On-line Relevant Anomaly Detection in the Twitter Stream: An Efficient Bursty Keyword Detection Model

Jheser Guzman
PRISMA Research Group
Department of Computer Science
University of Chile
Santiago, Chile
jguzman@dcc.uchile.cl

Barbara Poblete
PRISMA Research Group
Department of Computer Science
University of Chile
Yahoo! Labs Santiago
Santiago, Chile
bpoblete@dcc.uchile.cl

ABSTRACT

On-line social networks have become a massive communication and information channel for users world-wide. In particular, the microblogging platform Twitter, is characterized by short-text message exchanges at extremely high rates. In this type of scenario, the detection of emerging topics in text streams becomes an important research area, essential for identifying relevant new conversation topics, such as breaking news and trends. Although emerging topic detection in text is a well established research area, its application to large volumes of streaming text data is quite novel. Making *scalability*, *efficiency* and *rapidness*, the key aspects for any emerging topic detection algorithm in this type of environment.

Our research addresses the aforementioned problem by focusing on detecting significant and unusual bursts in keyword arrival rates or *bursty keywords*. We propose a scalable and fast on-line method that uses normalized individual frequency signals per term and a windowing variation technique. This method reports keyword bursts which can be composed of single or multiple terms, ranked according to their importance. The average complexity of our method is $O(n \log n)$, where n is the number of messages in the time window. This complexity allows our approach to be scalable for large streaming datasets. If bursts are only detected and not ranked, the algorithm remains with lineal complexity $O(n)$, making it the fastest in comparison to the current state-of-the-art. We validate our approach by comparing our performance to similar systems using the TREC Tweet 2011 Challenge tweets, obtaining 91% of matches with LDA, an off-line gold standard used in similar evaluations. In addition, we study Twitter messages related to the SuperBowl football events in 2011 and 2013.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Retrieval models, Information filtering*

Keywords

Twitter, Text Mining, Bursty Keyword Detection

ODD'13, August 11th, 2013, Chicago, IL, USA.

1. INTRODUCTION

Social media microblogging platforms, such as Twitter[1], are characterized by extremely high exchange rates. This quality, as well as its network structure, makes Twitter ideal for fast dissemination of information, such as breaking news. Furthermore, Twitter is identified as the first media source in which important news is posted [10] and national disasters (e.g. earthquakes, diseases outbreaks) are reported [6].

Each message that is posted on Twitter is called a *tweet*, and messages are at most 140-characters long. In addition, Twitter users connect to each other using a follower/followee directed graph structure. Users can support and propagate messages by using a *re-tweet* feature which boosts the original message by reposting it to the user's followers.

Given that Twitter has been adopted as a preferred source for instant news, *real-time detection* of emerging events and topics has become one of the priorities in on-line social network analysis. Research on emerging topic detection in text is now focused on the analysis of *streaming data* and *on-line identification* [1]. In particular, the analysis of microblog text is quite challenging, mostly because of the large volume and high arrival rate of new data. In this scenario an important part of the analysis must be performed in real-time (or close to real-time), requiring *efficient* and *scalable* algorithms.

This problem has generated increasing interest from the private and academic communities. New models are constantly being developed to better understand human behavior based on social media data in interdisciplinary work fields such as sociology, political science, economy and business markets [3, 26].

We target the problem of emerging topic detection in short-text streams by proposing an algorithm for *on-line Bursty Keyword Detection (BD)* (definitions in Figure 1). This in general is considered to be a first important step in the identification of emerging topics in this context. Our approach uses windows slicing and window relevance variation rate analysis on keywords.

We validate our methodology on the TREC Tweet 2011 dataset, using it to simulate streaming data. Our experiments indicate that this concept works better at detecting keyword bursts in text streams than other more complex state-of-the-art solutions based on queue theory [8, 14, 24]. Also we analyze tweets from the SuperBowl football events in 2011 and 2013 as case studies. We perform a detailed discussion on the behavior of noise in bursty keyword signals which is constituted mostly by stopwords. We compare our solution to LDA [2] as a ground truth, similarly to prior work [24]. Using LDA in a one-keyword per topic mode, we achieved more

[1] http://www.twitter.com

than 91% topic matches. This is a great improvement over similar approaches. Moreover, our implementation is efficient, achieving a complexity of $O(n \log n)$ in the average case, where n is the number of messages in the current window.

In detail, the contributions of our work are three-fold:

1. We introduce a scalable and efficient keyword burst detection algorithm for microblog text streams, based on window slicing and relevance window variations.
2. We present a technique for eliminating non-informative and irrelevant words from microblog text streams.
3. We present a detailed proof-of-concept system and validate it on a public dataset.

Definition 1: A *keyword* is an informative word used in an information retrieval system to indicate the content of a document.
Definition 2: A *bursty keyword* is defined as a word or set of words that suddenly appear in a text stream at an unusually high rate [14].
Definition 3: A *topic* [*noun*] is a discussion or conversation subject. In other words, it is a set of keywords with semantic association.
Definition 4: An *event* is a topic which in general, corresponds to a real-world occurrence. It is usually associated with a geographic location and a time.
Definition 5: The concept of *relevance* used in this work means: the representation of a term in a window. In other words "the probability of occurrence of a term in a Window".

Figure 1: Definitions used in the paper (Oxford Dictionary)

This paper is organized as follows. Section 2 presents an overview of relevant literature for this work. Section 3 presents a complete description of our proposal, divided in two parts: the burst detection algorithm and our proof-of-concept system. Section 4 summarizes our experimental validation and results on the SuperBowl 2011 and SuperBowl 2013 events. Section 5 delivers our conclusions and discuss future work.

2. RELATED WORK

Our work involves research in the areas of event detection and trend analysis for microblog text streams. In particular our goal is to *identify current popular candidate events listed by most popular bursty terms in the data stream*. In relation to this topic, several applications exist for detecting events like natural disasters and health alerts. For example, epidemics [11], wildfires [23], hurricanes and floods [20], earthquakes and tornados [7, 17].

Events have been modeled and analyzed over time using keyword graphs [18], link-based topic models [13], and infinite state automatons [8]. Leskovec *et al.* [12] perform analyses of memes for news stories over blogs and news data, and on Twitter data in [25]. Swan *et al.* [21, 22] deal with constructing overview timelines of a set of news stories.

On-line bursty keyword detection is considered as the basis for on-line emerging topic detection [14]. For topic detection in an off-line fashion, algorithms such as Latent Dirichlet Allocation (LDA) [2] or Phrase Graph Generation Method [19] can be used. Statistical methods and data distribution tests can also be used to detect bursty keywords [8].

Mathioudakis and Koudas introduce Twitter Monitor [14], a system that performs detection of topic trends (emerging topics) in the Twitter stream. Trends are identified based on *individual keyword* bursts and are detected in two steps. This system identifies bursts of keywords by computing the occurrence of individual keywords in tweets. The system groups keyword trends based on their co-occurrences. To detect bursts of keywords, they introduce the algorithm QueueBurst with the following characteristics: (1) *one-step analysis per keyword*, (2) *real-time analysis (based on the tweet stream)*, (3) *adjustable against "false" explosions*, and (4) *adjustable against spam*. In order to group sets of related bursty keywords, the authors introduce an algorithm named GroupBurst, which evaluates co-occurrences in recent tweets. Our current work focuses on keyword detection up to three term keywords, for which we compare to QueueBurst in Section 4. Twitter Monitor requires an intensive pre-processing step for determining its optimal parameter settings for each keyword and also for global variables. These parameter settings must be computed with a historical dataset.

A different approach is presented by Weng *et al.* [24], with their system EDCoW (Event Detection with Clustering of Wavelet-based Signals). EDCoW builds individual word signals by applying wavelet analysis to word frequencies. It filters away trivial words by looking at their corresponding signal auto-correlations. The remaining words are clustered to form events with a modularity-based graph partitioning technique. The wavelet transform is applied to a signal (time-series) created using the TF-IDF index [5]. Their approach was implemented in a proof-of-concept system, which they used to analyze online discussions about the Singapore General Election of 2011.

Another relevant study is that of Naaman *et al.* [15]. In this work the authors make two contributions for interpreting emerging temporal trends. First, they develop a taxonomy of trends found in data, based on a large Twitter message dataset. Secondly, they identify important features by which trends can be categorized, as well as the key features for each category. The dataset used by Naaman *et al.* consists of over 48 million messages posted on Twitter between September 2009 and March 2010 by 855,000 unique New York users. For each tweet in this dataset, they recorded its textual content, the associated timestamp and the user ID.

3. BURST DETECTION MODEL

We propose a methodology based on time-window analysis. We compute keyword frequencies, normalize by relevance and compare them in adjacent time windows. This comparison consists of analyzing variations in *term arrival rates* and their respective *variation percentages* per window. A similar notion (Discrete Second Derivative) has been used in the context of detection of bursts in academic citations [4]. We define *Relevance Rates (RR)* as the probability of occurrence of a non-stopword term in a window. We use RR to generalize burst detection making them independent of the arrival rate. Even though the public Twitter API only provides a stream which is said to be less than 10% of the actual tweets posted on Twitter, we believe our method can be easily adapted for the complete data stream using a MapReduce schema [see Section 4.2.3]. Arrival rates vary periodically (in a non-bursty way) during the day; depending on the hour, time zone, user region, global events and language (shown in Figure 4).

Bursty keywords are ranked according to their *relevance variation rate*. Our method avoids the use of statistical distribution analysis methods for keyword frequencies; The main reason is that this approach, commonly used in state-of-the-art approaches, increases the complexity of the process. We show that a simple relevance variation concept is sufficient for our purposes if we use good stopword filter and noise minimization analysis [see section 4.1.1].

To study the efficiency of our algorithm, which we name *Window Variation Keyword Burst Detection*, we implement a proof-of-concept system. The processes involved in this system are five. These modules are independent of each other and they have been structured for processing in threads. These modules are: Stream

Listener, Tweets Filterer, Tweets Packing, Window Processing and Keyword Ranker. This architecture allows us to process all the input data with linear complexity, making it scalable for on-line processing.

3.1 Data Pre-Processing

In this stage, data is pre-processed extracting keywords from each message, so that later on, our burst detection algorithm analyzes them (see Figure 2 and 3). This stage is composed of the following three modules: *(1) Stream Listener*, *(2) Tweet Filter* and *(3) Tweet Packing*.

1) Stream Listener Module: This module receives streaming data in the form of Twitter messages, which can come directly form the Twitter API or some other source[2] . Messages are received in JSON format[3]. This data is parsed and encapsulated. After the encapsulation of each message it is en-queued in memory for the next module in the pipeline. It should be noted that message encapsulation is prone to delays caused by the Internet bandwidth connection and Twitter's information delivery rate, which can cause data loss.

2) Tweet Filter Module: This module discards messages which are not written in languages accepted by our system. We perform language classification using a Naive Bayes classifier. This module also standardizes tweets according to the following rules:

- Treatment of special characters and separation marks: Replacing special characters and removal of accents, apostrophes, etc.
- Standardization of data: Upper and lower case conversion and replacement of special characters.

After normalization and language detection, the tweet is enqueued into queue Q_1 for posterior analysis.

3) Tweet Packing Module: Filtered and standardized tweets in queue Q_1, are grouped into a common set determined by creation timestamp, shown in Figure 2. This set of tweets, which we refer to as *Bag of Tweets*, represent an individual time-window. It is important to note that the arrival of tweets maintains a chronological order. In the case that an old or delayed tweet appears, it is included in the current window. Each of these windows is sent to the following stage for processing.

3.2 Bursty Keyword Detection

This process involves two modules. The second module must wait for the first module to finish processing a window in order to process an entire window at a time (serial mode). The algorithm shown in Figure 3 describes this process, where the second module starts in line 24.

1) Window Processing Module: Each keyword, composed of a single or adjacent word n-grams, is mapped into a hash table data structure. This structure manages keywords in addition to the information of its two adjacent windows and their rates. We consider as *n-grams* the n ordered correlative words.

The hash table allows access to keyword information in constant time for most cases *O(1)*, and in the worst case with complexity of *O(n)* when collisions occur. This process is detailed in the algorithm described in Figure 3. This data structure controls the complexity of the algorithm with optimal insertions and search $O(1)$.

2) Keyword Ranker Module: Bursty keywords are included implicitly into the hash table. Therefore, we extract bursty keywords by discarding those that do not classify as having a positive relevance variation. We discard non-bursty keywords using the criteria

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

[3] `https://dev.twitter.com/docs/tweet-entities`

Require: Global Queue Q_1 of previously filtered tweets, and Global Queue Q_2 of keyword bags. $window\_time$ is the earliest timestamp of the tweets in the same Bag.
Consider $window\_size$ as the time length of the window.

1: $initTime \Leftarrow null$
2: **while** $t' \Leftarrow get\_tweet\_from\_queue(Q_1)$ **do**
3: **if** $initTime = null$ **then**
4: $initTime \Leftarrow timestamp(t')$
5: $endTime \Leftarrow initTime + window\_size$
6: $TBag \Leftarrow \phi$
7: Set $initTime$ to $window\_time$ in $TBag$
8: **end if**
9: **if** $timestamp(t') < endTime$ **then**
10: $t'' \Leftarrow filter\_stopwords(t')$
11: $TBag \Leftarrow TBag \bigcup keywords(t'') \bigcup word\_nGrams(t'')$
12: **else**
13: $put(TBag)$ in Q_2
14: Create new $TBag$
15: $TBag \Leftarrow \phi$
16: Set $endTime$ to $window\_time$ in $TBag$
17: $initTime \Leftarrow endTime$
18: $endTime \Leftarrow initTime + window\_size$
19: **end if**
20: **end while**

Figure 2: Packer Thread

Require: Global Queue Q_2 of keyword bags, and a Global hash table HT mapping $< Keyword, < window_1, window_2, rates >>$

1: **while** $TBag \Leftarrow get\_bag\_from\_queue(Q_2)$ **do**
2: $TotalWords \Leftarrow size(TBag)$
3: **for all** $word \in TBag$ **do**
4: **if** $word \notin mapped\_keywords(HT)$ **then**
5: $window\_time \Leftarrow window\_time(TBag)$
6: $freq \Leftarrow 0$
7: $window \Leftarrow < window\_time, freq >$ {create a new window}
8: $put\_in\_hashtable\_as\_window1(word, window)$
9: **else**
10: $window \Leftarrow get\_from\_hashtable(word, HT)$
11: $w\_time \Leftarrow window\_time(window)$
12: $b\_time \Leftarrow window\_time(TBag)$
13: **if** $w\_time \neq b\_time$ **then**
14: $window_2 \Leftarrow window_1$ for $word$ in HT
15: $window\_time \Leftarrow window\_time(TBag)$
16: $freq \Leftarrow 0$
17: $window \Leftarrow < window\_time, freq >$ {create a new window}
18: $put\_in\_hashtable\_as\_window1(word, window)$
19: **end if**
20: Add 1 to $freq$ in $window_1$ for $word$ mapped in HT
21: **end if**
22: **end for**
23: $set\_relevance\_for\_each\_keyword\_with(TotalWords)$
24: $WordRank \Leftarrow \phi$
25: **for all** $mword \in mapped\_keywords(HT)$ **do**
26: $window_1 \Leftarrow get\_window_1(mword, HT)$
27: $window_2 \Leftarrow get\_window_2(mword, HT)$
28: $relevance_1 \Leftarrow get\_relevance(window_1)$
29: $relevance_2 \Leftarrow get\_relevance(window_2)$
30: **if** $(relevance_1 - relenvance_2) > 0$ **then**
31: $WordRank \Leftarrow WordRank \bigcup mword$
32: **end if**
33: **end for**
34: $WordRank' \Leftarrow quickSort(WordRank)$
35: display $WordRank'$
36: **end while**

Figure 3: Processor Thread

Table 1: Tweet language classification results

Language	%	Freq.
English	34.4	4,287,605
Romanic (Neo-Lat)	20.9	2,599,849
German	2.5	305,228
Others	42.2	5,253,562
TOTAL	100.0	12,446,244

Table 2: Average word count per message

Language	Words per Msg
English	7
Romanic (Neo-Lat)	12
German	10

described below, this prevents the size of the hash table from growing out of control:

1. It is the first occurrence of the keyword. We must wait until the next window to check it again.
2. We observe a negative variation in frequency rates between adjacent windows.
3. Low arrival rate: Many words do not appear frequently. We discard these words if the average arrival rate is lower than 1.0 (keywords per time-window).

The remaining keywords are sorted in descending order according to their *Relevance Variation Rate*. Keywords with the highest variation rate or *burstiness* are ranked in the top positions.

4. EVALUATION

In this section we explain empirical parameter settings for our system and perform a validation by comparing it against state-of-the-art methods. For evaluation and comparison purposes we adapt the evaluation used in [24] with LDA as a gold standard. LDA is used to process data off-line and generate topics listed by its most likely keyword.

4.1 Dataset Description

The datasets used in this experiment are the TREC Tweet 2011 Challenge dataset from the National Institute of Standards and Technology (NIST[4]): In addition, we use tweets related to the Super-Bowl 2013 football event obtained using Twitter API on February 3rd. As part of the TREC 2011 microblog track, Twitter provided identifiers for approximately 16 million tweets, sampled from its full data-stream between January 23rd and February 8th, 2011. The corpus is designed to be a reusable and representative sample of the Twitter stream. The TREC Tweet 2011 messages are obtained using a crawler, fed with tweet IDs provided by NIST for downloading messages directly from Twitter. As expected, not all messages were obtained in the downloading process, mostly because of: the tweet having been removed, transmission or server errors, or changes made in the access permissions by the owner of the message. Therefore, we only obtained $12,446,244$ tweets.

The origin of these messages is random regarding their geographic location, and their languages. To identify the language of each message, we use the java library *LangDetect*[5] which claims 93% accuracy in the detection of 59 languages in short messages[6]. Using this classifier we obtain the classification shown in Table 1.

In this work we only use English, Neo-Lat[7] and German languages. These correspond to 57.8% of the dataset with a total of $7,192,682$ messages. The main reasons for selecting these languages are:

- The use of the *space character* as a word delimiter.
- Same writing structure order (left to right and top to bottom).
- Same set of character symbols for writing.

In table 2 we show the average number of words per message in each language. Twitter's 140-character limit for messages is an advantage for the filtering and tokenizing steps which take place in lineal complexity $O(wn)$ using single words, and $O(w^2n)$ adding adjacent word n-grams. The variable n corresponds to the number of tweets to be processed and w the average number of words per message. Since w is a constant upper-bound, we conclude that the complexity of the algorithm described in Figure 2 remains lineal $O(n)$.

4.1.1 Stopword analysis

We consider *stopwords* as words used to connect objects, actions and subjects. Stopwords are omitted in our experiments (see Figure 2, line 10) and discarded since they do not add semantic value or represent events.

The *Window Variation Keyword Burst Detection* (BD) technique computes keyword arrival frequency in time-windows using a predefined window length that remains constant during the entire process. The use of the absolute frequency value is the most intuitive approach, but it does not work correctly because the arrival rate is not constant during the day. This is explained in correlation with sleep hours, work hours, days of the week or season of the year in which the tweet is posted (many of these changes can show periodicity in time). Other factors can affect tweet arrival rate behavior, with an important one being language and geographical location of the user at that moment. As shown in Figure 4, frequency signals decrease between 2:00 and 9:00 GMT hours, especially during sleeping hours. The signal slowly increases between 9:00 and 17:00 GMT hours, when people are at work. After that, the signal remains stable until 2:00 GMT the next day. Therefore, we opt for using *Relevance Variation rates* for window comparison, which creates independence between values and the hour of the day.

In Figure 4 we show frequency signals according to the arrival rate of each language. We observe that country time zone can affect the behavior of certain keywords. Portuguese tweets are clearly shifted some hours in comparison to other languages. We believe that this happens because a large number of tweets in this language originate in Brazil. Again, this effect is mitigated by the use of relative rates (or percentages) instead of absolute values.

A common technique to filter or discard stopwords is the use of predetermined list of stopwords per languages. In social media, especially in Twitter, the behavior of stopwords is dynamic. This occurs because the message limit of 140-characters forces the user to reduce or transform their message to utilize less space. Therefore, stopwords in Twitter messages vary from those of standard document lists. To identify stopwords for our experiment in this type of environment (microblogging), we download standard stopword lists[8] and map them onto a frequency histogram of our keyword

[4] `http://trec.nist.gov/data/tweets/`

[5] `http://code.google.com/p/language-detection`

[6] http://shuyo.wordpress.com/2011/11/28/language-detection-supported-17-language-profiles-for-short-messages/

[7] Romanic and Romance Languages: Spanish, French, Italian, Portuguese.

[8] Stopwords List: `http://snowball.tartarus.org/algorithms/`

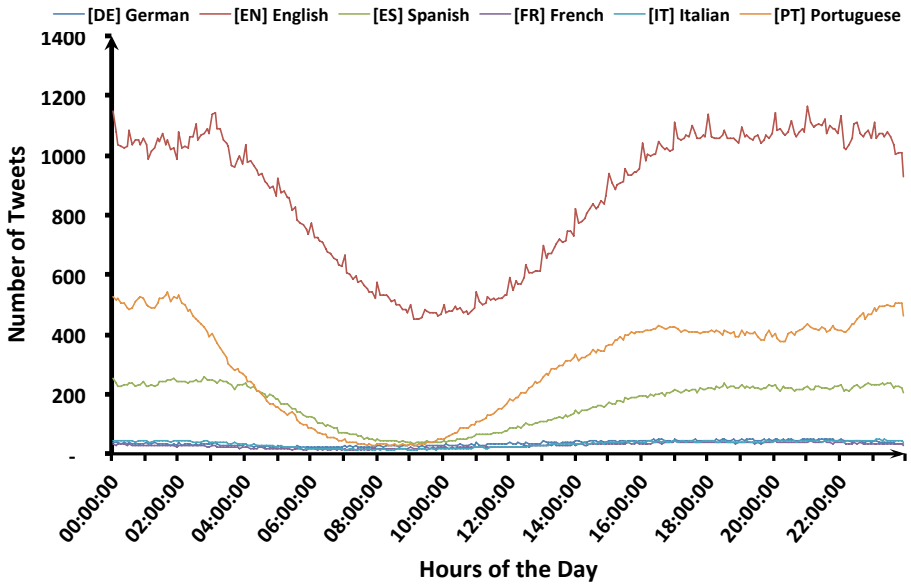

Figure 4: Tweet arrival rate per language (GMT timezone): Shows differences between language volumes and the shift in the signals due to different timezones.

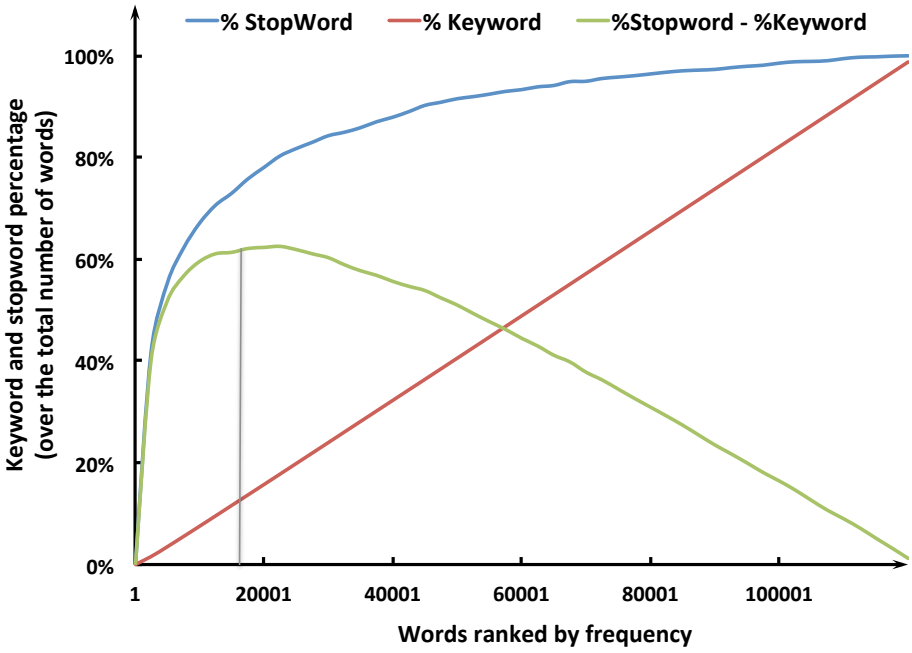

Figure 5: Keywords and stopwords ranked by frequency histogram (accumulative): The exact point where stopwords start decreasing their volume.

dataset (made up of the complete vocabulary found in our collection of tweets). We observe that most stopwords are concentrated in the area of the histogram which contains words with a high arrival rate. Therefore, we mark as candidate stopwords all keywords with more than $18,000$ frequency occurrence in our keyword data set. Nevertheless, by inspecting our stopword candidate list we see that it still contains some relevant keywords which we do not wish to mark as stopwords (false negatives). These cases occur because some relevant keywords have high occurrence rates during certain time periods. In order to avoid these cases, we compute the *Relative Standard Deviation* (RSD) rate[9] and use it to describe frequency signal stability of keywords. Intuitively, stopword signals should be noisier (i.e., have a higher $standard\_deviation$) than the signal of regular keywords because they do not represent an organized phenomena. Figure 6 shows that stopwords are **twice as noisy as regular words**, therefore we consider this as a parameter for stopword identification. Also, we measure RSD using different window sizes (shown in Figure 6) in order to identify the optimal *window size* for which stopwords and keywords are less affected by noise.

Using the previously described methodology, we obtain a final list of stopwords that contains $18,631$ words. With this, we remove 90% of total word arrivals, making the process **significantly faster**. It also **reduces the volume of words** for the final hash table in algorithm in Figure 3, therefore requiring less memory. Stopwords can be updated by performing off-line batch analysis which does not affect on-line performance.

4.1.2 Determining optimal window size

The stability of the signal (minimization of RSD) helped us determine the optimal window size for our algorithm. This is a crucial parameter that determines the performance of our solution.

- If the window size is too small, the occurrence of empty windows for a term increases (frequency equal to 0), making the noise rate increase and frequency rate tend to zero.
- On the other hand, if the window size is too large, the stability of the signal becomes constant and bursty keyword detection is delayed.

Therefore, it seems reasonable to place optimal window size somewhere between 17 minutes and 2 hours. In practice, we select windows of 20 minutes for actual tweet arrival rate for *fast detection* of bursts. This choice is practical because it divides a 24-hour day exactly, making the analysis easier to understand and to compare windows from different days. It is important to remark that this decision of 20 minutes is for the actual Twitter API arrival rate of tweets that represent less than 10% of the total system. We expect to reduce the time of the window size if the arrival rate is higher. The effect of using higher size on windows would not detect bursts because the signal will tend to the average value. Using 20 minutes as window size, we minimize the time of bursts detection keeping the noise level in the signal to a minimum.

4.2 Comparison to other methods

TwitterMonitor (TM) [14] is one of the earliest works in the field of detecting emerging topics on Twitter. Its core algorithm named QueueBurst uses five parameters and concepts of Queue Theory M/M/1. We used the recommended parameter settings from the technical paper provided directly to us by the authors, setting the tolerances ϵ_0 and ϵ_1 to 10^{-2}, and the ratio r to 2. The arrival rates (λ_0) per each keyword were calculated using the first week of tweets of the TREC Tweet 2011 dataset to set each keyword.

[9]Relative Standard Deviation (RSD): $Standard\_Deviation/Average$

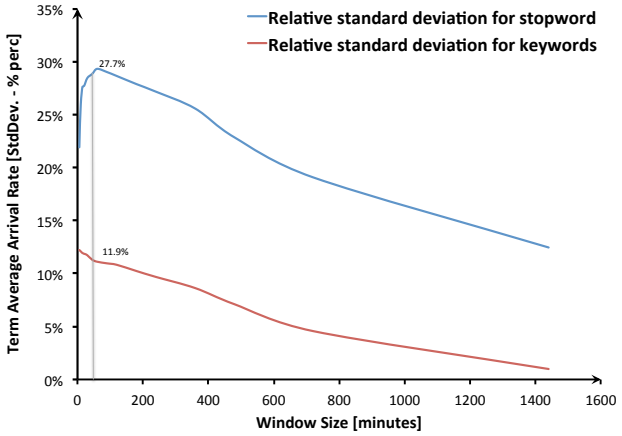

Figure 6: Relative standard deviation of keywords and stopwords: Noise in stopwords is higher than in keywords (non-stopwords). The 20-minute windows size stabilizes the noise in both signals keeping the window size as short as possible.

Weng *et al.* [24] developed EDCoW (Event Detection with Clustering of Wavelet-based Signals), a system that uses queueing technique for bursty keyword detection [8] and wavelet techniques to detect trends in Twitter.

As mentioned earlier, LDA (with the Gibbs Sampling technique for parameter estimation [16]) is a reasonable gold standard for our evaluation. This follows the approach used in the EDCoW article [24] to compare TM and EDCoW with our proposal. Given that with EDCoW there are no details available for the implementation, we can only perform a similar experiment and compare results to TM and our method.

Next, we analyze the complexity of TM, LDA and BD. We cannot analyze the complexity of EDCoW. It should be noted that LDA is an off-line method, therefore it competes with an advantage in relation to on-line methods TM, BD (and EDCoW) given that it has a complete view of the information, as opposed to the limited data that on-line methods use.

- TM is $O(wni)$, where n is the number of tweets to be processed, w is the average number of words per message and i is the iterations for generating the exponential aleatory variables for the M/M/1 queue. The parameter i cannot be determined exactly because of randomness, leaving the complexity of the algorithm in $O(ni)$. This algorithm analyzes each tweet in one pass, but this property creates delays because it cannot be parallelized.
- LDA is a statistical method that determines k topics that represent a document, where k is the number of the top events in that document. Assume we have N documents and a vocabulary of size V. The complexity of mean-field variational inference for LDA is $O(NKV)$. In this scenario, we assume a document to be the concatenation of all of the tweets in a same time-window (as in [24]). It should be noted that LDA does not constitute a perfect gold standard for burst detection in streaming data, but constitutes an approximation. Some topics might not be bursty and some bursts do not correspond to topics.
- For our *Window Variation Keyword Burst Detection* (BD) a threshold T can be included in order to truncate the number of bursty keywords returned. This, and an optimal selection algorithm [9], reduce complexity of the ranking algorithm to

Table 3: Examples of bursty keywords detected in the TREC Tweet 2011 dataset for February 6th, 2011

Window Time	Top 10 Keywords	Tweet Examples
17:20 GMT	liverpool, torres, justinbieber, meireles, chelsea, super, greenandyellow, blackandyellow, commercials, bowl	• berapa" chelsea liverpool? • A little over 6 hours until Super Bowl starts! Do you have everything you need? • #BrandBowl - Mullen Leverages @Twitter, @Radian6 to Rank the Best Super Bowl Ads http://cot.ag/hkL6H2 #sb45 • if torres is worth £50m I must be worth about a tenner . Nice one @LeedsLadAdam
20:40 GMT	corinthians, palmeiras, pra, julio, agora, jogo, chupa, gol, cesar, vai	• @ferpetucco palmeiras perdeu pro Corinthians ? O_O • 2T 37m. GOOOOOOOOOOOOOOOOLLLLLLLLLLLLL E DO CORINTHIANS!!!! • GOOOOOOOOOOL DO CORINTHIANS! Aleshow faz o dele! Palmeiras 0 x 1 Corinthians. #vccomenta
23:20 GMT	christina, superbowl, anthem, national, aguilera, sing, super, bowl, lea, singing	• National anthem over-under is 1:54. I think Christina Aguilera s hitting the over. • The Superbowl seems to be 90% entertainment, and 10% of the actual game. #bbcsuperbowl • superbowl lea michele cantandoooo :) • Que elegante se ve Christina, yo pense que iba a salir en minifalda con tanga :P.....:O

$O(Tn)$. Because T is constant it remains lineal $O(n)$. This technique does not require parameters to be reset at run time. Otherwise, if we decide not to use a threshold, the complete ranking would take $O(n\log(n))$.

We compare our BD algorithm to the topics returned by LDA on a one-keyword-per-topic basis. We do this to determine the percentage of topic matches on the TREC Tweet 2011 dataset. We also compared Twitter Monitor with LDA using the same dataset and assumptions, and discuss results obtained similarly in EDCoW [24].

4.2.1 Results and discussion

The comparison of BD and TM against LDA is performed window by window. We compare the number of keywords that overlap between each time-window and their respective match percentages (see Figure 7). Our system BD displays 91% coincidences with LDA. Comparing TM with LDA, there is only a 3% keyword match. We also looked at the overlap between bursty keywords reported by BD and TM, obtaining only a 14% match.

We believe that the low percentage of the coincidences between LDA and TM corresponds to the sensitivity of TM's empirical parameter settings. Also, TM keeps reporting the same keywords in time because they have not yet been dropped in the following windows (while they do not satisfy the hypothesis H_0 of the TM algorithm).

In addition, the results reported by Weng *et al.* [24] for their system EDCoW are of a 22% match with LDA in the best case.

Therefore, in our experimental evaluation we observed that our algorithm (BD) outperformed TM and EDCoW with 91% coincidences with LDA versus 3% and 22% of the other methods respectively. Figure 7 shows the percentage of coincidences against LDA on the TREC Tweet 2011 Dataset. These percentages were estimated for each 20-minute time-window in the dataset. The horizontal-axis considers the beginning of the first window on January 23rd 00:00 hours and that of the last window on February 8th 23:40 hours.

It should be noted that we acknowledge that LDA is not a perfect gold standard for burstiness detection. Nevertheless, we believe that it constitutes a reasonable approximation. In particular, LDA detects topics and not all topics text are bursty. For example, if a topic is constant through the entire dataset, then it is not bursty (i.e. fan conversations of celebrities such as Justin Bieber). Also, not all bursts constitute topics, for example random hashtags which interconnect otherwise unrelated messages (i.e. the popular hashtag `#fail` is put at the end many messages which depict failed situations).

In addition, we study an interesting event on February 6th, 2011, due to the Super Bowl XLV. We observe that keywords related to this event rapidly reach the top positions in the *word rank* list (see Table 3). Interestingly, before the Super Bowl started, a soccer match between Corinthians and Palmeiras was played in Brazil, which is also shown in the top keywords. Also, keywords related to Chelsea and Liverpool soccer match displayed burstiness too. On February 3rd, 2013, the SuperBowl XLVII event occurs as well, and specific keywords related to the event appear highly ranked (Black-out, 49ers, Ravens, among others).

Table 3 and 4 show the top-10 keywords of a specific time-window, and a random sample of tweets from this time period containing some of these keywords. Note that the tweets do not include all of the keywords which are listed in the first column. In this example some tweets and keywords contain noise, but this is normal when important events occur (e.g massive sport matches, concerts, and other public events).We list keywords ranked according to highest burstiness.

4.2.2 Experiment Repeatability

We used a public dataset, openly available libraries and an easy to acquire computer to make the experiment repeatable. In this experiment we used a Home Personal Computer (PC) with Core2Quad Q6600 2.4Ghz Intel Processor with 4 cores, 8 GB in RAM using Linux Ubuntu x64 11.10 version as Operative System. The pro-

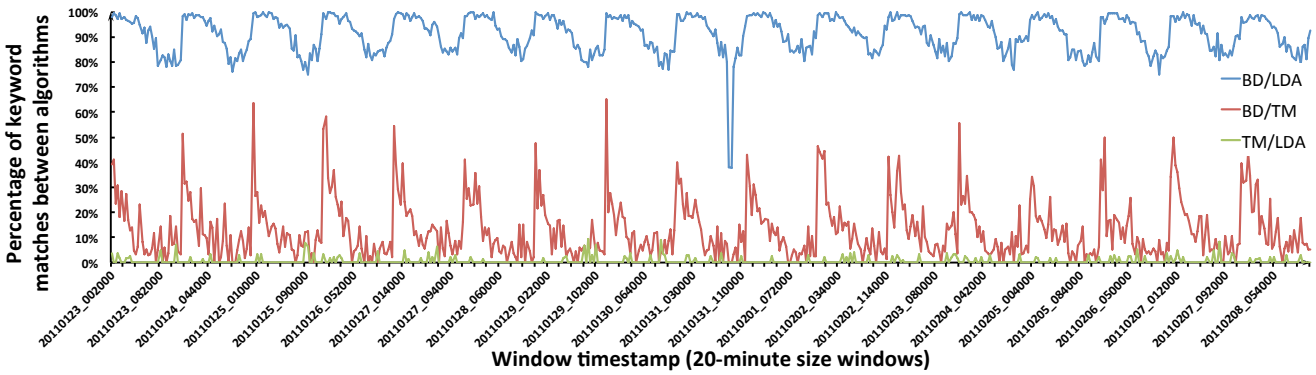

Figure 7: Comparison of the Window Variation Keyword Burst Detection algorithm with TwitterMonitor and LDA

Table 4: Examples of bursty keywords detected for the SuperBowl 2013 event, February 3rd and 4th, 2013

Window Time	Top 10 Keywords	Tweet Examples
Feb 3rd 21:20 GMT	ravens, beyonce, 49ers,#superbowlsunday, @mrbbutterfield, forward, hours, #superbowlxlvii, xlvii, halftime	• @ArianaGrande: @glitteryariana: @ArianaGrande 49ers or ravens? Beyonce lol made my day. • RT @NiallOfficial: Who do I follow in superbowl tonight american fans? 49ers or ravens? I don t know much about american football • 49ers: 27 Ravens: 24 Goes into quadruple OT. Monday declared national holiday.
Feb 3rd 23:40 GMT	ravens, touchdown, commercial, #thekiss, kaepernick, commercials, boldin, godaddy, tackles, @godaddy	• RT @lifestylist: #TheKiss? The Worst! @GoDaddy #NotBuyingIt. Wish they d spend that money on customer service instead of wasting it on a • RT @PiperJones: That GoDaddy commercial was just . . . ew.
Feb 4th 01:40 GMT	lights, power, outage, superdome, 49ers, ravens, turned off, stadium, luz, #lightsout	• RT @YourAnonNews: NFL can control everything except power in Superdome. What an embarrassment. 7 mins and counting. • RT @ShamelessProbs: The lights went out because we don t need football after Beyonce. • RT @billmarchi13: @TheShieldWWE turned off the lights and are making their way to take out #Flacco! @WWE #Superbowl

gramming language used to implement the algorithms was mainly Java. For implementing algorithm LDA we used the OpenSource GibbsLDA C++. The Dataset can be obtained upon request and must be acquired via NIST and downloaded using a crawler or an API on Twitter. We also will provide on request, code implementation of our algorithm for research purposes.

4.2.3 Scalability

After the process of a window by the Keyword Ranker Module, the entire system can be conceptually represented as a Node for a MapReduce Schema, using a specific term as a Key from merge data between other nodes in parallel processing. When we use various instances of the system as nodes, each one would have their own HashTable, and it is necessary to merge the sorted results of each one efficiently adding the frequencies of each repeated terms from all nodes, The computational cost of this procedure would take $O(n\log(n))$ using a variation of the merge step in MergeSort or much better a parallel variation of it. Once the hash tables data is merged, the method must recalculate the additional rates (arrival rate, relevance and variations). Thus, we will obtain a global result.

5. CONCLUSION

We have introduced a novel approach for on-line bursty keyword detection on the text streams. Our approach requires only the setting of the $window\_size$ parameter. It is efficient in the use of resources with a complexity of $O(n\log n)$ which makes the method *scalable*. This makes our approach easy to use and promising for on-line processing in comparison to other state-of-the-art methods. Experimental results indicate that our algorithm can scale to high tweet arrival rates while still producing high-quality results. Overall, our system produces an extraordinary keyword overlap against LDA, using very limited resources and memory.

In addition, we have presented an in-depth analysis of the behavior of *stopwords* on the Twitter stream and how to identify them. We also explain and justify the values for the parameters for our algorithm.

Future work is directed at identifying topics for fast and efficient detection of semantically coherent trends on Twitter.

Acknowledgements

The authors thank Michael Mathioudakis for his guidance in our implementation of the TwitterMonitor algorithm. We also thank Jorge Perez from the University of Chile for his valuable comments. Jheser Guzman was supported by CONICYT's Doctoral Program. Barbara Poblete was partially supported by FONDECYT grant 11121511 and Program U-INICIA VID 2012, grant U-INICIA 3/0612; University of Chile.

6. REFERENCES

[1] P. Adler and S. Kwon. Social capital: Prospects for a new concept. *Academy of management review*, pages 17–40, 2002.

[2] D. Blei, A. Ng, and M. Jordan. Latent Dirichlet Allocation. *The Journal of Machine Learning Research*, 3:993–1022, 2003.

[3] J. Bollen, H. Mao, and X. Zeng. Twitter mood predicts the stock market. *Journal of Computational Science*, 2011.

[4] S. Gaito, M. Zignani, G. P. Rossi, A. Sala, X. Wang, H. Zheng, and B. Y. Zhao. On the Bursty Evolution of Online Social Networks. *arXiv preprint arXiv:1203.6744*, 2012.

[5] K. Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21, 1972.

[6] S. Kellert. *In the wake of chaos: Unpredictable order in dynamical systems*. University of Chicago press, 1993.

[7] K. Kireyev, L. Palen, and K. Anderson. Applications of topics models to analysis of disaster-related Twitter data. In *NIPS Workshop on Applications for Topic Models: Text and Beyond*, 2009.

[8] J. Kleinberg. Bursty and hierarchical structure in streams. *Data Mining and Knowledge Discovery*, 7(4):373–397, 2003.

[9] D. Knuth. The art ofcomputer programming, vol. 3: Sorting and Searching. *Reading, MA: Addison-Wesley*, 19:496–503, 1973.

[10] H. Kwak, C. Lee, H. Park, and S. Moon. What is Twitter? a social network or a news media? In *Proceedings of the 19th international conference on World Wide Web*, pages 591–600. ACM, 2010.

[11] V. Lampos, T. De Bie, and N. Cristianini. Flu detector-tracking epidemics on Twitter. *Machine Learning and Knowledge Discovery in Databases*, pages 599–602, 2010.

[12] J. Leskovec, L. Backstrom, and J. Kleinberg. Meme-tracking and the dynamics of the news cycle. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 497–506. ACM, 2009.

[13] C. Lin, B. Zhao, Q. Mei, and J. Han. Pet: a statistical model for popular events tracking in social communities. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 929–938. ACM, 2010.

[14] M. Mathioudakis and N. Koudas. Twittermonitor: Trend detection over the Twitter stream. In *Proceedings of the 2010 international conference on Management of data*, pages 1155–1158. ACM, 2010.

[15] M. Naaman, H. Becker, and L. Gravano. Hip and trendy: Characterizing emerging trends on Twitter. *Journal of the American Society for Information Science and Technology*, 2011.

[16] X. Phan and C. Nguyen. Gibbslda++, a C/C++ Implementation of Latent Dirichlet Allocation (LDA) using Gibbs Sampling for parameter estimation and inference. `http://gibbslda.sourceforge.net/`.

[17] T. Sakaki, M. Okazaki, and Y. Matsuo. Earthquake shakes Twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web*, pages 851–860. ACM, 2010.

[18] H. Sayyadi, M. Hurst, and A. Maykov. Event detection and tracking in social streams. In *Proceedings of International Conference on Weblogs and Social Media (ICWSM)*, 2009.

[19] B. Sharifi, M. Hutton, and J. Kalita. Summarizing microblogs automatically. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 685–688. Association for Computational Linguistics, 2010.

[20] K. Starbird, L. Palen, A. Hughes, and S. Vieweg. Chatter on the red: what hazards threat reveals about the social life of microblogged information. In *Proceedings of the 2010 ACM conference on Computer supported cooperative work*, pages 241–250. ACM, 2010.

[21] R. Swan and J. Allan. Extracting significant time varying features from text. In *Proceedings of the eighth international conference on Information and knowledge management*, pages 38–45. ACM, 1999.

[22] R. Swan and J. Allan. Automatic generation of overview timelines. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 49–56. ACM, 2000.

[23] S. Vieweg, A. Hughes, K. Starbird, and L. Palen. Microblogging during two natural hazards events: What Twitter may contribute to situational awareness. In *Proceedings of the 28th international conference on Human factors in computing systems*, pages 1079–1088. ACM, 2010.

[24] J. Weng, Y. Yao, E. Leonardi, and F. Lee. Event detection in Twitter. In *Fifth International AAAI Conference on Weblogs and Social Media*, 2011.

[25] J. Yang and J. Leskovec. Patterns of temporal variation in online media. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 177–186. ACM, 2011.

[26] X. Zhang, H. Fuehres, and P. Gloor. Predicting stock market indicators through Twitter, I hope it is not as bad as I fear. *Procedia-Social and Behavioral Sciences*, 26:55–62, 2011.