Erschienen in: Information Systems ; 62 (2016). - S. 207-219
https://dx.doi.org/10.1016/j.is.2016.01.003

An evaluation of the run-time and task-based performance of event detection techniques for Twitter

Andreas Weiler*, Michael Grossniklaus, Marc H. Scholl

Department of Computer and Information Science, University of Konstanz, Germany

Keywords:
Event detection
Performance evaluation
Twitter social media data stream

A B S T R A C T

Twitter's increasing popularity as a source of up to date news and information about current events has spawned a body of research on event detection techniques for social media data streams. Although all proposed approaches provide some evidence as to the quality of the detected events, none relate this task based performance to their run time performance in terms of processing speed, data throughput, or memory usage. In parti cular, neither a quantitative nor a comparative evaluation of these aspects has been performed to date. In this article, we study the run time and task based performance of several state of the art event detection techniques for Twitter. In order to reproducibly compare run time performance, our approach is based on a general purpose data stream management system, whereas task based performance is automatically assessed based on a series of novel measures.

1. Introduction

With 271 million monthly active users[1] that produce over 500 million tweets per day,[2] Twitter is the most popular and fastest growing microblogging service. Microblogging is a form of social media that enables users to broadcast short messages, links, and audiovisual con tent. In the case of Twitter, these so called *tweets* can contain 140 characters[3] and are posted to a network of *followers* as well as to a user's public timeline. The brevity of tweets makes them an ideal mobile communication medium and Twitter is therefore increasingly used as an information source for current events as they unfold. For example, Twitter data has been used to detect earthquakes [41], to track epidemics [16], or to monitor elections [51].

In this context, an *event* is defined as a real world occurrence that takes place in a certain geographical location and over a certain time period [5]. For traditional media such as newspaper archives and news websites, the problem of event detection has been addressed by research from the area of Topic Detection and Tracking (*TDT*). However, topic detection in Twitter data streams introduces new challenges. First, Twitter “documents” are much shorter than traditional news articles and therefore harder to classify. Second, tweets are not redacted and thus contain a substantial amount of spam, typos, slang, etc. Finally, the rate at which tweets are produced is very bursty and continually increases as more people adopt Twitter every day.

Several techniques for event detection in Twitter have been proposed. However, most of these approaches suffer from two major shortcomings. First, they tend to focus exclusively on the information extraction aspect and

* Corresponding author.
E-mail addresses: Andreas.Weiler@uni-konstanz.de (A. Weiler), Michael.Grossniklaus@uni-konstanz.de (M. Grossniklaus), Marc.Scholl@uni-konstanz.de (M.H. Scholl).

[1] http://www.statista.com/study/9920/twitter-statista-dossier/ (August 18, 2015).

[2] http://www.sec.gov/Archives/edgar/data/1418091/000119312513390321/d564001ds1.htm (August 18, 2015).

[3] In August 2015, Twitter lifted this restriction for direct messages, but not for general tweets: http://blog.twitter.com/2015/removing-the-140-character-limit-from-direct-messages (August 18, 2015).

often ignore the streaming nature of the input. As a consequence, they make unrealistic assumptions, which limit their practical value. Examples of such assumptions include buffering entire months of Twitter data before processing it or fixing a complex set of parameters at design time using sample data. Second, very few authors have evaluated their technique quantitatively or comparatively. While most provide some qualitative evidence demonstrating their task based performance, very few consider run time performance. Therefore, little or no research to date has measured the computing cost of the same result quality for different approaches. We argue that understanding this trade off is particularly important in a streaming setting, where processing needs to happen in real time.

In this article, we present a method to evaluate the run time and the task based performance of current and future event detection techniques. In order to measure comparable run time performance numbers, we propose to "standardize" event detection techniques by implementing them based on a single data stream management system. Additionally, we developed several scalable measures to assess the task based performance of event detection techniques automatically, i.e., without painstakingly crafting a gold standard manually. The specific contributions of this article are as follows.

1. Streaming implementations of state of the art event detection techniques for Twitter that are consistent with respect to each other.
2. Detailed study of the task based and run time performance of well known event detection techniques.
3. Platform based approach that will enable further systematic performance studies for novel event detection techniques in the future.

This article is an extended presentation of Weiler et al. [49]. In comparison to the conference version, it features two additional contributions. First, this article provides a much more detailed survey of the state of the art in event detection techniques for Twitter data streams. Since we analyze several recent approaches that have not yet been discussed in other surveys, this article closes a gap to these surveys with respect to open domain event detection techniques. Second, this article studies the run time and task based performance of event detection techniques more in depth by applying extra measures. In terms of run time performance, we additionally examine the run time performance on a second hardware setting and also the memory requirements of each technique, whereas in terms of task based performance, we analyze how many repeated as well as common events are detected by the techniques.

The remainder of this article is structured as follows. Section 2 presents our survey of the state of the art in open domain event detection for Twitter data streams. In Section 3, we give a brief overview of Niagarino, the data stream management system that we used as an implementation platform. Section 4 describes the selected event detection techniques and their streaming implementations using Niagarino. Section 5 discusses the results of the evaluation that we performed in order to study the selected task based and run time performance of these event detection techniques. Finally, concluding remarks are given in Section 6.

2. Background

In recent years, a lot of research has been conducted in the area of event detection and tracking techniques for Twitter. Consequently, a number of surveys exist that document the current state of the art. For example, Survey 1 by Nurwidyantoro and Winarko [36] summarizes eleven techniques to detect disaster, traffic, outbreak, and news events. Survey 2 by Madani et al. [28] presents 13 techniques that each address one of the four challenges of health epidemics identification, natural events detection, trending topics detection, and sentiment analysis. A more general survey with a wide variety of research topics related to sense making in social media data is Survey 3 by Bontcheva and Rout [12]. The work defines five key research questions user, network, and behavior modeling as well as intelligent and semantic based information access. The part about semantic based information access also includes an overview about event detection techniques in social media data streams. They classify event detection methods into three categories: clustering based, model based, and those based on signal processing. Furthermore, an overview about techniques for "sub events" detection is presented. Finally, the most extensive survey to date is Survey 4 by Farzindar and Khreich [17] with a listing of 16 different techniques categorized by their detection methods, tasks, event types, application domains, and evaluation metrics.

Since the work presented in this paper targets approaches that support the detection of general (unknown) events [5], we will focus our survey on open domain event detection techniques that share this goal. To the best of our knowledge, Table 1 lists all existing approaches that fall into this category in ascending order of their year of publication. The table also summarizes what technique approaches to use to detect events. Finally, the last column indicates in which of the above mentioned surveys each approach is included. The four surveys are referred to by using the number assigned to them in the previous paragraph.

As shown in Table 1, many approaches use similar techniques, but with individual modifications or extensions. For example, several approaches are based on statistical models that are defined to detect bursty behavior of single terms or pairs of terms. Most of these approaches analyze the Document Frequency (DF) or the Inverse Document Frequency (IDF) [43] of terms over time. While this technique is very common, the precise definition of what is considered to be a "burst" varies significantly among approaches. Another degree of variation is the level of sophistication present in existing approaches, which ranges from wavelet analysis to simple threshold based decisions. Some approaches additionally include spatial information in the analysis and present event detection techniques that are specific to a local area. Finally, several approaches can be summarized as using term clustering

Table 1
Summarization of open domain event detection techniques for Twitter.

Reference	Technique	Survey(s)
Cheong and Lee [14]	Statistical model	2
Sankaranarayanan et al. [42]	Online clustering	4
Benhardus [10]	Statistical model	2
Cataldi et al. [13]	Temporal and social model	2
Lee and Sumiya [23]	Statistical model	4
Mathioudakis and Koudas [32]	Statistical model	2
Naaman et al. [34]	Statistical model	2
Petrović et al. [39]	Locality-sensitive hashing	1, 3, 4
Becker et al. [9]	Online clustering, SVM	3, 4
Lee et al. [22]	Spatiotemporal model	–
Long et al. [27]	Hierarchical divisive clustering	4
Marcus et al. [30]	Statistical model	3
Weng and Lee [51]	Discrete wavelet analysis	1, 4
Aggarwal and Subbian [3]	Clustering	–
Alvanaki et al. [6]	Statistical model	–
Cordeiro [15]	Continuous wavelet analysis	4
Ishikawa et al. [21]	Clustering and burst detection	1
Li et al. [24]	Statistical model	1
Nishida et al. [35]	Classification model	–
Osborne et al. [37]	Temporal model	1
Ritter et al. [40]	Latent variable model	–
Zimmermann et al. [54]	Clustering	–
Aiello et al. [4]	Statistical model, clustering	–
Bahir and Peled [8]	Filtering	–
Martin et al. [31]	Statistical model	–
Parikh and Karlapalem [38]	Hierarchical clustering	–
Walther and Kaisser [46]	Spatiotemporal model	–
Weiler et al. [50]	Spatiotemporal model	–
Guille and Favre [18]	Temporal and social model	–
Ifrim et al. [19]	Filtering, clustering	–
Weiler et al. [47]	Statistical model	–
Zhou and Chen [53]	Graphical model	–
Meladianos et al. [33]	Graph model	–
Thapen et al. [44]	Bio-surveillance algorithms	–

techniques as they focus on the similarity of tweets by computing different types of clusters over time.

In the following, we will only present approaches that have not yet been covered by an other survey. Moreover, the approaches of Weng and Lee [51], Cordeiro [15], and Weiler et al. [47] will be described in detail in Section 4 as they form the basis of our evaluation.

The *enBlogue* [6] approach uses a tumbling[4] window model over the time dimension to compute statistics about tags and pairs of tags. Based on these statistics, unusual shifts in correlations are identified, which are assumed to be caused by real world events most of the times. The intensity of these shifts is used to measure the degree of their unpredictability. This measure is then used to rank the tag pairs that express emergent events.

Ritter et al. [40] describe an approach for open domain event detection, which is based on latent variable models. Their technique first discovers event types by matching the data and then uses these types to classify and aggregate events. The aggregated events are then mapped to a calendar view that shows the top five events per day in terms of the event entity, event phrase, and event type. However, the authors do not discuss how to apply this approach directly to the streaming data.

A clustering technique that uses a novel similarity score is presented by Aggarwal and Subbian [3]. Instead of applying content only similarity measures, this technique uses a similarity score, which is based on the graph structure of Twitter. These clusters are monitored over time and the growth rate of clusters is used as an indicator for events. Finally, bursty clusters are marked as events. Zimmermann et al. [54] also propose a technique to detect "bursts". They apply a text stream clustering method that detects, tracks and updates global and local bursts of news in a two level topic hierarchy. However, they use the term "local" not as geographic property, but rather as a burst, which occurs in a previously detected global burst. Therefore, they are able to build a hierarchy between the relationships of global and local bursts.

Nishida et al. [35] introduce a model for topic classification that identifies changes of statistical properties within tweet streams on a word basis. The model switches between two probability estimates based on full and recent data for each word when detecting changes in word probability. This switching enables the model to achieve both accurate learning of stationary words and quick responses to bursty words. These bursty words can then be detected as events.

The "SocialSensor"[5] project that was conducted from 2011 until 2014 and was funded by the European Union led to a large number of publications in the research area of social media data processing and analysis. As one of the outcomes of this project, Aiello et al. [4] compare six topic detection methods that can also be used to detect events. Consequently, their comparison uses different Twitter data sets that are related to major events. The first technique uses document pivot topic detection with Locality Sensitive Hashing (*LSH*) [20] indexing, which is closely related to the technique proposed by Petrović et al. [39]. The remaining techniques are all graph based feature pivot topic detection that use the *SCAN* algorithm, the standard Latent Dirichlet Allocation (*LDA*) [11] technique, (soft) frequent pattern mining, as well as *BNgram*. As a conclusion, they find that standard natural language processing techniques perform well on very focused topics, but novel techniques designed to mine the temporal distribution of concepts are needed. Hereby, *BNgram* that is based on n grams co occurrence and $df\ idf_t$ topic ranking, consistently achieves the best performance across all settings. Martin et al. [31] describe another approach that was developed in the context of the SocialSensor project. They

[4] In their paper, the authors describe it as a *sliding* window, but only specify the size of the window and not the range of the slide. Personal communication with one of the authors confirmed that indeed a tumbling window is used.

[5] http://www.socialsensor.eu (August 18, 2015).

introduce a time dependent variant of the standard *tf idf* technique, which they use to detect events by grouping bursty phases together that often occur in the same set of tweets.

Bahir and Peled [8] discuss techniques to detect events through a set of adequate spatial, temporal, and textual filters. They show that major events may have detectable "abnormal" impact on geo social network activities, which allow them to be detected and tracked in real time. *ET* [38] is an efficient and scalable system to detect events from tweets. The system consists of three key components: an extraction scheme for keywords that represent events, an efficient storage mechanism to store their appearance patterns, and a hierarchical clustering technique based on the common co occurring features of keywords. Events are then determined through the hierarchical clustering pro cess. A novel algorithm for geo spatial event detection is described by Walther and Kaisser [46]. In a first step, all tweets in the stream within a given geographic region are monitored and places that show a high amount of activity are identified. In a second step, the resulting spatio temporal clusters of tweets are analyzed with a machine learning component in order to detect an event. This approach is closely related to our own *LLH* technique [50] that uses a combined log likelihood ratio approach for the geographic and time dimension of Twitter data in pre defined areas in order to detect events. Another approach that is closely related to *LLH* is presented by Lee et al. [22]. They propose an event detection technique for local areas, which clusters tweets in real time using the incremental *DBSCAN* algorithm. Finally, the location of each cluster is identified by analyzing the users' timezones.

Guille and Favre [18] propose Mention Anomaly Based Event Detection (*MABED*). The novel aspect of *MABED* is to leverage the frequency of dynamic links (i.e., mentions), which are contained in tweets, in order to detect impor tant events and to estimate the magnitude of their impact over the crowd. Ifrim et al. [19] present a technique that uses aggressive filtering of tweets based on their length and structure. This approach is combined with hierarchical clustering of tweets and ranking of the resulting clusters. The highest ranked clusters are then detected as events. Zhou and Chen [53] propose a graphical model called Location Time constrained Topic (*LTT*), which captures the content, time, and location of tweets. Using *LLT*, a tweet can be represented as a probability distribution over a set of topics by inference. The similarity between two mes sages is measured as the distance between their distribu tions. Finally, events are detected by performing efficient similarity joins.

The most recent research works that we are aware of are presented by Meladianos et al. [33] and Thapen et al. [44]. The first one deals with the topic of sub event detection. They use a technique that models a sequence of successive tweets in a short time interval as a weighted graph of words. Based on this graph sub events are iden tified using the concept of graph degeneracy and included in an event. The second technique adapts existing bio surveillance algorithms in order to detect localized spikes in Twitter activity that correspond to real events with a high level of confidence.

3. Niagarino overview

In order to realize streaming implementations of state of the art event detection techniques for Twitter, we use Niagarino,[6] a data stream management system that is developed and maintained by our research group. The main purpose of Niagarino is to serve as an easy to use and extensible research platform for streaming applica tions such as the one presented in this article. The con cepts embodied by Niagarino can be traced back to a series of pioneering data stream management systems, such as Aurora [2], Borealis [1], and STREAM/CQL [7]. In particular, Niagarino is an offshoot of NiagaraST [26], with which it shares the most common ground. In this section, we briefly summarize the parts of Niagarino that are relevant for this article.

In Niagarino, a query is represented as a directed acyclic graph $Q = (O, S)$, where O is the set of operators used in the query and S is the set of streams used to connect the operators. The Niagarino data model is based on relational tuples that follow the first normal form, i.e., have no nesting. Two types of tuples can be distinguished, data and metadata tuples. Data tuples are strongly typed and have a schema that defines the domains of all attri butes. All data tuples in a stream share the same schema, which corresponds to the output schema of the operator that generates the tuples and must comply with the input schema of the operator that consumes the tuples. In con trast, metadata tuples, so called messages, are untyped and typically self describing. Therefore, different messages can travel in the same stream. Messages are primarily used to transmit data and operator statistics in order to coor dinate the operators in a query. Each stream is bidirec tional consisting of a forward and a backward direction. While data tuples can only travel forward, messages can travel in both directions.

Based on its relational data model, Niagarino imple ments a series of operators. The selection (σ) and projec tion (π) operator work exactly the same as their counter parts in relational databases. Other tuple based operators include the derive (f) and the unnest (μ) operator. The derive operator applies a (user defined) function to a sin gle tuple and appends the result value to the tuple. The unnest operator splits a "nested" attribute value and emits a tuple for each new value. A typical use case for the unnest operator is to split a string and to produce a tuple for each term it contains. Apart from these general operators, Niagarino provides a number of stream specific operators that can be used to segment the unbounded stream for processing. Apart from the well known time and tuple based window operators (ω) that can be tum bling or sliding [25], Niagarino also implements data driven windows, so called frames [29]. Stream segments form the input for join ($\bowtie$) and aggregation (Σ) operators. As with derive operators, Niagarino also supports user defined aggregation functions. Niagarino operators can be partitioned into three groups. The operators described

[6] http://www.informatik.uni-konstanz.de/grossniklaus/software/niagarino/

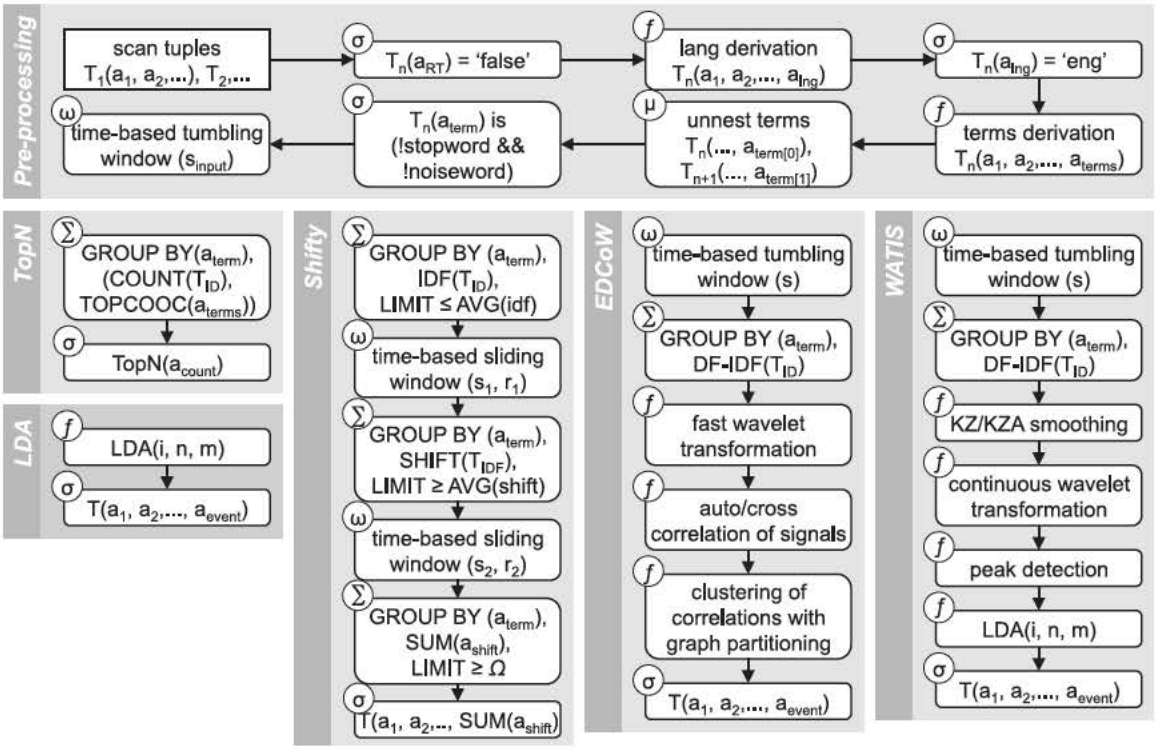

Fig. 1. Niagarino query plans of the five selected event detection techniques.

above are general operators, whereas source operators read input streams and sink operators output results. Each query can have multiple source and sink operators. This classification is similar to the notion of spouts and bolts used in Twitter's data stream management system Storm [45].

Niagarino is implemented in Java 8 and relies heavily on its new language features. In particular, anonymous functions (λ expressions) are used in several operators in order to support lightweight extensibility with user defined functionality. The current implementation runs every operator in its own thread. Operator threads are scheduled implicitly using fixed size input/output buffers that block upon underflow or overflow as well as explicitly through backward messages.

4. Event detection techniques

We focus on techniques with the specific task of first story detection, i.e., the detection of general (unknown) events, which is defined as a subtask of *TDT* [5]. In this section, we briefly describe the five state of the art techniques that we selected for our study in terms of their functionality and the parameters used. Fig. 1 illustrates these techniques by means of Niagarino query plans that use the operators described in the previous section. As can be seen in the figure, all of these techniques use the same preprocessing steps before the streaming tuples enter the actual event detection phase. The preprocessing selects all tweets that are non retweets and in English. Additionally, each tuple is enriched with the derived distinct terms of the tweet that are not contained in a standard English stop word list or can be considered noise (e.g., less than three characters, unknown characters, repetition of the same pattern, or terms without vowels).

The *TopN* algorithm assigns each individual term a single value based on the Inverse Document Frequency (IDF) over an entire time window. All values are then sorted and the top n terms are reported as events together with their top m most frequently co occurring terms, which are also obtained by using the IDF measure.

The *Latent Dirichlet Allocation* (*LDA*) is a hierarchical Bayesian model that explains the variation in a set of documents in terms of a set of n latent "topics", i.e., distributions over the vocabulary. Since LDA is normally used for topic modeling, we equate a topic to an event. For each time window, LDA extracts n events that are described by m terms. The parameter i defines the number of iterations performed in the modeling phase, where a higher value typically increases the quality of the detected events. To perform the LDA, we use *Mallet*,[7] an existing Java library.

Our own *Shifty* [47] technique calculates a measure that is based on the shifts of IDF values of single terms in pairs of successive sliding windows of a predefined size. First, the IDF value of each term in a single window (with size s_{input}) is continuously computed and compared to the average IDF value of all terms within that window. Terms with an IDF value above the average are filtered out. The

[7] http://mallet.cs.umass.edu (August 18, 2015).

next step builds a window with size s_1 that slides with range r_1 in order to calculate the shift from one window to the next. In this step, the shift value is again checked against the average shift of all terms and only terms with a shift above the average are retained. In the last step, a new sliding window with size s_2 that slides with range r_2 is created. The total shift value is computed as the sum of all shift values of the sub windows of this window. If this total shift value is greater than the pre defined threshold Ω, the term is detected as event and reported together with its top 4 co occurring terms.

The first step of the *Event Detection with Clustering of Wavelet based Signals* (*EDCoW*) [51] algorithm is to parti tion the stream into intervals of s seconds and to build DF IDF signals for each distinct term in the interval. These signals are further analyzed using discrete wavelet analysis that builds a second signal for the individual terms. Each data point of this second signal summarizes a sequence of values from the first signal with length Δ. The next step then filters out trivial terms by checking the corresponding signal auto correlations against a threshold γ. The remaining terms are then clustered to form events with a modularity based graph partitioning technique. Insignif icant events are filtered out using a threshold parameter ϵ. Since this approach detects events with a minimum of two terms, we introduced an additional enrichment step that adds the top co occurring terms to obtain events with at least five terms.

The *Wavelet Analysis Topic Inference Summarization* (*WATIS*) [15] algorithm also partitions the stream into intervals of s seconds and builds DF IDF signals for each distinct term. Due to the noisy nature of the Twitter data stream, signals are then processed by applying the adap tive Kolmogorov Zurbenko filter (KZA) [52], a low pass filter that smoothens the signal by calculating a moving average with i_{kz} iterations over n intervals. It then uses continuous wavelet transformation to construct a time/ frequency representation of the signal and two wavelet analyses, the tree map of the continuous wavelet extrema and the local maxima detection, to detect abrupt increase in the frequency of a term. To enrich events with more information, the previously mentioned LDA algorithm (with i_{lda} iterations) is used to finally report events that consist of five terms each.

5. Evaluation

The evaluation of event detection techniques is itself a challenging task. Determining an F_1 score in terms of precision and recall would require a ground truth (gold standard) to which the detected events can be compared. Due to lack of such a ground truth for the Twitter data stream, some existing approaches have been evaluated using a manually created ground truth or based on user studies, if at all. Since both of these methods are very time consuming and do not scale, we have experimented with a number of measures that can be applied automatically. In this section, we discuss the motivation behind these measures and present detailed results that were obtained by using them.

5.1. Measures

In order to evaluate different techniques automatically, we define the following main measures (some with sub measures) that are used for the individual ratings.

Run time performance: Run time performance is mea sured as the number of tweets per second that a technique is able to process (throughput).

Memory usage: This measure captures how much space is required by each technique. Since all techniques process a stream of data, i.e., processing never stops, we are interested to study how memory usage evolves over time and how it is bounded.

Precision (*search engine*): The first precision measure is defined as the percentage of events that can be verified with the use of a search engine (e.g., www.google.com). For each detected event, the search engine is queried using the five event terms and a specific date range. A rating between 1 and 10 (*GoogleN*) is computed by checking how many of the first ten result hits point to a news website. News websites are identified based on a whitelist of domain names containing sites such as CNN, CBS, Reuters, NYTimes, and the Guardian. Based on this measure, detected events can be rated with respect to their news worthiness on or at least one day after the detection date.

Precision (*DBPedia*): The second precision measure is calculated using the DBPedia[8] data set, which contains the abstracts (long versions) from all Wikipedia articles. In order to query the roughly four million English abstract, the native XML database BaseX[9] is used. For each detected event, the number of matching abstracts in DBPedia is computed using XQuery Full Text. We have defined three sub measures. *DBPedia5* is the precision using all five event terms, *DBPedia4S* only uses the top four event terms, and *DBPedia4A* queries DBPedia with all subsets of car dinality four. For the first two measures, an abstract is considered a match to an event if it contains *all* terms that were used in the query. For the third measure, an abstract matches if it contains all terms of *one* of the combinations.

Recall: In order to compute the recall, *Bloomberg*[10] was crawled as their archive maintains a list of the most important news articles for each day. Crawling individual days leads to an average of about 200 events per day. Each crawled news item is then tokenized and cleaned by the same processes as the tweets. As a consequence, the short description of each news item by a series of terms can be very similar to the one obtained from the tweets. In order to calculate the similarity between detected events and a news item, $\mathbf{eventSim}(e_1, e_2)$ is used, which is based on the Levenshtein distance.

$$\mathbf{levSim}(t_1, t_2) = 1.0 \quad \text{lev}(t_1, t_2)/\max(\{|t_1|, |t_2|\}) \tag{1}$$

$$\mathbf{termSim}(t_1, t_2) = \begin{cases} 0 & \text{levSim}(t_1, t_2) < minTermSim \\ 1 & \text{otherwise} \end{cases} \tag{2}$$

[8] http://dbpedia.org (August 18, 2015).
[9] http://basex.org (August 18, 2015).
[10] http://www.bloomberg.com/archive/news/ (August 18, 2015).

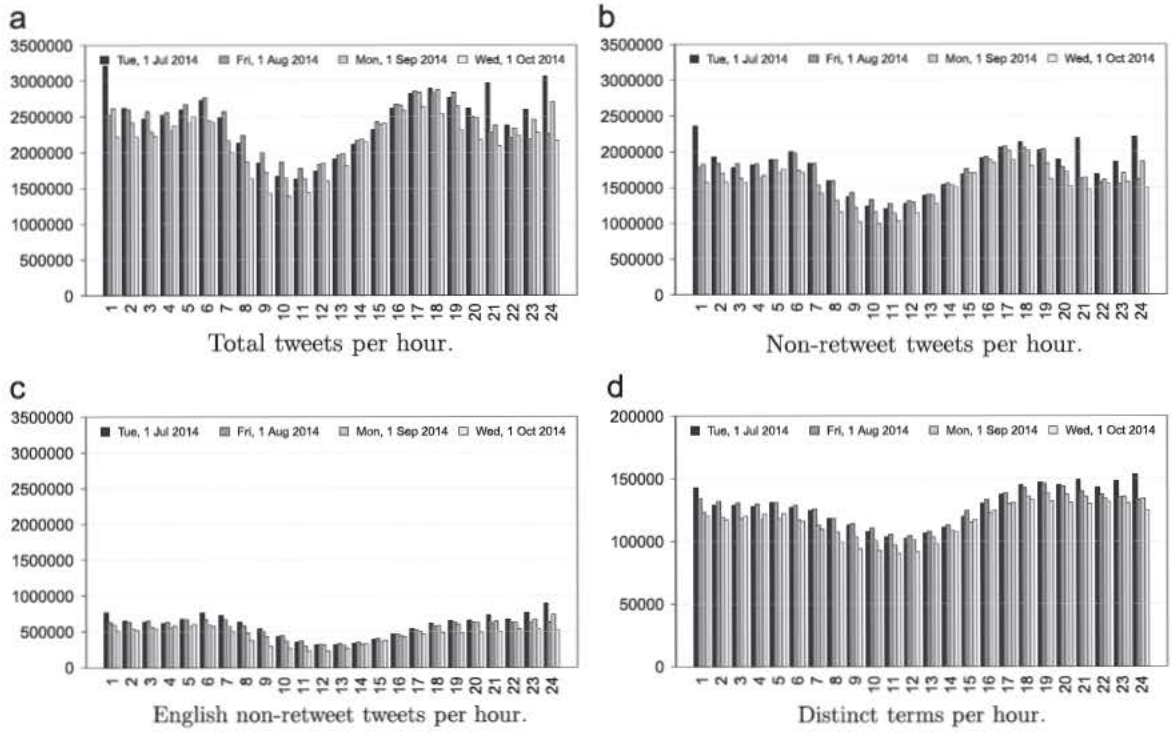

Fig. 2. Statistics of the Twitter data set.

$$\textbf{eventSim}(e_1, e_2) = \frac{1}{N_i} \sum_{0,j}^{N}{}_{0} \text{termSim}(e_1[t_i], e_2[t_j]) \qquad (3)$$

The motivation behind $\textbf{eventSim}(e_1, e_2)$ is to compensate for misspellings or alternate spellings of terms as well as for different term sets describing similar events. An event is represented as an alphabetically sorted list of terms $e = [t_0, \ldots, t_n]$. Each term $t_1 \in e_1$ is compared to each term $t_2 \in e_2$ using the $\textbf{levSim}(t_1, t_2)$, which is the Levenshtein distance normalized to the range [0…1]. If the similarity of a term of e_1 to a term of e_2 is above the threshold *minTermSim*, this combination is marked as hit and the algorithm continues with the next term of e_1. Finally, $\textbf{eventSim}(e_1, e_2)$ aggregates the number of hits and normalizes it with the number of terms.

In an effort to obtain a reasonable amount of hits, the parameters of this formula are set rather low. The parameter *minTermSim* is set to 0.7 and the overall limit for **eventSim** is set to 0.2. Two sub measures are defined for the recall. *Bloom1D* calculates the recall just for the given date, whereas *Bloom2D* also includes the following day.

Duplicate Event Detection Rate (*DEDR*): This measure is also based on the event similarity defined above in order to calculate the pairwise similarity of all events for one single technique and data set. Two sub measures have been defined. For *ADEDR* (almost duplicate event detection rate) the parameter *minTermSim* is set to 0.8 and the limit for *eventSim* is set to 0.5, whereas for *FDEDR* (full duplicate event detection rate) the *minTermSim* stays the same but the limit for *eventSim* is set to 0.9.

Repeated Event Detection Rate (*REDR*): This measure is derived from the *DEDR* measure and reflects the rate of repeatedly detected events per technique. In order to evaluate the results of the studied event detection techniques, we use a number of different data sets (cf. Section 5.2). Therefore, it is possible that a technique reports the same events for each of these data sets. Since the points in time at which our data sets were captured from the Twitter data stream are spaced one month apart, it is likely that these repeated events are artifacts of the technique. However, apart from measuring the percentage of repeating events for a series of single evaluations, this measure can also be used to evaluate how many of the "hit" events are detected repeatedly.

Common Event Detection Rate (*CEDR*): This measure is also derived from the *DEDR* measure and captures the percentage of events that are detected in common by pairs of techniques. With this measure, we can therefore study which techniques have a lot of commonly detected events. Furthermore, it is possible to derive which techniques detected the same "hit" events.

5.2. Data sets

The data sets used in the study presented in this article consist of 10% of the public live stream of Twitter for four days. Using the Twitter Streaming API[11] with the so called "Gardenhose" access level, which is a randomly sampled sub stream, we collected data for the first day of July, August, September, and October. Fig. 2 provides statistics of the initial data set as well as for the processing steps

[11] https://dev.twitter.com (August 18, 2015).

Table 2
Average number of detected events per techniques and dataset.

	Dataset				
	Jul1	**Aug1**	**Sep1**	**Oct1**	**AVG**
Technique					
Top15	360	360	360	360	360
LDA500	360	360	360	360	360
Shifty	327	316	354	402	350
EDCoW	353	375	396	409	383
WATIS	270	261	287	276	273

that are common to all techniques (cf. Fig. 1). Fig. 2a presents the total number of tweets for the chosen days grouped by the hour (given in $GMT+1$). As can be seen, the rate of tweets follows a regular daily pattern. On average, the incoming stream contains 2.3 million tweets/hour or 35,000 tweets/minute. Fig. 2b shows the hourly tweet volumes after filtering out retweets at an average of 1.6 million tweets/hour. After the next step, shown in Fig. 2c, the data sets are further reduced to an average of 500,000 tweets/hour by filtering out tweets that are not in English. Finally, Fig. 2d shows an average of 120,000 distinct terms/hour that have been derived from all English tweets.

5.3. Experimental setup

In order to be able to compare the results of the five chosen techniques in a fair way, they have to be aligned in terms of the rate and number of events detected. The rate can be controlled by setting the time window on which a technique is performed. Since we are interested in (near) real time event detection, a window of one hour was used. Note that *Shifty* is the only true streaming algorithm that reports results continuously, whereas all other techniques only produce results after each hour. The number of events that are detected can be controlled by setting the specific parameters of each technique. Given that our recall measure assumes an average of 200 events per day and compensating for events that are detected multiple times, we aim for about 350 events per day. The parameter settings used are described below, whereas the actual number of detected events per day and technique are shown in Table 2.

TopN: Per hour, the top $n=15$ events are reported together with $m=5$ co occurring terms to obtain a total of 360 events per day

LDA: LDA is set to perform $i=500$ iterations and to report 15 events, described by $m=5$ terms each, per hour, yielding again a total of 360 events per day.

Shifty: The IDF value is calculated over 1 min intervals. The size of the window used to compute the IDF shift is $s_1 = 2$ min. The size of the window that aggregates and filters the IDF shift is $s_2 = 4$ min. Both windows slide by range $r_1 = r_2 = 1$ min. By setting the threshold $\Omega = 0.35$, we obtain all terms with a minute by minute IDF value that increases more than 35% over four minutes.

EDCoW: The size of the initial intervals is set to $s=10$ s and the number of intervals that are combined by the wavelet analysis to $\Delta = 32$, yielding a total window size per value of 320 s. The other parameters are set to the same values as in the original paper ($\gamma = 1$ and $\epsilon = 0.2$). As the original paper fails to mention the wavelet type that was used, we experimented with several types. The results reported in this article are based on the *Discrete Meyer* wavelet, which showed the best performance.

WATIS: The length of initial intervals is set to $s=85$ s. For the KZ/KZA analysis, $n=5$ intervals and $i_{kz}=5$ iterations are used, yielding a total window size of 425 s. LDA is set to perform $i_{lda}=500$ iterations and report a description with five terms per detected event.

5.4. Results

In the following, we present the results of our evaluation of event detection techniques in terms of run time and task based performance. Rather than discussing all results that we have obtained, we focus on the most significant measures and outcomes. While we do not claim that our measures are absolute, it should be noted that these results support relative conclusions.

5.4.1. Run time performance

Run time performance was measured in two different settings. The first setting ($M1$) consisted of Oracle Java 1.8.0_25 (64 bit) on server grade hardware with 2 Intel Xeon E5345s processors at 2.33 GHz with 4 cores each and 24 GB of main memory. The second setting ($M2$) consisted of Oracle Java 1.8.0_40 (64 bit) on server grade hardware with 1 Intel Xeon E5 processors at 3.5 GHz with 6 cores and 64 GB of main memory. Regardless of the physical memory, the *Xmx24G* flag of the Java Virtual Machine (JVM) was used in both settings to limit the maximum memory to 24 GB. The corresponding results for all techniques in terms of throughput (tweets/second) are given in Fig. 3. The error bars denote the variance of run time performance across the four data sets, which is mostly very stable. Taking into account the average rate of 35,000 tweets/minute (583 tweets/second), we can derive that all techniques are able to process the 10% stream in real time in both settings. However, taking a 100% stream ($\sim$5830 tweets/second) into account, both *LDA500* and *WATIS* would be too slow to process the stream in real time in the $M1$ setting. For both of these event detection techniques, the number of LDA iterations could be reduced, i.e., trading off result quality for performance. Finally, we point out that our experimental setup is stacked against our own technique, *Shifty*. In contrast to the other approaches that can only process tweets at the end of each one hour window, *Shifty* processes tweets continuously and can therefore amortize its processing cost over the one hour window. Furthermore, we note that

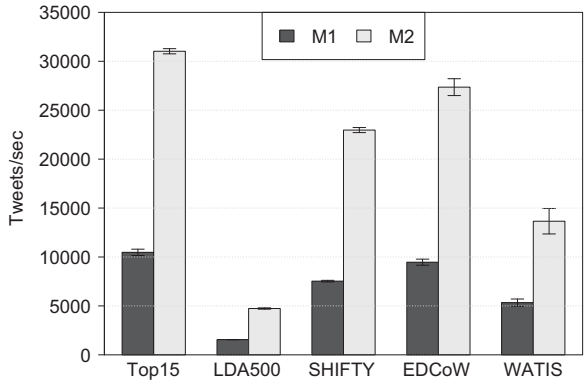

Fig. 3. Average run-time performance.

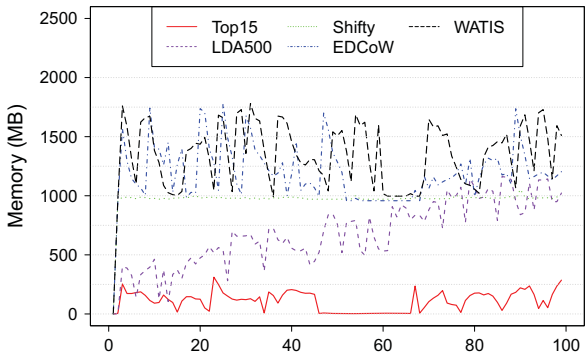

Fig. 4. Memory usage.

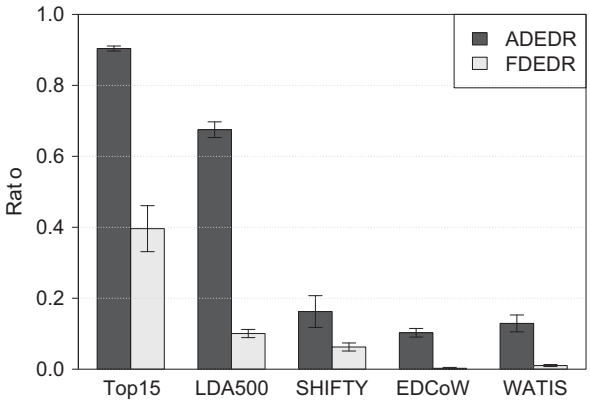

Fig. 5. Average duplicate rate for *ADEDR* and *FDEDR*.

the run time performance of all techniques scales by the same factor ($\sim 3\times$) when running them in setting *Mach2*. Therefore, we can reason that the performance difference between the techniques is always the same, no matter what hardware setting it is used.

5.4.2. Memory usage

Apart from the run time performance, the amount of memory required by each technique to process the data is another important factor. In order to compare memory usage fairly, we measured the memory consumption at a fixed number of measurement points. We first recorded

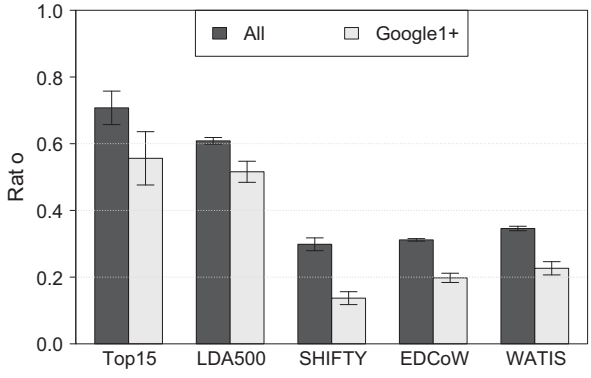

Fig. 6. Average ratio of repeated events for **all** and **Google1+** events.

the total run time needed by a technique to process all four datasets. Based on this total run time, we then derived the intervals at which memory usage has to be measured in order to obtain a total of 100 measurements. Before each measurement was taken, the garbage collector of the JVM was invoked to ensure better reproducibility of the results. All measurements reported in this article were recorded using the *M2* setting. Fig. 4 plots the amount of used memory in megabytes at each of the 100 measurement points. As expected, the *TopN* technique requires the least memory. The memory usage of *LDA* is similarly low at the beginning of the computation, but steadily increases over time. Since our implementation relies on a third party library to perform the Latent Dirichlet Allocation, we are not able to explain this increasing memory usage. Both *EDCoW* and *WATIS* require substantially more memory with a usage that fluctuates between 1 GB and 1.8 GB. In contrast, *Shifty* requires constant memory of about 1 GB. Again, this result is not unexpected as *Shifty* is the only truly streaming algorithm that we study in this article.

5.4.3. Task based performance

The first measure of task based performance that we will examine is the duplicate event detection rate. Results obtained using both the *ADEDR* and *FDEDR* sub measures are given in Fig. 5. In comparison to the other three techniques, both *Top15* and *LDA500* detect a large number of duplicates. This result is explained by the fact that these techniques identify events based on the absolute frequency of terms, i.e., without considering changes in the relative frequency. The *ADEDR* of the remaining three techniques is relatively low in the range of 15 18%. *Shifty*'s *FDEDR* stayed consistently below 10% in all our experiments, whereas *EDCoW* and *WATIS* do hardly detect any duplicates at all. Finally, the results also show that there is little deviation in the detected number of duplicates over the four days in our data set.

Apart from the duplicate event detection rate, we defined two more measures, which are based on the event similarity. First, we evaluate the ratio of repeated events for each of the techniques. Fig. 6 shows the average ratio of repeated events per technique, with error bars indicating the standard deviation over all four data sets. This measure is calculated based on the *ADEDR* of all pairwise combinations of event sets that a given technique

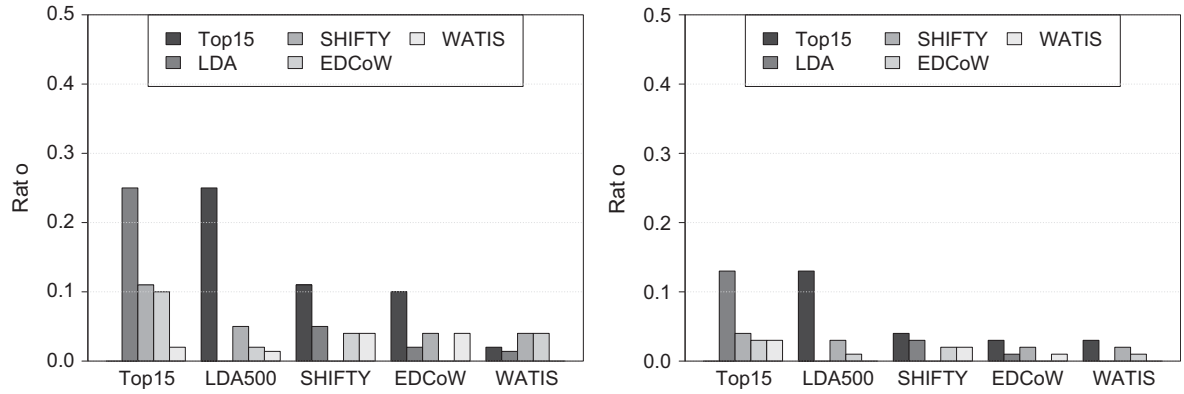

Fig. 7. Average ratio of common events for **all** events on the left side and **Google1+** events on the right side.

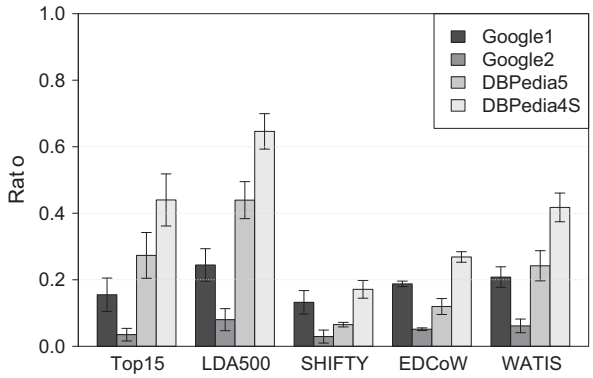

Fig. 8. Average precision.

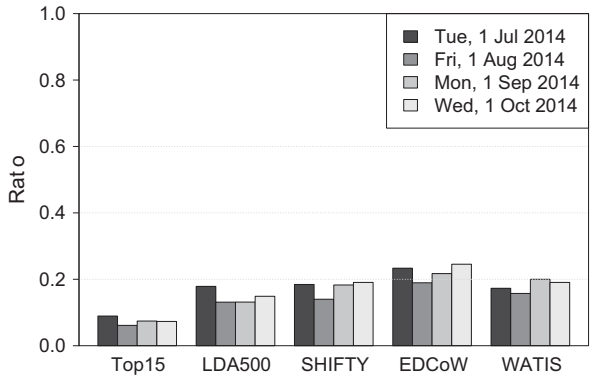

Fig. 9. Recall using *Bloom1D*.

reported for the datasets, i.e., July August, July September, July October, and August September. We can derive that the event detection techniques produce repeated events with a ratio of less than 20% for all events as well as for Google1+ events. The Google1+ events is the subset of reported results that are identified as results in the pre cision study. In contrast to these event detection techni ques, *Top15* and *LDA500* report a lot of similar events for the different datasets: about 60% for all events and about 55% for Google1+ events.

Second, we evaluate the ratio of common events between the different techniques. For this measure, we set the *minTermSim* to 1.0 and the limit for *eventSim* to 0.8. Fig. 7 shows the average ratio of common events between the techniques for all and for Google1+ events. We can observe that the *Top15* and *LDA500* techniques share a lot of events for both types of events. In contrast to the other techniques, they share about 25% of all events and 15% of Google1+ events, which is more than twice as much as the other techniques. However, the other three techniques do not share a high number of events among themselves either. The reason for this might be that the algorithms used by these techniques are simply too different. Never theless, it is possible to deduce based on these results that the number of events that these three techniques have in common with *Top15* or *LDA500* is very low, both for all and for Google1+ events.

Apart from the duplicate, repeated, and common event detection rates, we have also studied the task based per formance of the selected techniques in terms of precision and recall. Fig. 8 summarizes the precision results of all techniques obtained with the *Google1*, *Google2*, *DBPedia5*, and *DBPedia4S* measures. We omit results from the *DBPe dia4A* as our experiments showed that they are not very discriminating. Even though the measures we defined yield a wide range of precision values, their relative ratio is always the same. Since our goal is to comparatively eval uate event detection techniques, we conclude that our measures are sound with respect to this criterion. Again, *Top15* and *LDA500* stand out with higher precision values than the other three techniques. The reason for this result is that our precision measures are slightly biased towards approaches that report duplicate or repeated events.

Fig. 9 shows the recall results for the *Bloom1D* measure. *Bloom2D* is omitted as the results are almost exactly the same. First of all, it can be seen from the figure that the recall of all techniques is relatively low at 10 20%. Note that our recall measure is based on the Bloomberg news website, which lists an average of 200 topics per day. Even though techniques were configured to report about 1.5× as many events, our recall measure is nevertheless ambi tious. For example, it is difficult to imagine that enough people will tweet about a topic such as Heathrow's cargo statistics in order to detect it as an event. However, since

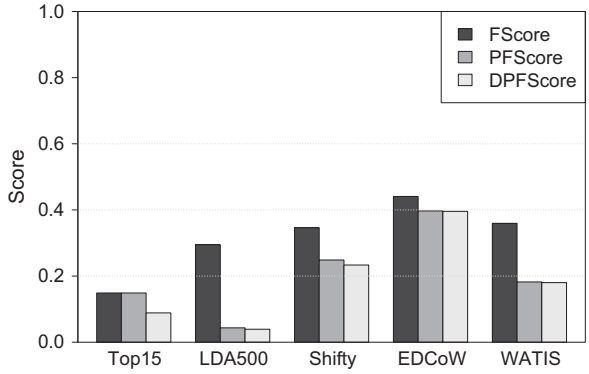

Fig. 10. Average rating scores.

we are only interested in relative measures, these low recall figures are not a problem. Rather, we can observe that *Top15* and *LDA500* generally have a lower recall than the other three techniques. As this outcome is to be expected due to the high duplicate event detection rate of these techniques, we can again conclude that our measure for recall is sound.

In order to summarize the most discriminating measures presented in this article, we define three scoring functions that can be used to compare the run time and task based performance of event detection techniques. The three scoring functions are defined as follows:

$$\textbf{FScore} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{4}$$

$$\textbf{PFScore} = (\text{FScore} \times \text{performance}) \tag{5}$$

$$\textbf{DPFScore} = \text{PFScore} \times (1 \quad \mathit{DEDR}) \tag{6}$$

The first score, *FScore*, denotes the F_1 score that is calculated by using the value of the *Google1* and *Bloom1D* measures for precision and recall, respectively. Alternatively, using *DBPedia5* leads to very similar results. The second score, *PFScore*, also factors in the performance rate of the technique. Performance values are normalized to the range [0…1] by setting the maximum processing rate that we measured to 1. Note that we almost get the same normalized performance score for the techniques, regardless of whether performance was measured on *M*1 or *M*2. Finally, the last measure, *DPFScore*, also includes the duplicate event detection rate of the technique. In the following, we have used the value of the *FDEDR* measure to calculate *DPFScore*.

Based on these definitions, Fig. 10 shows the scores that were assigned to each of the five techniques as averages over the four days in the evaluation data set. Even though *Top15* scores relatively high in terms of precision, its *FScore* is low due to a poor recall because of duplicates. As *Top15* is consistently the fastest technique in our experiments, its *PFScore* is equal to its *FScore*. The high *DEDR* of *Top15* has a noticeable negative effect on its *DPFScore*. *LDA500*'s *FScore* is relatively high, but comes at a high performance penalty, which negatively affects both its *PFScore* and *DPFScore*. Based on these results, we can conclude that neither *Top15* nor *LDA500* is suitable event detection techniques. This result is not surprising as both of these techniques have originally not been developed for this task.

In contrast, the scores of *Shifty*, *EDCoW*, and *WATIS* are much better. In particular, none of these techniques suffer significantly from duplicate event detection. *Shifty* and *WATIS* have a similar *FScore*, but are both negatively affected by their performance score. However, since *Shifty*'s streaming algorithm was forced to an hourly reporting scheme for the sake of comparability, this score is still a good result for our technique. *EDCoW* scores impressive results for all scoring functions, which confirms that its status as the most cited event detection technique is well deserved. This work however is the first to provide comparative and quantitative evidence for *EDCoW*'s quality.

Finally, we note that duplicate events are not always undesired, e.g., when tracking re occurring events or changes in event descriptions. The need to study event detection techniques in both settings motivates our separate definitions of *FScore*, *PFScore*, and *DPFScore*. Both *LDA500* and *Top15* could be extended to explicitly avoid the detection of duplicate events. However, since the other techniques do allow for duplicates, we have chosen not to do so in this study.

6. Conclusion

In this article, we addressed the problem of comparatively and quantitatively studying the task based and run time performance of state of the art event detection techniques for Twitter. In order to do so, we have presented a two pronged approach. First, we ensure comparable run time performance results by providing streaming implementations of all techniques based on a data stream management system. Second, we propose several new measures that can assess the relative task based performance of event detection techniques. The detailed study described in this article has shown that these measures are sound and which of them are most discriminating. Finally, we defined scoring functions based on selected measures that revealed how the different techniques relate to each other as well as where their strengths and weaknesses lie.

As immediate future work, we plan to take advantage of our platform based approach to study further techniques, e.g., *enBlogue* [6] and the approach of Petrović et al. [39]. At the same time, the currently implemented techniques could be improved to process data continuously. Furthermore, the influence of the pre processing on run time and task based performance should be studied. In our platform based approach, we can easily remove existing operators (e.g., retweet filtering) and replace them with new operators (e.g., part of speech tagging or named entity recognition). Finally, a deeper evaluation of how the different parameters of a technique influence the trade off between run time and task based performance could give rise to adaptive event detection techniques.

At the time of writing this article, some of these works have already started and Weiler et al. [48] report on initial results that confirm the outcome of the evaluation

presented here based on further measures and additional event detection techniques.

Acknowledgment

This work is supported by Grant No. GR 4497/4 of the Deutsche Forschungsgemeinschaft (DFG). Additionally, the authors would like to thank Christina Papavasileiou and Harry Schilling for their contributions to the implementation of WATIS and EDCoW.

References

[1] D.J. Abadi, Y. Ahmad, M. Balazinska, U. Çetintemel, M. Cherniack, J. Hwang, W. Lindner, A. Maskey, A. Rasin, E. Ryvkina, N. Tatbul, Y. Xing, S.B. Zdonik, The design of the borealis stream processing engine, In: Proceedings of the International Conference on Innovative Data Systems Research (CIDR), 2005, pp. 277–289.

[2] D.J. Abadi, D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, S. Zdonik, Aurora: a new model and architecture for data stream management, VLDB J. 12 (2) (2003) 120–139.

[3] C.C. Aggarwal, K. Subbian, Event detection in social streams, In: Proceedings of the SIAM International Conference on Data Mining (SDM), 2012, pp. 624–635.

[4] L.M. Aiello, G. Petkos, C. Martin, D. Corney, S. Papadopoulos, R. Skraba, A. Göker, I. Kompatsiaris, Sensing trending topics in twitter, IEEE Trans. Multimedia 15 (6) (2013) 1268–1282.

[5] J. Allan, Topic Detection and Tracking: Event-based Information Organization, Kluwer Academic Publishers, Norwell, MA, USA, 2002.

[6] F. Alvanaki, S. Michel, K. Ramamritham, G. Weikum, See What's enBlogue: real-time emergent topic identification in social media, In: Proceedings of the International Conference on Extending Database Technology (EDBT), 2012, pp. 336–347.

[7] A. Arasu, S. Babu, J. Widom, The CQL continuous query language: semantic foundations and query execution, VLDB J. 15 (2) (2006) 121–142.

[8] E. Bahir, A. Peled, Identifying and tracking major events using geo-social networks, Soc. Sci. Comput. Rev. 31 (4) (2013) 458–470.

[9] H. Becker, M. Naaman, L. Gravano, Beyond trending topics: real-world event identification on twitter, In: Proceedings of the International Conference on Weblogs and Social Media (ICWSM), 2011, pp. 438–441.

[10] J. Benhardus, Streaming trend detection in twitter, National Science Foundation REU for Artificial Intelligence, Natural Language Processing and Information Retrieval, University of Colarado, Final Report, 2010, pp. 1–7.

[11] D.M. Blei, A.Y. Ng, M.I. Jordan, Latent Dirichlet allocation, J. Mach. Learn. Res. 3 (2003) 993–1022.

[12] K. Bontcheva, D. Rout, Making sense of social media streams through semantics: a survey, Semantic Web 5 (5) (2014) 373–403.

[13] M. Cataldi, L.D. Caro, C. Schifanella, Emerging topic detection on twitter based on temporal and social terms evaluation, In: Proceedings of the Workshop on Multimedia Data Mining (MDMKDD), 2010, pp. 4:1–4:10.

[14] M. Cheong, V. Lee, Integrating web-based intelligence retrieval and decision-making from the twitter trends knowledge base, In: Proceedings of the Workshop on Social Web Search and Mining (SWSM), 2009, pp. 1–8.

[15] M. Cordeiro, Twitter event detection: combining wavelet analysis and topic inference summarization, In: Proceedings of the Doctoral Symposium on Informatics Engineering (DSIE), 2012.

[16] A. Culotta, Towards detecting influenza epidemics by analyzing twitter messages, In: Proceedings of the Workshop on Social Media Analytics (SOMA), 2010, pp. 115–122.

[17] A. Farzindar, W. Khreich, A survey of techniques for event detection in Twitter, Comput. Intell. 31 (1) (2015) 132–164.

[18] A. Guille, C. Favre, Mention-anomaly-based event detection and tracking in twitter, In: Proceedings of the International Conference on Advances in Social Networks Analysis and Mining (ASONAM), 2014, pp. 375–382.

[19] G. Ifrim, B. Shi, I. Brigadir, Event detection in twitter using aggressive filtering and hierarchical tweet clustering, In: Proceedings of the Workshop on Social News on the Web (SNOW), 2014, pp. 33–40.

[20] P. Indyk, R. Motwani, Approximate nearest neighbors: towards removing the curse of dimensionality, In: Proceedings of the Annual ACM Symposium on Theory of Computing (STOC), 1998, pp. 604–613.

[21] S. Ishikawa, Y. Arakawa, S. Tagashira, A. Fukuda, Hot topic detection in local areas using Twitter and Wikipedia, In: Proceedings of the Workshop on Complex Sciences in the Engineering of Computing Systems (CSECS), 2012, pp. 1–5.

[22] C.-H. Lee, H.-C. Yang, T.-F. Chien, W.-S. Wen, A novel approach for event detection by mining spatio-temporal information on microblogs, In: Proceedings of the International Conference on Advances in Social Networks Analysis and Mining (ASONAM), 2011, pp. 254–259.

[23] R. Lee, K. Sumiya, Measuring geographical regularities of crowd behaviors for Twitter-based geo-social event detection, In: Proceedings of the Workshop on Location Based Social Networks (LBSN), 2010, pp. 1–10.

[24] C. Li, A. Sun, A. Datta, Twevent: segment-based event detection from tweets, In: Proceedings of the International Conference on Information and Knowledge Management (CIKM), 2012, pp. 155–164.

[25] J. Li, D. Maier, K. Tufte, V. Papadimos, P.A. Tucker, No pane, no gain: efficient evaluation of sliding-window aggregates over data streams, SIGMOD Rec. 34 (1) (2005) 39–44.

[26] J. Li, K. Tufte, V. Shkapenyuk, V. Papadimos, T. Johnson, D. Maier, Out-of-order processing: a new architecture for high-performance stream systems, In: PVLDB, vol. 1 (1), 2008, pp. 274–288.

[27] R. Long, H. Wang, Y. Chen, O. Jin, Y. Yu, Towards effective event detection, tracking and summarization on microblog data, In: Proceedings of the International Conference on Web-age Information Management (WAIM), 2011, pp. 652–663.

[28] A. Madani, O. Boussaid, D.E. Zegour, What's happening: a survey of tweets event detection, In: Proceedings of the International Conference on Communications, Computation, Networks and Technologies (INNOV), 2014, pp. 16–22.

[29] D. Maier, M. Grossniklaus, S. Moorthy, K. Tufte, Capturing episodes: may the frame be with you, In: Proceedings of the International Conference on Distributed Event-Based Systems (DEBS), 2012, pp. 1–11.

[30] A. Marcus, M.S. Bernstein, O. Badar, D.R. Karger, S. Madden, R.C. Miller, TwitInfo: aggregating and visualizing microblogs for event exploration, In: Proceedings of the International Conference on Human Factors in Computing Systems (SIGCHI), 2011, pp. 227–236.

[31] C. Martin, D. Corney, A. Göker, A. Macfarlane, Mining newsworthy topics from social media, In: Proceedings of the BCS SGAI Workshop on Social Media Analysis, 2013, pp. 35–46.

[32] M. Mathioudakis, N. Koudas, TwitterMonitor: trend detection over the twitter stream, In: Proceedings of the International Conference on Management of Data (SIGMOD), 2010, pp. 1155–1158.

[33] P. Meladianos, G. Nikolentzos, F. Rousseau, Y. Stavrakas, M. Vazirgiannis, Degeneracy-based real-time sub-event detection in twitter stream, In: Proceedings of the International Conference on Weblogs and Social Media (ICWSM), 2015, pp. 248–257.

[34] M. Naaman, J. Boase, C.-H. Lai, Is it really about me? Message content in social awareness streams, In: Proceedings of the Conference on Computer Supported Cooperative Work (CSCW), 2010, pp. 189–192.

[35] K. Nishida, T. Hoshide, K. Fujimura, Improving tweet stream classification by detecting changes in word probability, In: Proceedings of the International Conference on Research and Development in Information Retrieval (SIGIR), 2012, pp. 971–980.

[36] A. Nurwidyantoro, E. Winarko, Event detection in social media: a survey, In: Proceedings of the International Conference on ICT for Smart Society (ICISS), 2013, pp. 1–5.

[37] M. Osborne, S. Petrović, R. McCreadie, C. Macdonald, I. Ounis, Bieber no more: first story detection using Twitter and Wikipedia, In: Proceedings of the Workshop on Time-aware Information Access (TAIA), 2012.

[38] R. Parikh, K. Karlapalem, ET: events from tweets, In: Proceedings of the International Conference on World Wide Web (WWW) (Companion Volume), 2013, pp. 613–620.

[39] S. Petrović, M. Osborne, V. Lavrenko, Streaming first story detection with application to twitter, In: Proceedings of the Conference on the North American Chapter of the Association for Computational Linguistics (HLT), 2010, pp. 181–189.

[40] A. Ritter, Mausam, O. Etzioni, S. Clark, Open domain event extraction from twitter, In: Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD), 2012, pp. 1104–1112.

[41] T. Sakaki, M. Okazaki, Y. Matsuo, Earthquake shakes twitter users: real-time event detection by social sensors, In: Proceedings of the

International Conference on World Wide Web (WWW), 2010, pp. 851–860.
[42] J. Sankaranarayanan, H. Samet, B.E. Teitler, M.D. Lieberman, J. Sperling, TwitterStand: news in tweets, In: Proceedings of the International Conference on Advances in Geographic Information Systems (SIGSPATIAL), 2009, pp. 42–51.
[43] K. Spärck Jones, A statistical interpretation of term specificity and its application in retrieval, In: Document Retrieval Systems, Taylor Graham Publishing, 1988, pp. 132–142.
[44] N.A. Thapen, D.S. Simmie, C. Hankin, The Early Bird Catches The Term: Combining Twitter and News Data For Event Detection and Situational Awareness. CoRR abs/1504.02335, 2015.
[45] A. Toshniwal, S. Taneja, A. Shukla, K. Ramasamy, J.M. Patel, S. Kulkarni, J. Jackson, K. Gade, M. Fu, J. Donham, N. Bhagat, S. Mittal, D.V. Ryaboy, Storm @Twitter, In: Proceedings of the International Conference on Management of Data (SIGMOD), 2014, pp. 147–156.
[46] M. Walther, M. Kaisser, Geo-spatial event detection in the twitter stream, In: Proceedings of the European Conference on Information Retrieval (ECIR), 2013, pp. 356–367.
[47] A. Weiler, M. Grossniklaus, M.H. Scholl, Event identification and tracking in social media streaming data, In: Proceedings of the EDBT Workshop on Multimodal Social Data Management (MSDM), 2014, pp. 282–287.
[48] A. Weiler, M. Grossniklaus, MH. Scholl, Evaluation measures for event detection techniques on twitter data streams, In: Proceedings of the British International Conference on Databases (BICOD), 2015, pp. 108–119.
[49] A. Weiler, M. Grossniklaus, M.H. Scholl, Run-time and task-based performance of event detection techniques for twitter, In: Proceedings of the International Conference on Advanced Information Systems Engineering (CAiSE), 2015, pp. 35–49.
[50] A. Weiler, M.H. Scholl, F. Wanner, C. Rohrdantz, Event identification for local areas using social media streaming data, In: Proceedings of the Workshop on Databases and Social Networks (DBSocial), 2013, pp. 1–6.
[51] J. Weng, B.-S. Lee, Event detection in twitter, In: Proceedings of the International Conference on Weblogs and Social Media (ICWSM), 2011, pp. 401–408.
[52] W. Yang, I.G. Zurbenko, Kolmogorov–Zurbenko Filters, Wiley Interdiscip. Rev.: Comput. Stat. 2 (3) (2010) 340–351.
[53] X. Zhou, L. Chen, Event detection over Twitter social media streams, VLDB J. 23 (3) (2014) 381–400.
[54] M. Zimmermann, I. Ntoutsi, Z.F. Siddiqui, M. Spiliopoulou, H.-P. Kriegel, Discovering global and local bursts in a stream of news, In: Proceedings of the Symposium on Applied Computing (SAC), 2012, pp. 807–812.