Online Burst Detection Over High Speed Short Text Streams

Zhijian Yuan¹, Yan Jia², and Shuqiang Yang³

Computer School, National University of Defense Technology, Changsha, China
yuan.georgie@gmail.com,
jiayanjy@vip.sina.com,
sqyang9999@126.com

Abstract. Burst detection is an inherent problem for data streams and it has attracted extensive attention in research community due to its broad applications. In this paper, an integrated approach is introduced to solve burst events detection problem over high speed short text streams. First, we simplify the requirement by considering burst event as a set of burst features, then the processing speed can be accelerated and multiple features can be identified simultaneously. Second, by using the ratio of the number of documents with specific feature and total number of documents during a period of time as the measurement, our solution adapts to any kind of data distribution. Then we propose two algorithms to maintain the ratio in the sliding window. Finally, we propose a burst detection algorithm based on Ratio Aggregation Pyramid (RAP) and Slope Pyramid (SP) data structure, which are extended from Aggregation Pyramid (AP). Our algorithm can detect burst in multiple window sizes simultaneously and is parameter-free. Theoretical analysis and experimental results verify the availability, efficiency and scalability of our method.

1 Introduction

Web and wireless applications such as BBS, forum, BLOG and SMS have brought massive text data, which occurred as high speed short text streams for its strong time sensitivity and short in length. One of the most important mining tasks is find the abrupt ‘hot’ events (also called burst events) from these data. Burst, means abnormal number of data (such as document related to a certain topic) emerge during a period of time. Finding burst occurring time and duration is our goal. Burst detection widely used in web topic mining, commercial information acquisition, astronomical observation, network monitoring, and fraud detection etc. Moreover, in many cases, multiply windows size should be monitored real time, simultaneously and distribution of text streams is not uniform, these take more challenge.

There already have some works deal with burst detection problem. Most of them are focused on numeric streams[1-4], [1] develop an efficient algorithm for detecting adaptive aggregation bursts in a data stream given a burst ratio. [2, 4] solve the elastic burst detection problem which allow identifying bursts in multiple window sizes simultaneously, [3] introduce Lasting Factor and Abrupt Factor in the general definition

of bursts. All of above works could not apply to text directly. There also have some works focused on text streams[5-7], [5] model text stream as an infinite-state automaton, and model bursts as state transitions, [7] proposed a feature-pivot clustering approach to detect bursty events based on the feature distributions. [5-7] don't deal with the performance and precision problem that can not solve our problem properly. All of the related works will be discussed detailed in section 2.

There are two kinds of method to find burst events in text streams; one is using clustering to group similar texts together, like K-Means, and identifies each cluster has burst or not, but no appropriate clustering algorithms can group short text streams fast and exactly. Another way is consider burst event as burst features for an event always show as a set of burst features. For example, the event "SARS" consists of the features "Outbreak", "Atypic", "Respire". Currently, it has no work can deal with multiple features and multiple window sizes simultaneously.

Our contributions can be summarized as follows:

1. We consider burst event as a set of burst features and give an integrated solution to detect bursts on multiple features simultaneously.
2. We adopt the ratio of the number of specific feature and total number of documents in a period of time as the measurement of the feature, so that our solution can adapts to any kind of data distribution density. We also propose two algorithms to maintain the ratio in the sliding window.
3. We propose a burst detection algorithm using Ratio Aggregation Pyramid and Slope Pyramid data structure, which extend Aggregation Pyramid data structure. Our algorithm can detect burst in multiple window sizes simultaneously and is parameter-free for each window.

2 Related Work

Burst Detection on data streams[8] and time serials has broad applications, including Gamma Ray[4], stock, Email[5], Blog[6, 9, 10], Instant Message, News[7] and so on. Many efficient algorithms have been developed.

[5]studied the bursty and hierarchical structure in temporal text streams to discover how high frequency words change over time. The text stream is modeled as an infinite-state automaton, where bursts are modeled as state transitions. Since the algorithm in [5] could not be applied to document streams where the distribution of documents was not uniform, [6]extended[5] algorithm and proposed a method to deal with Blog and BBS which do not exhibit uniform distribution. [5, 6] focused on modeling the bursty behaviors and our goal is finding an efficient solution to solve the burst detection problem in short text streams.[7] proposed a novel parameter-free probabilistic approach, called feature-pivot clustering. [7] tried to detect bursty events based on the feature distributions. [7] did not give the way to deal with multi-feature simultaneously. None of [5-7] can detect burst in multiple window sizes.

Detecting burst across multiple window sizes, the naive algorithm can only check each window size of interest once at a time. It requires $O(kN)$ time. (k is windows number, N is the length of sequence). [4] presented a general data structure called Shifted Binary Tree for detecting interesting aggregates over such elastic windows in nearly linear time. However, this algorithm is not efficient when the probability of bursts is

high frequency. As a result, [2] improved [4] 's data structure and algorithm, It included a family of data structures including Aggregation Pyramid, Shifted Aggregation Tree and a heuristic search algorithm to find an efficient structure when given the input. An aggregation pyramid is an N -level isosceles triangular-shaped data structure built over a time window of size N . Level 0 has N cells and is in one-to-one correspondence with the original time series. Level h has $N-h$ cells, the i^{th} cell stores the aggregate of the $h+1$ consecutive data in the original time series starting at time i . The problem in [4] and [2] is that they require many parameters - burst threshold for each window. Thus, it is difficult to find an effective way to tune these parameters. Our methods use ratio threshold for bursting measurement and only need two parameters.

[1] introduced a novel concept of adaptive aggregation burst to detect double-side relative bursts dynamically. [1] argued that ratio threshold for bursting measurement and adaptive window size was more suitable for data stream analysis applications. However, these methods can not deal with multiple window sizes and can not be applied to text streams directly while ours can.

3 Algorithms for Burst Detection

3.1 Problem Statement

Given a text stream $X_1, X_2, X_3, \dots, X_n$, an aggregation function F , S_t and S_t' are the latest two consecutive subsequences with the same length of t , $1 \leq t \leq n/2$. When X_n comes, $\forall j \in 1..i$, $\beta_i > 1$, if $F(S_t) > \beta_i F(S_t')$, then an increasing burst occurs on t -length window, $\forall j \in 1..i$, $0 < \beta_d < 1$, $F(S_t) < \beta_d F(S_t')$, then a decreasing burst occurs. β_i is the increasing threshold and β_d is the decreasing threshold. The problem of burst detection is finding the value pair (X_n, j) , indicating that an increasing burst or a decreasing burst occurs in this window.

3.2 A Framework to Solve This Problem

As we discussed in section 1, using clustering to group short text streams is slow and inaccurate, so we adopt the second method. We use simplified vector to represent short text data. Data stream is changing with time, so we should choose a relative value rather than an absolute term as a metric and algorithm could adapt to different data distribution density. In this paper, we use the ratio of the number of specific feature and total number of documents to measure the relative quantity of the feature.

Algorithms to judge whether there are bursts in data streams can be classified into two categories: one is based on the absolute value, called threshold value, namely if the aggregation value is beyond the scope, a burst is announced. This method is simple and intuitionist. However, the disadvantage is that it must assign a threshold to each window size. Another is based on the relative value, called ratio value, namely if the ratio of two neighboring windows is beyond the scope, a burst is announced. In this method, each window sizes and data distribution densities can use the unified parameter. Our work adopts the second method and eliminates the limitation of current algorithms, which cannot deal with multiple time-windows simultaneously.

Our framework is outlined in Fig. 1. There are three major steps:

1. Transforming text stream to simplified vector.
 2. Aggregating multi-vectors to a single vector, which is the basic element of the sliding window.
 3. Based on the above 2 steps, updating the Ratio Pyramid Aggregation and Slope Pyramid Aggregation data structure, and detecting burst for each feature.
- The follow sections will discuss each step in details.

Fig. 1. The flow chart for solving the burst detection problem in short text streams

3.3 Forming Single Element in Sliding Window

Definition 1. Features set $H = \{(index, name) | 1 \leq index \leq N, name \in Vocabulary\}$, $N = |Vocabulary|$, $Vocabulary$ is appointed by domain expert.

Definition 2. Text stream $D = \{D_1, D_2, D_3, \dots, D_i\}$, Document D_i could be denoted by $D_i = \{(index_m, count(index_m)) | name_m \in Vocabulary(D_i)\}$.

After Splitting words, data cleaning, counting features and labeling, short text is transform to simplified vector. The length of vector is $|Vocabulary(D_i)|$.

It may improper to treat one document as basic element in sliding window for the high speed of text stream. We should specify a time period T . All the vectors in T are aggregated as one vector. As we know, data stream fluctuate with time. Therefore, choosing a relative value instead of an absolute value to measure the value of feature can make our algorithms suitable for different data distribution density. In this paper, we choose the **Ratio** as our metric, defined as:

$$Ratio(Feature) = \frac{Count(Feature)}{Count(Document)} \quad Feature \in H.Name$$

Obviously, These will increase the complexity from $O(n)$ to $O(kn)$, k is a constant.

There are two kinds of algorithms to compute *Ratio*: one is online algorithm. As soon as one vector comes, each item in this vector is analyzed. The other is batched process algorithm, which saves the coming element in *T* and processing together. The flow of these two algorithms list here:

Table 1. Online Processing Algorithm Vs Batched Processing Algorithm

Online Processing Algorithm	Batched Processing Algorithm
Input: text vector $D_1, D_2, D_3, \dots, D_n$ in time period <i>T</i> Output: result vector <i>V</i> 1: $V \leftarrow \Phi$; 2: for $i \leftarrow 1$ to n 3: for each (Item in D_i) do 4: if (Item.index in $V.index$) then 5: $V[Item.index] = V[Item.index]$ + Item.count; 6: else 7: $V.add(Item.index,$ Item.count); 8: end if 9: end for 10: end for 11: for each (Item in V) 12: $V[Item.index] =$ $V[Item.index]/n$; 13: return <i>V</i> ;	Input: text vector $D_1, D_2, D_3, \dots, D_n$ in time period <i>T</i> Output: result vector <i>V</i> 1: $V \leftarrow \Phi$; 2: for $i \leftarrow 1$ to n 3: Cursor(i) $\leftarrow 1$; 4: Item(i) $\leftarrow D_i(Cursor(i))$; 5: end for 6: repeat 7: $min \leftarrow \text{Min}(Item(i).index), 1 \leq i \leq n$; 8: minCollec- tion $\leftarrow \{j \mid Item(j).index = min\}$; 9: count $\leftarrow 0$; 10: for each (m in minCollection) do 11: count $\leftarrow$ count + Item(m).count; 12: Cursor(m) $\leftarrow$ Cursor(m) + 1; 13: Item (m) $\leftarrow D_m(Cursor(m))$; 14: end for 15: $V.add(\text{MinIndex}, \text{tempCount})$; 16: until Cursor(i)=EOF(D_i) $1 \leq i \leq n$; 17: return <i>V</i> ;

3.4 Data Structures and Algorithms for Burst Detection

Ratio Aggregation Pyramid

Our Algorithms for burst detection are based on the data structure called Ratio Aggregation Pyramid and Slope Pyramid. These two data structures are derived from Aggregation Pyramid, as we introduce above. In computing the cell's value of $1 \sim N-1$ level, an aggregation function *F* must be specified, commonly is **SUM**. Here, because the element in sliding window is *ratio*, we define an aggregation function F_R , which can be calculated as follows:

$$F_R(Ratio_1, Ratio_2) = \frac{Count(Term_1) + Count(Term_2)}{Count(Document_1) + Count(Document_2)}$$

We treat the value of this function as the basic element in the sliding window. We call the Aggregation Pyramid (AP) with aggregation function F_R as Ratio Aggregation Pyramid (RAP).

Fig. 2 is an example of AP and RAP. We suppose that the feature number in the sliding window is $\{1,4,0,3,2,3\}$ and $\{1/105, 4/354, 0/98, 3/209, 2/117, 3/533\}$. We can calculate the value of the entire cell in AP and RAP.

Fig. 2. Comparison of Classical Aggregation Pyramid and Ratio Aggregation Pyramid

Slope Pyramid

Algorithms to judge whether there are bursts in the aggregation tower are divided into two kinds: one kind uses absolute value, called threshold value, namely if cell value beyond the scope, then recognized this is a burst. This method is simple and direct-viewing and the shortcoming is that a threshold must be given to each window size (level). Another kind uses relative value, called ratio value, namely each time calculates the ratio of two neighboring windows, if the ratio beyond the scope, then recognized this as a burst. This ratio is equal to the slope. All windows size and data distribution density can use the unified parameter. This article uses the second method. The formula to compute Slope Pyramid is based on Ratio Aggregation Pyramid. It can be present as:

$$\text{Slope}(h) = \text{cell}(h,t)/\text{cell}(h,t-h-1)$$

Slope Pyramid only has one cell on one level. This cell stores the ratio of the latest two windows. Fig. 3 demonstrates how to compute the value of all the cell value in the Slope Pyramid.

Fig. 3. Calculating Slope Pyramid through Ratio Aggregation Pyramid

Data Structure Updating and Burst Detection

When a new element is coming to sliding window, Ratio Aggregation Pyramid and Slope Pyramid update timely and identify if the value of the cell in Slope Pyramid is beyond the scope. If it is true, record the sliding window and the burst feature.

Formula for updating RAP is shown as following:

$$\text{cell}(h, t) = F_R(\text{cell}(h-1, t-1), \text{cell}(0, t))$$

$\text{cell}(h, t)$ means that the cell is in level h and in the end of time t .

After RAP has been updated, SP can be calculated and be checked if the value of the entire cell is in the normal scope. SP only needs to be calculated to level $\lfloor t/2 \rfloor$.

Fig. 4 demonstrates how to update RAP when a new element is coming.

Fig. 4. Updating for Ratio Aggregation Pyramid

4 Experimental Evaluation

We have collected one-year (from 2005-01-01 to 2005-12-31) short documents (2,503,385 text items) containing news, forum and BBS from a web site. We implement our framework using JavaTM and Eclipse 3.0.1. Our experiments are performed on Inter Pentium4 2.4 HZ PC with 1.0G main memory. From all the features, we select eight burst features for demonstration, which is shown in Fig. 5.

Fig. 5. The x-axis is the date starting from Jan. 1st, 2005 to Dec. 31st, 2005. The y-axis is the feature number in that day.

We first evaluate the efficiency of RAP. We have tested three groups of thresholds. The window size of each tested group is 4, 8, 12, 16 and 32. The experiment results are shown by Fig. 6. We can see that the burst number detected by RAP is only a little bit less than AP. This is because the fluctuation of the data distribution is small.

Fig. 6. Burst number detected based on AP and RAP with various thresholds

We also compare the executing time of RAP with AP. As is shown in Fig. 7, RAP need more time than AP. That is because RAP takes more time in updating ratio aggregation pyramid than that of AP does. Furthermore, RAP needs additional time to compute SP.

Finally, we evaluate the effects of thresholds to burst number. Fig. 8 reveals that increasing the *increasing threshold* and decreasing the *decreasing threshold* can reduce the number of the burst. For not all burst are significant, so users can adopt an appropriate threshold to get the demanding burst number.

Fig. 7. processing time between AP and RAP

Fig. 8. Burst number detected with various thresholds and window sizes

5 Conclusion and Future Work

In this paper, we have proposed a simple but efficient solution for the burst events detection problem. Experimental results show that our algorithm is effective. It performs better when processing object is short text and distribution fluctuates greatly. However, many works can be improved. When generating simplified vector, we can establish the concept of semantic space models, giving synonymous with the terminology, so that the number of words and vector dimensions can be reduced, hence enhancing the efficiency and accuracy of algorithms. When the data stream flows are increasing, the usage of sampling and how to use the outline structure for high-speed data flows become open problems, which are left as our future work. In addition, burst forecasting and good visualization are also attractive.

References

1. Zhou Aoying, Qin Shouke, and Qian Weining. Adaptively Detecting Aggregation Bursts in Data Streams. in Lecture Notes in Computer Science. 2005.
2. Xin Zhang and Dennis Shasha. Better Burst Detection. in the 22nd International Conference on Data Engineering. 2006. Atlanta, Georgia, USA.

3. Tingting Chen, Yi Wang, Binxing Fang, and Jun Zheng. Detecting Lasting and Abrupt Bursts in Data Streams Using Two-Layered Wavelet Tree. in International Conference on Internet and Web Applications and Services/Advanced International Conference on Telecommunications. 2006.
4. Yunyue Zhu and Dennis Shasha. Efficient elastic burst detection in data streams. in Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining. 2003. Washington, D.C.: ACM Press.
5. Jon Kleinberg. Bursty and hierarchical structure in streams. in Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining. 2002. Edmonton, Alberta, Canada: ACM Press.
6. Toshiaki Fujiki, Tomoyuki Nanno, Yasuhiro Suzuki, and Manabu Okumura. Identification of Bursts in a Document Stream. in First International Workshop on Knowledge Discovery in Data Streams. 2004. Pisa, Italy.
7. Gabriel Pui Cheong Fung, Jeffrey Xu Yu, Philip S. Yu, and Hongjun Lu. Parameter free bursty events detection in text streams. in Proceedings of the 31st international conference on Very large data bases. 2005. Trondheim, Norway: VLDB Endowment.
8. Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and issues in data stream systems, in Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems. 2002, ACM Press: Madison, Wisconsin.
9. Ravi Kumar, Jasmine Novak, Prabhakar Raghavan, and Andrew Tomkins. On the bursty evolution of blogspace. in Proceedings of the 12th international conference on World Wide Web. 2003. Budapest, Hungary: ACM Press.
10. Tomoyuki Nanno, Toshiaki Fujiki, Yasuhiro Suzuki, and Manabu Okumura, Automatically collecting, monitoring, and mining japanese weblogs, in Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters. 2004, ACM Press: New York, NY, USA.