive and negative, reaching a precision of more than 80% in a corpus with just these two categories.

The training corpus of tweets as well as the analyzed tweets are preprocessed as follows:

- Removing urls, references to usernames, and hashtags
- Reduction of replicated characters (e.g. *looooveeee* → *love*)
- Identifying emoticons and interjections and replacing them with polarity or sentiment expressions (e.g. :-) → *good*)

The features considered by the classifier are lemmas, multiwords, polarity lexicons, and valence shifters. In the following, we describe the polarity lexicons and valence shifters.

Polarity Lexicons: For each language, we built a polarity lexicon with both *Positive* and *Negative* entries from different sources. The English lexicon is the result of unifying the following lexical resources:

- AFINN-111[5] contains $2,477$ word forms, which were lemmatized and converted into $1,520$, positive and negative lemmas.
- Hedonometer [17] contains about $10,000$ frequent words extracted from tweets which were classified as expressing some degree of happiness.
- Hu&Liu list [18] contains over $6,800$ words out of which 5 positive and negative lemmas were selected $5,695$.
- Sentiwordnet-3.0 [19] contains more than $100,000$ entries. We selected a subset of $6,600$ positive and negative lemmas with the highest polarity values.
- Finally, we have built a polarity lexicon with $10,850$ entries by merging the previous ones.

The resources used to elaborate the Spanish lexicon are the following:

- Spanish Emotion Lexicon (SEL) [20] contains $2,036$ words.
- A list of synonyms (ExpandSEL) was automatically extracted by expanding SEL using a dictionary of synonyms.
- A list of polarity words was semi-automatically extracted from the training corpus (CorpusLex).
- Finally, we have built a polarity lexicon with $4,564$ entries by merging the previous ones.

Valence shifters: We take into account negative words that can shift the polarity of specific lemmas in a tweet. In the presented work, we will make use of only those valence shifters that reverse the sentiment of words, namely *negations.* The strategy to identify the scope of negations relies on the PoS (part-of-speech) tags of the negative word as well as of those words appearing to its right in the sequence.

[5]`http://arxiv.org/abs/1103.2903`

B. Integration into a Big Data infrastructure

In order to store the tweets collected by the Tweet Mining Module we have used Apache HBase. The Tweet Mining Module fills a buffer as the tweets are retrieved in such a way that when the buffer is full, all its content is dumped in the HBase corresponding table. Each row of the table has the following fields: a key (it matches the tweet original ID) and several attributes of the tweet stored in a single column family. These attributes are: user ID, nickname of the publisher, language, date of the post and the text of the tweet.

MapReduce (Hadoop) jobs can be launched when the Tweet Mining Module has finished or when it is still working. The MapReduce application performs reads from the HBase table which contains all the stored tweets. The initial and final date, the prefetch value, the table name, the internal task ID (it will determine the output file) and the target terms must be set when starting a job. Every tweet should match with each term to be processed (not necessarily in a contiguous manner).

HBase tables are split into regions and a mapper is launched for every region. While retrieving data from a HBase table, one region is equivalent to a split or map task. The MapReduce execution consist in a set of mappers, whose number is determined by the quantity of regions of the table, and two reducers. The mappers process a given set of terms.

Each accepted tweet is evaluated at the map function using the sentiment analysis classifier (*CitiusSentiment*) to get a result which takes the values -1, 0 or 1 (negative, neutral and positive, respectively). In order to integrate the classifier into the Hadoop infrastructure we should use Hadoop Streaming as it was originally implemented using Perl. Note that Hadoop Streaming is an utility provided by Hadoop to execute applications written in languages different from Java. However, important degradations in the performance were detected using Hadoop Streaming with respect to using Java codes. For this reason, we have used Perldoop [21] to translate Perl code into Java.

The reducer computes the total sum of the query. The final value contrasts the positive tweets with the negative ones. This value is eventually normalized from the total number of processed tweets, in a scale that goes from 0 to 1. This scale represents the positivity ratio of the query. Results lower than 0.45 are considered negative and upper 0.55, positive. The remaining intermediate values represent a neutral result. Both Hadoop output files and HBase data are stored in HDFS.

VI. Performance Results

In order to test our Sentiment Analysis system we set up a Hadoop cluster in Amazon EC2. The instances were created with Amazon Linux AMI running Apache Hadoop 2.7.2 and Apache HBase 1.1.4. Amazon gives their users the possibility of running a wide variety of virtual machines in their EC2 infras-

TABLE I

Average number of unique collected tweets per second, filtered by language.

Spanish		**English**	
Streaming API	Full System	Streaming API	Full System
5.6	**259.2** (46.3×)	15.1	**275.4** (18.2×)

tructure. In our case, the cluster consists of 5 nodes of the `r3.2xlarge` instance type. This kind of EC2 instances has the following characteristics:

- CPU: Intel Xeon E5-2670 v2 (Ivy Bridge microarchitecture)
- Cores per node: 8
- RAM Memory per node: 61 GiB
- Disk: Each node has a 50 GB SSD General Purpose disk

According to the specifications provided by Amazon, the network performance of the `r3.4xlarge` instances is “high” so the theoretical bandwidth value is 1 Gbps.

The Tweet mining module has been evaluated using one `c4.8xlarge` instance. This system contains Intel Xeon E5-2666 v3 (Haswell microarchitecture) processors, there are in total 36 cores (72 threads).

A. *Tweet mining module evaluation*

In order to test the capabilities of this module, the performance was tested comparing the tweets per second collected considering the full system detailed previously in Section IV and also using the Streaming API thread working alone for Spanish and English in separated tests. The target terms were two lists of the 5,000 most frequent words for the Spanish and English languages. The *CREA* (Reference Corpus of the Current Spanish) list from RAE[6] was used for Spanish. In the case of English the terms were extracted from a 6,000 words list from `insightin.com`[7].

Tests were performed distributing the selected terms over 72 threads. Results are shown in Table I. The amount of tweets that the Streaming API can provide for a certain language is very low compared to the results obtained when executing our approach. For instance, our system collects 46.3× more tweets per second in Spanish than the Streaming API. Even if the language is not filtered when using the Streaming API the stream rate was at most 40 tweets per second.

B. *Sentiment analysis evaluation*

Our system, CitiusSentiment, took part as a participant in two different sentiment analysis competitions: TASS [22] and SemEval (task 9) [23]. TASS is an experimental evaluation workshop for sentiment analysis and online reputation analysis focused on Spanish language. SemEval (task 9 or 10) is focused on sentiment analysis in English microblogging. In both competitions, the systems were trained with different training annotated corpora. In TASS we used as input dataset the training corpus of tweets provided by the organization. This set contains 7,216 Spanish tweets, which were tagged with several polarity values: Positive, Negative, and Neutral. In SemEval, we used the training dataset of English tweets provided by the organization, which contains 6,408 tweets. The systems were evaluated against different test datasets of tweets annotated with the polarity tags. In some subtasks, further tags were used in order to distinguish between strong and weak positive tweets, between strong and weak negative tweets, or between neutral (neither positive nor negative) and objective (no polarity at all) tweets.

Tables II and III show the performance results of our system in TASS and SemEval, respectively. We compare the F-score achieved by our system with respect to those reached by both the best and worst systems, as well as the average considering all systems in each competition. The F-score is a weighted average of the *precision* and *recall*. Precision is the number of all positive results returned by the system divided by the number of results which were actually returned, while recall is the number of correct positive results returned by the system divided by the number of positive results that should have been returned. F-score reaches its best value at 1 and worst score at 0. Each competition consists in several tasks. For example, in TASS, Task-1 makes use of 6 polarity tags, Task-2 only uses 4 tags, and Task-3 deals with determining the polarity at entity level. In SemEval, each task corresponds to a different test dataset: LiveJournal sentences (Task-1), SMS messages (Task-2), tweets of SemEval 2013 (Task-3), tweets of SemEval-2014 (Task-4), and tweets with sarcasm (Task-5).

For each subtask, the ranking of CitiusSentiment is enclosed in brackets. For instance, it was ranked as the 3th best system out of 13 participants in Task-2 of TASS, and it was the first system out of 5 in Task-3 of the same competition. The results obtained in TASS contest, focused on Spanish tweets, were significantly better that those obtained in SemEval, focused on English short messages. Yet, in all tasks, including those of SemEval, the scores of CitiusSentiment are clearly above the average.

C. *Evaluation of the Big Data infrastructure*

In order to test the processing times and scalability of our system 68 Spanish terms were selected as targets for the Tweet Mining Module. These terms were relevant in mid-February 2016, when the com-

[6] `http://corpus.rae.es/lfrecuencias.html`
[7] `http://www.insightin.com/esl/`

TABLE II
F-score and ranking of our system (CitiusSentiment) in the TASS competition: Sentiment analyis on Spanish tweets.

System	Task-1	Task-2	Task-3
Best system	61.6	68.6	39.4
CitiusSentiment	55.8 (4/13)	66.8 (3/13)	39.4 (1/5)
Worst system	12.6	23.0	30.7
Average	43.3	53.0	36.9

TABLE III
F-score and ranking of our system (CitiusSentiment) in the SemEval competition, namely task 9 focused on sentiment analysis in Twitter (only English microtexts).

System	Task-1	Task-2	Task-3	Task-4	Task-5
Best system	74.8	70.2	72.1	70.9	58.1
CitiusSentiment	64.5 (29/50)	58.2 (24/50)	63.2 (24/50)	62.9 (27/50)	46.1 (25/50)
Worst system	29.3	24.6	34.2	33.0	28.9
Average	59.9	52.4	56.4	56.9	42.8

pilation of tweets started, e.g., political parties, famous people, new devices, etc. A total amount of 8,016,139 tweets written in Spanish (3.1 GiB) were stored in the HBase database. The selected terms for the tests were: *corrupción* (corruption), *gobierno* (government) and *elecciones* (elections). Table IV shows the number of matches (tweets containing the considered word) and the positivity ratio obtained by the system related to each term.

Due to the fact that Hadoop handles each region of a HBase table like a split, the original table was split into equal parts progressively, getting always a number of regions power of two from 1 to 32. A total amount of 50 GiB of RAM per node was configured for YARN containers, 7 GiB for every map or reduce container. 3GiB for HBase and the rest for other Hadoop processes and the OS. The number of cores per node (YARN) was set to 8.

HBase tables are split in regions according to the established policies. For this experiment, the split policy that was used was the `ConstantSizeRegionSplitPolicy`. Thereby, every region is split when one of their column families exceeds the value for the `hbase.hregion.max.filesize` attribute. This parameter was set to 20GiB, getting only one region in the table for all the tweets that were stored. For the purpose of obtaining the different number of regions, the tests were performed on the table in many iterations. In every iteration, each region was manually divided in two equal parts through the HBase Shell. It is important to set some properties to the Scan instance that will be used by the mappers. The cache blocks option must be disabled (the MapReduce job is a batch reading and this option is intended for frequently accessed rows). Furthermore, an important parameter when performing readings is the client prefetch (called caching). This refers about how many rows from the table could be retrieved in a single RPC call by the client. Using the default value (1) will cause a huge degradation of the execution performance. For the tests, the chosen values were 50 and 5,000 (rows).

Results of the experiments carried out are displayed in Table V, which show that the system scales up quite good when the physical resources are enough to handle the launched tasks. Note that our system is capable of processing more than 8 million tweets in few minutes. Considering the term *elecciones*, the one with the lowest number of coincidences, using 32 mappers degrades the performance due to the overhead. In addition, low values for the caching parameter in the Scanner object causes worse results. This behavior is clearly detected with few regions. Additional tests were executed using other values for caching, but higher values for this parameter do not show relevant differences with respect to the results shown in Table V.

VII. Conclusions

Twitter can be considered as a large source of short texts (tweets) containing user opinions. Making sentiment analysis on tweets is challenging from the natural language processing perspective, but also in terms of performance when huge amounts of tweets should be processed. Both challenges are addressed in this paper.

First, our sentiment classifier (*CisitusSentiment*) has obtained good results considering both Spanish and English tweets. We must highlight that our classifier took part as a participant in two different sentiment analysis competitions, reaching scores clearly above the average with respect to other approaches.

Regarding the second challenge, we propose a new system that takes advantage of parallel architectures using Big Data technologies with the aim of speeding up the sentiment analysis process. The system consists of three modules. The first module is respon-

TABLE IV
Successfully processed tweets, matches and positivity ratio for the selected Spanish terms.

Term	Processed tweets	Matches	Positivity ratio
corrupción	8,016,139	195,590	0.304 (negative)
gobierno		99,601	0.433 (negative)
elecciones		30,001	0.577 (positive)

TABLE V
Processing time (in minutes) for the selected Spanish terms.

	corrupción		*gobierno*		*elecciones*	
	Scanner caching (prefetch)					
No. of reg.	50	5,000	50	5,000	50	5,000
1	52.5	50.7	29.2	27.8	12.4	10.9
2	30.2	27.5	15.9	15.0	7.4	6.7
4	17.9	16.6	10.2	9.1	4.7	4.0
8	11.2	8.4	5.7	5.5	3.2	2.7
16	7.4	6.5	4.5	4.1	2.4	**2.1**
32	5.3	**5.1**	3.9	**3.7**	2.4	2.3

sible of collecting the tweets. It has been designed in such a way that captures 46× more tweets per second than the standard Twitter Streaming API. The second module uses Hadoop to process the tweets captured by the first module. Performance results show that our system is able to analyzed more than 8 million tweets in a few minutes on a small cluster. Finally, a GUI is also provided for the users.

Acknowledgment

This work was supported by MINECO (ref. TIN2014-54565-JIN) and RedPLIR-Xunta de Galicia (ref. CN2014/034). It was also supported by AWS in Education Grant award.

References

[1] "Apache Hadoop home page," `http://hadoop.apache.org/`, [Online; accessed February, 2016].

[2] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," in *Symposium on Operating System Design and Implementation*, 2004, pp. 10–10.

[3] Satish Narayana Srirama, Pelle Jakovits, and Eero Vainikko, "Adapting scientific computing problems to clouds using MapReduce," *Future Generation Computer Systems*, vol. 28, no. 1, pp. 184 – 192, 2012.

[4] Jared Kramer and Clara Gordon, "Improvement of a Naive Bayes Sentiment Classifier Using MRS-Based Features," in *Joint Conf. on Lexical and Computational Semantics*, 2014, pp. 22–29.

[5] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan, "Thumbs Up?: Sentiment Classification Using Machine Learning Techniques," in *Conf. on Empirical Methods in Natural Language Processing*, 2002, vol. 10, pp. 79–86.

[6] Franco Salvetti, Christoph Reichenbach, and Stephen Lewis, "Opinion polarity identification of movie reviews," in *Computing Attitude and Affect in Text: Theory and Applications*, pp. 303–316. 2006.

[7] Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede, "Lexicon-based methods for sentiment analysis," *Comput. Linguist.*, vol. 37, no. 2, pp. 267–307, 2011.

[8] Saif M. Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu, "Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets," *CoRR*, vol. abs/1308.6242, 2013.

[9] X. Saralegi and I. San Vicente, "TASS: Detecting Sentiments in Spanish Tweets," in *Workshop on Sentiment Analysis at SEPLN*, 2012.

[10] Aliaksei Severyn, Alessandro Moschitti, Olga Uryupina, Barbara Plank, and Katja Filippova, "Multi-lingual opinion mining on youtube," *Inform. Processing & Management*, vol. 52, no. 1, pp. 46–60, 2015.

[11] David Vilares, Miguel A. Alonso, and Carlos Gómez-Rodríguez, "On the usefulness of lexical and syntactic processing in polarity classification of twitter messages," *Journal of the American Society for Information Science*, vol. 66, no. 9, pp. 1799–1816, 2015.

[12] B. Liu, E. Blasch, Y. Chen, D. Shen, and G. Chen, "Scalable sentiment classification for big data analysis using naive bayes classifier," in *IEEE Int. Conference on Big Data*, 2013, pp. 99–104.

[13] L. Banić, A. Mihanović, and M. Brakus, "Using Big Data and sentiment analysis in product evaluation," in *Int. Convention on Information Communication Technology Electronics Microelectronics*, 2013, pp. 1149–1154.

[14] A. H. A. Rahnama, "Distributed real-time sentiment analysis for big data social streams," in *Int. Conf. on Control, Decision and Information Technologies*, 2014, pp. 789–794.

[15] J. Conejero, P. Burnap, O. Rana, and J. Morgan, "Scaling archived social media data analysis using a hadoop cloud," in *Conf. on Cloud Computing*, 2013, pp. 685–692.

[16] Chris Manning, Prabhakar Raghadvan, and Hinrich Schütze, *Introduction to Information Retrieval*, Cambridge University Press, Cambridge, MA, USA, 2008.

[17] Peter Sheridan Dodds, Kameron Decker Harris, Isabel M. Kloumann, Catherine A. Bliss, and Christopher M. Danforth, "Temporal patterns of happiness and information in a global social network: Hedonometrics and Twitter," *PLoS ONE*, vol. 6, no. 12, pp. e26752, 2011.

[18] Bing Liu, Minqing Hu, and Junsheng Cheng, "Opinion Observer: Analyzing and Comparing Opinions on the Web," in *Int. World Wide Web conference*, 2005, pp. 342–351.

[19] Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani, "SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining," in *Int. Conf. on Language Resources and Evaluation*, 2010, pp. 2200–2204.

[20] Grigori Sidorov, "Empirical study of opinion mining in spanish tweets," *Lecture Notes in Artificial Intelligence*, vol. 7629-7630, 2012.

[21] J. M. Abuín, J. C. Pichel, T. F. Pena, P. Gamallo, and M. García, "Perldoop: Efficient execution of Perl scripts on Hadoop clusters," in *Int. Conf. on Big Data*, 2014, pp. 766–771.

[22] Pablo Gamallo, Marcos Garcia, and Santiago Fernández-

Lanza, "A naive-bayes strategy for sentiment analysis on spanish tweets," in *Workshop on Sentiment Analysis at SEPLN (TASS)*, 2013, pp. 126–132.

[23] Sara Rosenthal, Preslav Nakov, Alan Ritter, and Veselin Stoyanov, "SemEval-2014 Task 9: Sentiment Analysis in Twitter," in *Int. Workshop on Semantic Evaluation*, 2014.