Event Detection from Social Network Streams Using Frequent Pattern Mining with Dynamic Support Values

Nora Alkhamees
School of Computer Science and Electrical Engineering
University of Essex
Colchester, UK
Email: nyaalk@essex.ac.uk

Maria Fasli
School of Computer Science and Electrical Engineering
University of Essex
Colchester, UK
Email: mfasli@essex.ac.uk

***Abstract*—Detecting events from streams of data is challenging due to the characteristics of such streams: data elements arrive in real-time and at high velocity, and the size of the streams is typically unbounded while it is not possible to backtrack over past data elements or maintain and review the entire history. Social networks are a good source for event identification as they generate huge amount of timely information representing what users are posting and discussing. In this research, we are developing methods for event detection from streams of data. More specifically, we are presenting a framework for detecting the daily occurring events or topics occurring in social network streams related to major events. Our approach utilizes the Frequent Pattern Mining method to detect the daily occurring frequent patterns, which are going to be our detected events. In addition, we propose a dynamic support definition method to replace the fixed given one. An experiment was run on two streams relating to two different major events to examine the detected events and to test our support definition method. The UK General Elections 2015 stream holds more than one million tweets, and the Greece Crisis 2015 stream contains more than 150k tweets. The detected events were evaluated against news headlines published the same day the event was found. The results revealed that the higher the streaming level (bigger window size), the more accurate the detected events. We also show that for too small sized windows, a more strict support definition method is needed to avoid detecting false or insignificant events.**

***Keywords*-Stream Reasoning, Frequent Pattern Mining, Event Identification from Social Network Stream.**

I. Introduction

In the current era of social networks, almost everyone has now become a virtual broadcaster sharing incredible number of messages, especially these days with the widespread use of smart phones and availability of mobile networks. The Pew Research Centre survey on social network users [1] showed that three out of four people online use social networks. Even more surprising, it showed that not only young people are increasingly using social networks but the number of people of age 65 and older has more than tripled since 2010. Mainly in this research we are interested in benefiting from the availability of data in social network streams to develop methods for detecting and identifying events.

Figure 1. Public Tweets

Figure 2. News Tweets

Sometimes events spread faster in social media rather than other forms of media, and they are widely discussed there. This is clearer in unplanned events than already known and planned ones. Consider for example the shooting event that happened in Ferguson, Missouri, USA which turned into national and international news with 3.6 million tweets posted from the 9th of August (the day of shooting) until the 17th of August. The first story emerged on Twitter before any other news channel [1]. Furthermore, when tracking events published in social networks, one can sense people's reaction towards the event in comparison to news wires which are more formal and objective, refer to figures 1 and 2 for a comparison between tweets posted from news and from public users regarding the Ferguson shooting event.

An event is defined as a real world occurrence of something with an associated place and time period [4]. Event identification is the process of looking for events.

Each day we are interested in finding all frequent patterns that are eligible for being events by applying a Frequent Pattern Mining (FPM) method. In order to run any FPM algorithm, the support threshold which is a metric for

[1]http://www.pewresearch.org/fact-tank/2014/08/20/cable-twitter-picked-up-ferguson-story-at-a-similar-clip/

item retrieval must be specified in advance. Any item with frequency greater than or equal to the minimum support value is retrieved. However, using a fixed support value with frequent pattern mining to detect events may not be efficient especially in changeable and dynamic environments such as microblogs and social networks, where the number of daily posts is not fixed or predetermined. Hence, a dynamically defined support value is more appropriate as it can be calculated based on each day window size and occurrence of keywords.

In this work, and using a social network stream, we want to develop methods for detecting the daily occurring events if any. Using a FPM method with a dynamic calculated support value, we want to identify the frequent patterns which are going to be our detected events. Typically, from every day window-batch consisting of current day text-posts we have to calculate its support value, and then use that support value to detect events. The support is a critical value, an accurate definition of it can lead to accurate and valid identified events. Moreover a low support value may lead to mixed topics or events with too many selected terms, while on the other hand, with high support values fewer terms are selected which might lead to very generic topics or events. The support is calculated based on keywords occurrence and frequency. Experiments were applied to evaluate our event identification framework on a Twitter stream for the UK General Elections (GE) 2015 event and the Greece Crisis 2015 event. Furthermore, the detected events were evaluated against what was published in news headlines in the World Wide Web regarding the under study event. If a detected topic or event was found the same day a news headline is published, then it is said a true event, and an insignificant or false event otherwise.

Our motivation is to provide an event detection method that is capable of identifying events regardless of the window size, using a dynamically calculated threshold based on people's engagement with a major event to replace the fixed given one. This framework can be used by any person or software that is interested in knowing the effect and growth of major events to discover its source topics/events.

The rest of this paper is organized as follows. Next we briefly describe the related work. The following section discusses social network stream analysis and event identification followed by the description of the support definition method, which is the core of this paper. The subsequent section presents the experimental evaluation, analysing and discussing both the experiments conducted and the findings. The paper ends with a summary and the conclusions.

Table I
TRADITIONAL VS STREAM REASONING

Traditional Reasoning	**Stream Reasoning**
Analysis in that case is applied to all available data.	Analysis is hard to be applied to all data (streams are unbounded), so the idea of sliding window was invented.
Processing here starts when a query is fired and ends when either result is found or all data is scanned.	Stream reasoning requires continuous processing it never ends.

II. RELATED WORK

A. Stream Reasoning

A data stream is a continuous sequence or flow of data over time in real-time and often at high velocity [2]. Data streams produce a huge amount of data, and can come from different sources: sensors, website click streams, financial markets, social networks, etc. An example of such streams is the posted tweets in Twitter.com. Twitter is an online social networking service that enables users to send and receive short 140-character message posts called "tweets". An average of 6000 tweets tweeted every second in Twitter [2]. In a data stream, the data items are usually ordered by time-stamp value and in most cases are processed on arrival. The data items in data streams can be either structured data (e.g., temperature data, tick prices); semi-structured data (e.g., HTTP log streams); or unstructured data (e.g., emails, tweets).

With the huge growth in computational devices such as embedded processors as well as applications that consume and produce data in real time, there is an increasing need for processing and reasoning methods that are able to deal with continuous streams of data and increasingly so, from multiple streams. Stream reasoning was first explored by Della Valle et al. in 2009 in [3]. They described stream reasoning as "an unexplored, yet high impact, research area", where reasoning is applied in real time to noisy data streams, so as to support decision making. Table I shows a comparison between traditional reasoning and stream reasoning on how analysis is performed and when it starts and terminates.

Existing stream processing systems fall under two categories: (i) Data Stream Management Systems (DSMS), which are part of the database community, or (ii) Complex Event Processing (CEP) systems, which are part of the event based community [4]. DSMSs [5] inherit the relational data model and an expressive query language typical of Data Base Management Systems (DBMS). Such systems use the Continuous Query Language (CQL), where queries are continuously running. CEPs on the other hand are commonly used in event driven environments, where a time element must be embedded in the stream, the main idea is continuous

[2]http://www.internetlivestats.com/twitter-statistics/

processing to detect occurring events [6]. Instead of queries, rules are defined to indicate the occurrence of an event using the Event Processing Language (EPL).

B. Topic Detection Methods

Topic detection methods are classified to be under three major categories [7]: Document-pivot methods, Feature-pivot methods, and Probabilistic topic methods.

1) Document-Pivot: In Document-pivot methods, topics are represented as a set of related documents. Typically such methods compute similarity between either pair of documents or between a document and a cluster. Document-pivot approaches differ mainly in calculating the similarity. For instance, in [8] similarity calculations were based on Term Frequency-Inverse Document Frequency (TF-IDF) weighting scheme. It compares the TF-IDF score of the incoming tweet with the TF-IDF score for the first tweet in each cluster along with the TF-IDF score for the most common words in that cluster. The comparison results in either adding that tweet to the best matching cluster or creating a new cluster. Petrovic et al. in [9] proposed a modification on the Locality Sensitive Hashing (LSH) approach to be able to find the best matching similar document in a faster way. In another approach [10], a graphical model named Location-Time Constrained Topic (LTT) was proposed, which identifies the content, time, and location of each posted social post. As a result, a post is represented as a probability distribution and the similarity between two posts is calculated based on the distance of their distributions.

2) Feature-Pivot: The feature-pivot approach mainly focuses on grouping terms based on their occurrence to represent a topic. It is a two-step approach, starting with selecting targeted terms based on their frequency or burstiness, then clustering terms based on some inter-term similarity. Feature-pivot approaches differ in terms of selection criteria. For instance, in [11] term selection was based on an "energy" measure of a term. The term's "energy" is calculated based on both term frequency and the importance of the user who posted the post. Depending on its "energy", a term is clustered using a graph based algorithm to detect events. Another approach in [12] is the Event Detection with Clustering of Wavelet-based Signals (EDCoW), which selects terms by applying wavelets analysis based on term frequency, then clusters these terms based on a modularity-based graph partitioning technique to represent an event. The Frequent Pattern Mining (FPM) [13] has been used in feature-pivot approaches to measure the co-occurrence of any number of terms instead of a pairwise terms co-occurrence. Some efforts using the FPM to detect topics are [14]–[16].

3) Probabilistic Topic Model: The probabilistic topic model treats the topic detection issue as a probability inference problem. A topic is represented as a distribution between both terms and documents. The most well-known probabilistic topic model is the Latent Dirichlet Allocation (LDA) [17], where documents are represented as mixture of latent topics. The learning and interpretation in LDA is naturally done using Variational Bayes [18] and other approaches including Gibbs Sampling [19]. A supervised version of the traditional LDA was proposed such as in [20].

C. Frequent Pattern Mining

Frequent patterns are itemsets or transactions that occur in a dataset with frequency no less than a predefined threshold called "support". Frequent Pattern Mining was first proposed for the market basket analysis science by Agrawal et al. [21]. It analyses a customer's shopping basket to find association between customers and their bought items. FPM introduces three basic frequent itemset mining methodologies: Apriori, FP-Growth, and Eclat [13]. FP-Growth [22] is a frequent itemset mining method which requires less number of database scans, no candidate generation and works in a divide-and-conquer way. Furthermore it is suitable for both huge number of database transactions and for longer patterns. It is based on a compact constructed tree that is then mined to find frequent patterns instead of depending on the whole data set.

1) Frequent Pattern Mining from Online Data Streams: Online data streams share a number of features which are currently considered the stream reasoning challenges. These data streams are of unknown size, irregular data arrival rate, and a single scan is only possible without backtracks [23]. The traditional methods for mining frequent itemsets in a static DB requires a number of DB scans. However this is not applicable in online data streams due to memory and computational constraints. Therefore traditional methods cannot be directly applied on data streams [23], as it is impossible to go through the whole data stream in a single scan. According to [24] mining frequent itemsets from data streams falls in one of the following: landmark model, fading model, and a sliding window model.

A landmark model considers all data from a specified point of time a "landmark" to the present time. Usually it starts at the beginning of the stream till the current time, and it treats all data equally. A fading model works in the same way as the landmark model except it assigns different weights to data. New data transactions are given higher weights as an indicator that latest data is more important. Finally the sliding window model uses a sliding window that slides over the data stream to find frequent itemsets. This window can be either a transaction based window consisting of a fixed number of transactions or time based window of a fixed time length.

The first attempt in using the FPM to detect events from Twitter was in [14], where they used the Frequent Pattern Stream Mining Algorithm (FP-Stream) proposed by [25]. Another effort by Petkos et al. in [15] was dedicated to topic detection from Twitter. They applied a softened

version of the FPM named Soft Frequent Pattern Mining (SFPM) Algorithm. In [26], an improved version of the SFPM was devised to deal with dynamic environments. A more recent approach in [16] proposes a High Utility Pattern Clustering (HUPC) framework for detecting topics from microblog streams. It is a two-step framework, starting with detecting a representative High Utility Pattern (HUP) from the microblog stream then grouping these patterns into topic clusters.

III. Social Network Stream Analysis and Event Identification

Before describing our framework we first provide some definitions and clarifications. Let a social network stream S represent the posted text-posts at a given time stamp T. Let all recent text-posts arriving in fixed time interval (e.g., single day) represent the current window-batch S_T, and text-posts that arrived the previous time interval (previous day) represent the previous window-batch S_{T-1}. Each text-post s_i belongs to a window-batch and is represented as a bag-of-words. In addition, a window-batch size is not fixed, it may consist of any number of text-posts. Each window-batch will have a dynamically calculated support value Spp_T depending on that window-batch received text-posts. A window batch S_T may have no events detected at all $E_T = \{\phi\}$ or N events $E_T = \{E_1, E_2, ..., E_L N\}$.

An abstract model of our approach for event detection from social networks stream is shown in figure 3. We have three main components applied on each window-batch: Support definition, Detection of frequent patterns (FPs), and Post-processing. Once we have received a window-batch for the current time stamp (S_T) from our social network stream (i.e text-posts posted during the current day), we want to dynamically calculate the support value (Spp_T). In the next step, and by using the (Spp_T) value, we want to apply the FPM algorithm to identify and detect the frequent patterns from each window-batch. Finally a post-processing step is applied to the found frequent patterns in order to have more concise and compact detected events.

As the General Elections 2015 stream was bigger and with more windows than the Greece Crisis stream, we will be focusing mainly on this when it comes to showing the data stream details and some analysis snapshots.

A. Support Definition

Support definition is the core of our event identification framework. We seek to develop a calculated support value that has the potential to detect events. In particular, a low support value may lead to a huge number of detected topics or events (frequent patterns) with too many selected terms, while on the other hand with a high support value fewer terms are selected which can lead to very generic and vague or false topics or events.

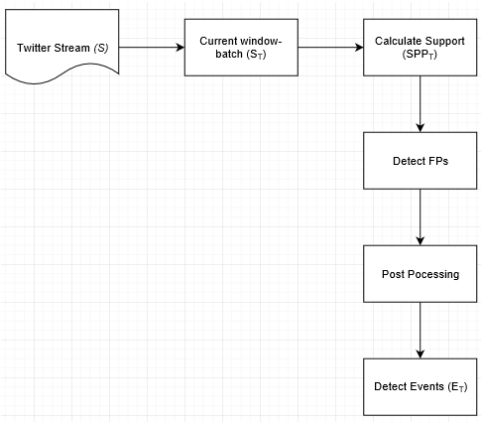

Figure 3. Event Detection Abstract Model

In FPM, the support is a metric for term selection and retrieval. Terms with frequency satisfying the support value are retrieved while others are ignored. Each day a different support value will be calculated depending on that day's posted text-posts.

We have tried different and various ways and applied various methods and techniques to calculate the support value. We seek to obtain support values that are proportional to and reflect the window size. This means a higher support value for big windows and a lower one for small sized windows. The support definition in each window is mainly focusing on terms' frequency. So, initially we start by counting the frequency of each term in all text-posts that belong to the same window-batch, in other words, we count the frequency of every term on a daily basis. We found that words occurring only once (with frequency 1) are almost equivalent to one-third of the total number of distinct words in a window-batch. As words appearing once will not help in finding the important and frequent patterns, they were removed as a preliminary step.

Depending only on the terms' frequency average in a window-batch to be a metric for term's retrieval is not enough for defining the support, the number of terms exceeding the average is still big, and because we are looking for frequent patterns that are truly frequent and are as concise as possible, it is insufficient just to adopt the terms' frequency average for defining the support. Table II shows all the 29 window-batches calculated average values, and figure 4 shows a graph presenting the number of terms exceeding the average value in each window-batch. As shown from the high number of terms satisfying the average in figure 4, yet more calculations are required to avoid having low support values. Equation 1 shows the calculation for the terms' frequency average for a single window batch, where N is the total number of words in a window. As a result, additional metrics need to be considered for defining a strict window support value.

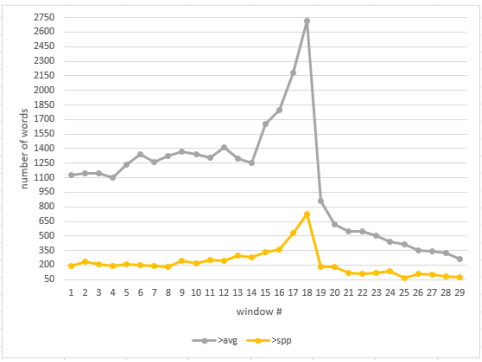

Figure 4. Number of words exceeding both the average and the support for each window

$$\bar{t} = \frac{\sum t_i}{N} \quad (1)$$

We are seeking support values that are greater than the terms' frequency average in a window and at the same time are proportional to each window size. As a result, our approach for calculating the support was to multiply each window terms' frequency average by that window's median value. The median is a statistical value that separates higher values from the lower ones, it is the middle value of a sorted list [27]. Accordingly, a list of each window's distinct words along with its frequency was sorted in descending order according to frequency to find the median value. Median values for all the 29 window-batches are shown in table II. As a result the found support values were proportional to each window size. Again refer to table II to see the calculated support value for each window batch, and to figure 4 to see a graph showing the number of words exceeding the calculated support values in comparison to the total number of words exceeding the average values. Equation 2 shows the method for the calculation of support.

$$Spp = avg(S_T) \times median(S_T) \quad (2)$$

As a result, considering the median along with the terms' frequency average for each window to define the support value, yields reasonable results in comparison to other methods that were investigated and is proportional to the size of the window.

The Algorithm for defining the support value is illustrated in figure 5, where the support calculation process for every incoming window-batch from the social network stream is shown. In lines 2 and 3, both the total and distinct number of words are counted to be used for finding the Average and Median values in lines 4 and 5. Line 6 shows the support definition equation. Each window-batch calculated support value is used for retrieving terms with frequency greater than or equal to the support.

Input: $SNstream$: Social network stream
Output: Spp: The support calculated value
1: **for** each incoming window S_T in SNstream **do**
2: $distinctwords$= count-distinct-words(S_T)
3: $totalwords$= count-all-words(S_T)
4: avg = calculateavg($ditinctwords$,S_T)
5: $median$ = calculatemed($ditinctwords$,S_T)
6: $Spp = avg \times median$
7: **return** Spp
8: **end for**

Figure 5. Defining the Support Value Algorithm

Table II
WINDOWS' DATA ANALYSIS

W#	Tweets	all-words	distinct-words	AVG	Med	Spp
1	23836	151971	6303	24.11	6	145
2	24294	150417	6364	23.64	5	118
3	27210	174002	6518	26.7	6	160
4	33622	215318	6719	32.05	6	192
5	27759	188474	7021	26.84	6	161
6	23427	151191	6660	22.7	6	136
7	22511	144524	6592	21.92	6	132
8	22032	141516	6441	21.97	6	132
9	29957	193169	7463	25.88	6	155
10	27023	176840	7035	25.14	6	151
11	35546	238658	8144	29.3	6	176
12	34281	217300	8240	26.37	6	158
13	34410	227812	8012	28.43	5	142
14	27052	172363	6896	24.99	5	125
15	57347	347061	10387	33.41	6	200
16	73571	466232	12152	38.37	6	230
17	171829	998267	18623	53.6	6	322
18	388966	2595697	28614	90.71	6	544
19	13648	88096	4958	17.77	5	89
20	7942	47426	3571	13.28	4	53
21	5545	35827	3007	11.91	4	48
22	4081	24199	2553	9.48	4	38
23	3744	21982	2411	9.12	4	36
24	2429	15194	1960	7.75	3	23
25	2262	12920	1755	7.36	4	29
26	1607	10126	1449	6.99	3	21
27	2046	13349	1486	8.98	3	27
28	2215	15308	1651	9.27	3	28
29	1699	11242	1248	9.01	3	27

Refer to table II to view the total number of words, number of distinct words, terms' average frequency, median, and the support for all the 29 window-batches.

Table III
DATA COLLECTION SUMMARY

Event	no. of tweets	no. of windows
General Elections 2015	1m	29
Greece Crisis 2015	150k	17

IV. EXPERIMENTAL EVALUATION

A. Data Collection, Description and Preparation

We want to choose a major event that is happening or is planned to happen and collect text-posts related to it from a microblog stream. Once posts are starting to be collected we aim to develop methods for identifying topics occurring or taking place within that event every day.

From Twitter stream and using the Twitter streaming API[3], which enables the extraction of tweets in real-time, based on certain query parameters like: certain keywords, certain location, etc. We queried the Twitter API for the following terms (British Elections, GE2015, VoteGE15, GE15) to collect tweets related to the UK General Elections 2015, and the following terms for the Greece Crisis 2015 (greece crisis, greece bailout, greece referendum, grexit, greece eu, greece eurozone). We got more than one million tweet for the General Elections event collected in weekdays from 9am- 5pm in the period from 15-4-2015 till 26-5-2015. While we got more than 150k tweets collected in the period from 29-6-2015 till 16-7-2015 from Twitter for 2 hours daily for the Greece crisis event, refer to table III for the data streams collection summary. The retrieved tweets were placed in a MongoDB [4], which is a NOSQL database that is capable of handling unstructured data.

As a preliminary step and before applying the FP-Growth algorithm, we tokenized every tweet text to tokens not only depending on white space but with keeping in mind mentions @, URLs and hashtags #. After that we applied some pre-processing steps such as removing punctuations, mentions, hashtags, URLs, and stop words. As we are interested in finding topics that occur every day, we chose the window model to represent our stream. The window size was defined to be a single day and depending on the date a tweet was posted it will belong to a certain window. The total number of tweets in each window is not fixed, for example from our GE stream the total number of tweets in the first day of our tweet collection, which was 16th of April 2015 is equal to 23836 tweets while on day 15 which is the 6th of May 2015 it was 57348 tweets.

B. Applying the Frequent Pattern Mining

We are using the FP-Growth algorithm which is based on constructing the FP-tree as a first step, then mining and traversing through that tree to find frequent patterns. Additionally we want to use the daily calculated support value to detect the frequent patterns from each window. So for every window-batch, we initially start by creating a list of all terms sorted by frequency. We kept terms with frequency satisfying the calculated support value and discard the rest. Then the FP-tree is constructed for each window based on terms' frequency. Lastly the constructed FP-tree is traversed to find and extract the daily occurring frequent patterns. These found patterns are assumed to be the topics or events discussed on that certain day. The FP-Growth algorithm was chosen among other FPM algorithms [13] (Apriori and Eclat) because it best suits our data stream by requiring less stream scans, and no candidate patterns are generated. The text-posts are assumed to be the transactions or itemsets, and the text-post terms are assumed to be the items. We applied the FP-Growth algorithm using the SPMF: An Open Source Data Mining Java Library [28]. A snapshot of the detected frequent patterns for a single window-batch is shown in figure 6.

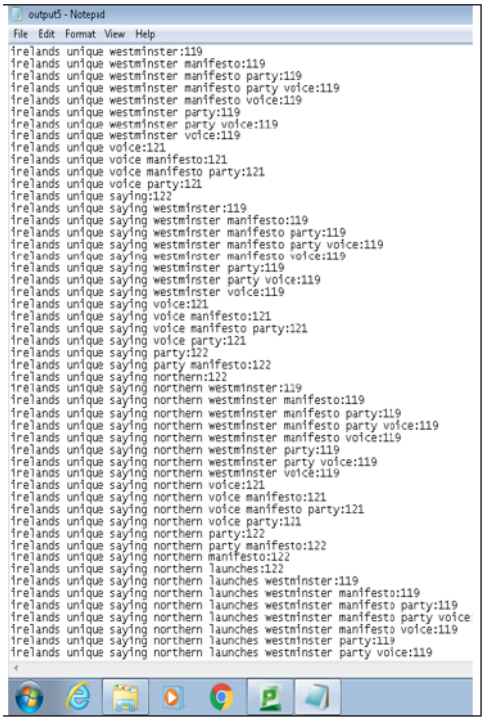

Figure 6. Found frequent patterns from a single window-batch

C. Post processing

The found frequent patterns after applying the FP-Growth algorithm with our dynamically calculated support values were partially duplicated (refer to figure 6), and some of them were irrelevant patterns. We are not interested in finding all the frequent patterns from our stream, instead we want to have patterns that are eligible for being topics

[3]https://dev.twitter.com/tags/streaming-api
[4]https://docs.mongodb.com/

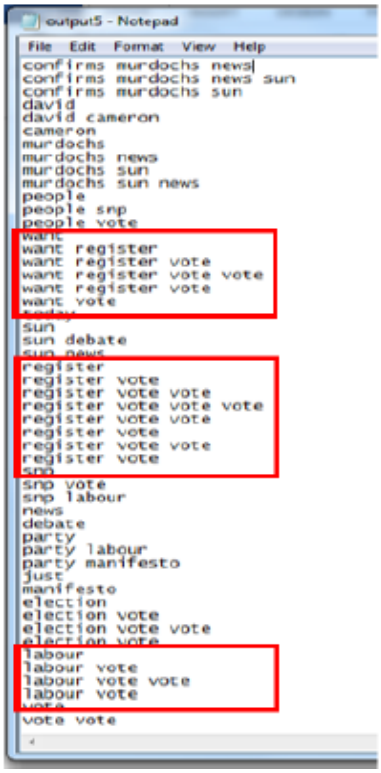

Figure 7. Non-relevant patterns from a single window-batch

or events.

In addition, some studies showed that large number of tweets associated with major events may be irrelevant or misinforming. A study of the Ferguson unrest in 2014 in [29] showed that almost 25% of posted tweets were actually rumours. Another study in [30] considering the Hurricane Sandy in 2012 showed that more than 10k unique tweets out of 1.7m tweets contained fake images. They also found that 68% of the fake tweets spreading was through retweets, and the top 30 users out of 10,215 users resulted in 90% of the fake tweets being retweeted.

As a result, we further need to consider a post processing step in order to discover more useful and compact patterns. Accordingly only patterns belonging to big branches are preserved, a branch is said to be big if its size exceeds the calculated support value for that window. Hence if the branch size exceeds the support value then it is kept, otherwise it is ignored. Applying the branch size restriction helped in avoiding non-relevant and meaningless patterns which in most cases belong to small size branches, figure 7 shows an example of some non-relevant patterns. Afterwards from the retained big branches we look for the longest pattern and omit all the subset ones to avoid repetition. Snapshots of the found frequent patterns for some windows after implementing both the branch size constraint and the omitting of subset patterns are shown in figure 8.

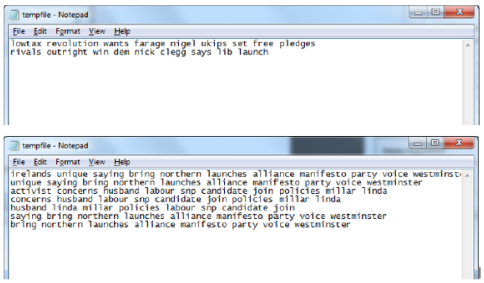

Figure 8. Snapshots of some windows frequent patterns

As some of the found frequent patterns were related to the same subject and discuss the same topic, we had to add a similarity metric to choose a representative pattern instead of having them all. The Cosine Similarity was used with a high threshold set to 75% to ensure that only similar patterns are compared with each other and to limit the similarity comparison only to patterns that are truly related to each other. So, if two patterns were at least 75% similar to each other the longer pattern was kept while the other one was ignored. The cosine similarity calculation is shown in equation 3, where A and B are the compared patterns.

$$COS(\Theta_{A,B}) = \frac{\overrightarrow{A} \cdot \overrightarrow{B}}{\|\overrightarrow{A}\|\|\overrightarrow{B}\|} \quad (3)$$

D. Discussion

With our event detection framework from data streams, we can detect events occurring every day if any. The goal is not to identify all the daily occurring frequent patterns from every window-batch, but instead, to detect and find a frequent pattern that is eligible for being an event. Looking for events or topics that emerge every day is challenging, we basically depend on only a single window-batch to define its support value and to identify and detect events if any.

Evaluating the found topics was done manually, we looked for news headlines published in the World Wide Web the same day a topic (frequent pattern) was found and compared them. If the detected event or topic keywords matches a news headline keywords and is found the same day the news was published, then it is said a true event, and a false or spurious event otherwise.

From the GE stream all the found frequent patterns till window 20 (12th of May 2015) were related to news headlines and were found the same day the headline was published on the World Wide Web. Except for window 18 which is one day after the elections date where the lingering effect of the elections results was still showing. The detected events still seem to be associated with the aftermath of the elections and topics that people were discussing during the elections like NHS, disability allowance etc. On the other hand, frequent patterns found few days after the elections date (more precisely from Monday 13th of May onwards)

were less related to events but instead they were arguing and disseminating various things. The support calculated values in this period dramatically fall as a result of the lower number of posted tweets. Consequently keywords co-occurrence in these windows is small as well, refer to table II to see the number of posted tweets, keywords, and support calculated values. Consider for example a support value equal to 23, this means the terms of any tweet getting at least 23 retweets is satisfying the support value. A too low support value will lead to finding frequent patterns that are vague and does not represent an event. Therefore a more strict support value is needed in less active days (window-batches with fewer text-posts) to avoid detecting spurious events. So, depending on the number of text-posts found in a window-batch, the support value is calculated as follows in equations 4 and 5. Equation 5 shows the small sized window equation, where we double the median value in order to have an even stricter support value. However a window is considered a small sized window if the number of posted tweets is less than one-third of the average number of total tweets in the stream. For the GE stream a window with less than 10k tweets is considered small size window. So windows 20 till 29 are considered small sized windows and require the small window support calculation method.

Big window-batch:

$$Spp = avg(S_T) \times median(S_T) \tag{4}$$

Small window-batch:

$$Spp = avg(S_T) \times (2 \times median(S_T)) \tag{5}$$

After applying the low support value restriction, all the calculated support values still reflect their window-batch size, which in turn will help in retrieving only targeted and potential terms. The calculated support value for window 18, which is the biggest window with (388966) tweets containing (28614) distinct terms, is equal to 544, whereas the support value for window 26, which is the smallest window with only 1609 tweets containing (1449) distinct terms, is 41. For window 10, which is a medium sized window with 27023 tweets containing (7035) distinct terms, the support is 151.

Moreover, all the detected events after applying the strict support definition method from the GE stream matched news headlines regardless of the window-batch size, refer to table IV where each window is shown with its detected frequent patterns along with the associated news headline.

When event detection is applied in realtime or near realtime the decision of an upcoming window-batch size is not determined as the size of the stream is unknown in advance. We regard the window size allocation and decision as a 2-class (big or small) classification problem. As a result a logistic regression (LR) model [31], [32] was built to decide whether a window-batch is big or small size window. Logistic regression is a machine learning algorithm for binary classification. It is a widely used statistical method for analysing a dataset in which there are one or more independent variables that determine a binary outcome. The LR model is shown in equation 6, where β_0 is the intercept and $\beta_1, \beta_2, .., \beta_n$ are coefficients associated with the independent variables $x_1, x_2, .., x_n$. In this work, we have the number of streamed tweets in a window as the independent variable which determines a binary outcome (1 for big size window and 0 for small size window). Our logistic regression model was built using the GE stream as a training set, and have learnt that the intercept $\beta_0 = -72.550$ and the coefficient $\beta_1 = 0.0538$.

$$\frac{1}{1 + e^{-(\beta_0 + \sum_{i=1}^{n} \beta_i x_i)}} \tag{6}$$

For the Greece Crisis stream, we explored 17 windows, due to space restrictions it has not been possible to show the detected events. However, all the detected events using our event detection framework along with the LR model matched news headlines. Furthermore, the nature of the Greece Crisis event is different from the General Elections event. This was clear from the high number of detected events for some windows, where events were unsustainable, successive and fast.

V. Conclusion

This paper has shown the applicability of our event identification framework on streams of data. Specifically it showed how using the FP-Growth algorithm along with our dynamic support definition method detects the occurring events or topics from the Twitter stream. The identified events depend on what was frequently mentioned and extensively discussed in the investigated stream. Our support definition depends on the window-batch size and the occurrence of keywords. Experiments were conducted on Twitter stream for the UK General Elections 2015 and the Greece Crisis 2015 to detect the occurring events and to evaluate the support definition method. The detected events were evaluated against what was published on the same day in news wires. If the detected event was found the same day a news headline is published, then it is said to be a true event and a false event otherwise. Results showed that bigger windows with high number of tweets detect more accurately events than smaller ones. As a result, a strict support version was proposed for small windows to avoid detecting insignificant events.

In the near future, we further want to evaluate our event detection framework against the SFPM framework in [15] along with developing a method to rank the detected events if more than one is found on a single day.

Table IV
WINDOW DETECTED EVENTS AND NEWS HEADLINES

Window no.	Frequent pattern	News headline
1-4	No found frequent patterns	
5	Rattled, badly, obviously, polls, sir, snp, john, says, tories, majors, speech	SNP says Sir John Major's speech 'very foolish'. This attitude doesnt respect democracy and is completely wrong. The Tories are obviously badly rattled by the polls. http://www.theguardian.com/politics/live/2015/apr/21/election-2015-live-labour-john-major-blackmail-snp-nicola-sturgeon-ed-miliband?page=with%3Ablock-55361517e4b046f7a16b5992
6	No found frequent patterns	
7	Underestimates, aware, ifs, says, cuts, Miliband, lab, Scottish, labours, gets, excited	Nicola Sturgeon has also been responding to the IFS report on Twitter. Before Scottish Lab gets excited about IFS, they should be aware that Ed Miliband says it underestimates Labour's cuts. http://www.theguardian.com/politics/live/2015/apr/23/election-2015-live-ifs-verdict-labour-conservatives-liberal-democrats-snp-tax-and-spending-plans?page=with%3Ablock-553900cbe4b0401b98b64bdd
8	No found frequent patterns	
9	Breaking, dems, news, lib, snp, labour, poll, Scotland, new, tns	BREAKING NEWS: New TNS Scotland poll (SNP 57, Labour 1, Lib Dem 1). http://www.telegraph.co.uk/news/general-election-2015/11566036/Poll-Labour-reduced-to-one-seat-in-Scotland.html
10	No found frequent patterns	
11	Forget, votes, win,win, Scotland, seats, working, polls, lets, stronger, elections Enough, labour, people, says, jim, murphy, need, change, east, end, years, Glasgow, correct Forms, voting, candidates, east, labours, postal, major, hull, missing, including, cockup	Poll: SNP on course for clean sweep in Scotland. http://www.telegraph.co.uk/news/politics/SNP/11570881/polls.html Again we stand here tonight proud to be Labour. With Labour change is coming to the east end of Glasgow, East end of Manchester, Liverpool,Cardiff, Edinburgh. http://www.scottishlabour.org.uk/blog/entry/no-more-waiting.-no-more-wasted-lives.-jim-murphy#sthash.SvcIUJfo.dpuf A major election cock-up in Hull. The council has sent out a large batch of postal ballot papers in the Hull East constituency which have omitted the names of the Labour candidate Karl Turner and the Green candidate Sarah Walpole. http://news.channel4.com/election2015/04/29/
12	Running, stop, vote, snp, snp, time, sun, sun, Scottish, country	The Sun and its sister paper, the Scottish Sun, have endorsed different parties in the general election. http://www.bbc.co.uk/news/election-2015-scotland-32523804
13	Sco, Miliband, snp, snp, tories, work, let, mps, need, going, definitely, lots, protect Shortest, history, suicide, note, Scottish, political, says, let, Miliband, tories, work	Ed Miliband would rather lose than do SNP deal. http://www.scotsman.com/news/politics/ed-miliband-would-rather-lose-than-do-snp-deal-1-3759986 Miliband said. If the price of having a Labour government was coalition or a deal with the Scottish National Party, its not going to happen.. http://www.telegraph.co.uk/news/general-election-2015/politics-blog/11576757/Ed-Milibands-SNP-lie-could-damage-him-more-than-Nick-Cleggs-tuition-fee-promise.html
14	Try, absolute, murphy, Glasgow, jim, eddie, izzard, heard, protestors, chaos, streets Condemns, violent, aggressive, event, protestors, izzard, eddie, campaign, labour, glasgow	BBC Scotlands James Cook described the scene of the pair trying to deal with the protesters as absolute chaos. http://www.huffingtonpost.co.uk/2015/05/04/eddie-izzard-jim-murphy-glasgow-rally_n_7203100.html Scottish Labour leader Jim Murphy and comedian Eddie Izzard were heckled by opponents during general election campaigning in Glasgow. http://www.bbc.co.uk/news/election-2015-scotland-32581803
15	No found frequent patterns	
16	Pinned, throttled, wall, galloways, supporters, told, George, Bradford, fucking, jew	A burly Asian man in a black suit and sunglasses rushes up and grabs me round the neck, pinning me to a low perimeter wall. Get out, you fucking Jew, he shouts. http://www.politico.eu/article/galloway-bradford-elections-uk-ge2015/
17	No found frequent patterns	
18	Teacher, time, poor, sick, old, disabled, unemployed, immigrant, student, unless, alive, nurse, doctor, Disability, suffer, family, kind, congrats, lose, job, get, tory, voters, hope, sick, better, Vulnerable, mourning, environment, rip, today, day, nhs, rights, human, national, welfare, state	
19	Fee, licence, rid, equalities, secretary, secretary, wants, minister, voted, bbc, against, culture, justice, hanging	BBC licence fee in doubt as John Whittingdale is named culture secretary. https://www.theguardian.com/media/2015/may/11/john-whittingdale-culture-secretary-bbc-charter-renewal
20	fee, equalities, secretary, secretary, voted, wants, minister, bbc, against	The BBC will be forced to slash its drama output if the licence fee is cut, according to the man who commissioned hits like Poldark, Sherlock, Call the Midwife and Wolf Hall. http://www.bbc.co.uk/news/entertainment-arts-32702746
21	Enquiries, reports, make, kent, following, seat, police, south, fraud, electoral, thanet	Kent Police investigate allegations of election fraud after Ukip,leader Nigel Farage's general election defeat in South Thanet. http://www.kentonline.co.uk/thanet/news/police-probe-election-fraud-allegations-36864/
22-29	No found frequent patterns	

REFERENCES

[1] A. Perrin., "Social networking usage: 2005-2015," Pew Research Center, Report 1296, October 2015. [Online]. Available: http://www.pewinternet.org/2015/10/08/2015/Social-Networking-Usage-2005-2015/

[2] S. Chakravarthy and Q. Jiang, *Stream data processing: A quality of service perspective: modeling, scheduling, load shedding, and complex event processing*. Springer Science & Business Media, 2009, vol. 36.

[3] E. Della Valle, S. Ceri, D. F. Barbieri, D. Braga, and A. Campi, *A first step towards stream reasoning*. Springer, 2009.

[4] A. Margara, J. Urbani, F. van Harmelen, and H. Bal, "Streaming the web: Reasoning over dynamic data," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 25, pp. 24–44, 2014.

[5] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom, "Models and issues in data stream systems," in *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM, 2002, pp. 1–16.

[6] D. C. Luckham, *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2001.

[7] L. M. Aiello, G. Petkos, C. Martin, D. Corney, S. Papadopoulos, R. Skraba, A. Göker, I. Kompatsiaris, and A. Jaimes, "Sensing trending topics in twitter," *IEEE Transactions on Multimedia*, vol. 15, no. 6, pp. 1268–1282, 2013.

[8] S. Phuvipadawat and T. Murata, "Breaking news detection and tracking in twitter," in *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2010 IEEE/WIC/ACM International Conference on*, vol. 3. IEEE, 2010, pp. 120–123.

[9] S. Petrović, M. Osborne, and V. Lavrenko, "Streaming first story detection with application to twitter," in *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 2010, pp. 181–189.

[10] X. Zhou and L. Chen, "Event detection over twitter social media streams," *The VLDB journal*, vol. 23, no. 3, pp. 381–400, 2014.

[11] M. Cataldi, L. Di Caro, and C. Schifanella, "Emerging topic detection on twitter based on temporal and social terms evaluation," in *Proceedings of the Tenth International Workshop on Multimedia Data Mining*. ACM, 2010, p. 4.

[12] J. Weng and B.-S. Lee, "Event detection in twitter." *ICWSM*, vol. 11, pp. 401–408, 2011.

[13] J. Han, H. Cheng, D. Xin, and X. Yan, "Frequent pattern mining: current status and future directions," *Data Mining and Knowledge Discovery*, vol. 15, no. 1, pp. 55–86, 2007.

[14] J. Guo, P. Zhang, L. Guo *et al.*, "Mining hot topics from twitter streams," *Procedia Computer Science*, vol. 9, pp. 2008–2011, 2012.

[15] G. Petkos, S. Papadopoulos, L. Aiello, R. Skraba, and Y. Kompatsiaris, "A soft frequent pattern mining approach for textual topic detection," in *Proceedings of the 4th International Conference on Web Intelligence, Mining and Semantics (WIMS14)*. ACM, 2014, p. 25.

[16] J. Huang, M. Peng, and H. Wang, "Topic detection from large scale of microblog stream with high utility pattern clustering," in *Proceedings of the 8th Workshop on Ph. D. Workshop in Information and Knowledge Management*. ACM, 2015, pp. 3–10.

[17] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[18] C. W. Fox and S. J. Roberts, "A tutorial on variational bayesian inference," *Artificial intelligence review*, vol. 38, no. 2, pp. 85–95, 2012.

[19] X. Han and T. Stibor, "Efficient collapsed gibbs sampling for latent dirichlet allocation [j]," *Journal of Machine Learning Research*, vol. 13, pp. 63–78, 2010.

[20] D. Ramage, D. Hall, R. Nallapati, and C. D. Manning, "Labeled lda: A supervised topic model for credit attribution in multi-labeled corpora," in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*. Association for Computational Linguistics, 2009, pp. 248–256.

[21] R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases," in *Acm sigmod record*, vol. 22, no. 2. ACM, 1993, pp. 207–216.

[22] J. Han, J. Pei, Y. Yin, and R. Mao, "Mining frequent patterns without candidate generation: A frequent-pattern tree approach," *Data mining and knowledge discovery*, vol. 8, no. 1, pp. 53–87, 2004.

[23] H. M. Nabil, A. S. Eldin, and M. A. E.-F. Belal, "Mining frequent itemsets from online data streams: Comparative study," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 4, no. 7, 2013.

[24] P. S. Tsai, "Mining frequent itemsets in data streams using the weighted sliding window model," *Expert Systems with Applications*, vol. 36, no. 9, pp. 11 617 – 11 625, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417409002723

[25] C. Giannella, J. Han, J. Pei, X. Yan, and P. S. Yu, "Mining frequent patterns in data streams at multiple time granularities," *Next generation data mining*, vol. 212, pp. 191–212, 2003.

[26] S. Gaglio, G. L. Re, and M. Morana, "A framework for real-time twitter data analysis," *Computer Communications*, vol. 73, pp. 236–242, 2016.

[27] A. M. Mood, *Introduction to the Theory of Statistics*. NY, USA: McGraw-hill, 1950.

[28] P. Fournier-Viger, A. Gomariz, T. Gueniche, A. Soltani, C.-W. Wu, V. S. Tseng *et al.*, "Spmf: a java open-source pattern mining library." *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3389–3393, 2014.

[29] A. Zubiaga, M. Liakata, R. Procter, K. Bontcheva, and P. Tolmie, "Towards detecting rumours in social media," *arXiv preprint arXiv:1504.04712*, 2015.

[30] A. Gupta, H. Lamba, P. Kumaraguru, and A. Joshi, "Faking sandy: characterizing and identifying fake images on twitter during hurricane sandy," in *Proceedings of the 22nd international conference on World Wide Web*. ACM, 2013, pp. 729–736.

[31] S. H. Walker and D. B. Duncan, "Estimation of the probability of an event as a function of several independent variables," *Biometrika*, vol. 54, no. 1-2, pp. 167–179, 1967.

[32] D. W. Hosmer Jr and S. Lemeshow, *Applied logistic regression*. John Wiley & Sons, 2004.