Towards Adaptive Event Detection Techniques for the Twitter Social Media Data Stream

Michael Grossniklaus, Marc H. Scholl, and Andreas Weiler
Department of Computer and Information Science
University of Konstanz, Germany
firstname.lastname@uni-konstanz.de

Abstract

Social media data streams are an invaluable source for timely and up-to-date information about current events. As a consequence, several event detection techniques have been proposed in the literature in order to tap this information source. However, most of these proposals focus on the information extraction aspect of the problem and tend to ignore the streaming nature of the input. The work conducted in our research group therefore intends to address these stream-related challenges, such as detecting events incrementally, reporting them in (near) real-time, and coping with fluctuations and spikes in the social media data stream. In this article, we report on the results that we obtained so far and outline our research agenda for the remainder of this work.

1 Introduction

Twitter currently has 320 million monthly active users who author over 500 million *tweets* per day that consist of up to 140 characters each.[1] These impressive usage statistics make Twitter the most popular and fastest-growing microblogging service on the planet. In the domain of social media, microblogging enables users to send short messages, links, and audiovisual content to a network of *followers*, as well as to their own public timeline. Due to their brevity, tweets are an ideal mobile communication medium, which is evidenced by the fact that 80% of Twitter's active users are on mobile devices. As a consequence, several proposals have been made to leverage social media data streams as "social sensors" [15] in order to obtain information about current events as they unfold. For example, Twitter data has been used to alert people in case of an outbreak of an infectious disease [9], to quickly respond to natural disasters [15], and to monitor political elections [21].

The problem of detecting events in text-based corpora is not a novel one and has been addressed by research from the area of Topic Detection and Tracking (TDT) for traditional media such as newspaper archives and news websites. In these domains, an *event* is defined as a real-world occurrence that takes place in a certain geographical location and over a certain time period [3]. In comparison to these information sources, social media data streams such as Twitter introduce additional challenges. First, tweets are much shorter than traditional documents and therefore harder to classify. Second, tweets do not undergo an editorial process and can thus

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

[1] https://about.twitter.com/company (November 24, 2015)

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.

The techniques that have been proposed for event detection in social media and, in particular, for Twitter have more or less focused exclusively on the information extraction aspect of the problem. Because of this research direction, the challenges that are related to the streaming nature of the input data have so far been largely ignored by these approaches. For example, many techniques use (large) tumbling windows to process the stream, rather than online or streaming algorithms, and are therefore often unable to report events in (near) real-time. Furthermore, event detection often depends on a complex set of parameters, such as thresholds that control what is considered to be an event. Existing approaches typically assume that these parameters can be calibrated empirically by running the technique on sample data until it produces the desired result. Since the data in the stream may change both qualitatively and quantitatively over time, we argue that techniques that are based on fixed parameters are neither realistic nor feasible.

The work that our research group conducts on this topic intends to address this need for *streaming* and *adaptive* event detection techniques for Twitter. Due to this focus, our work is situated in the area of Data Stream Management Systems (DSMS) research. Since event detection and tracking is a vast field of research in itself, we concentrate on the specific task of first story detection, *i.e.*, the detection of general (unknown) events, which has been defined as a subtask of TDT [3]. In this article, we report results that we obtained so far and outline future research directions. We begin in Section 2 by giving a brief overview over the state of the art in event detection techniques for Twitter, including our own. Section 3 presents an evaluation platform that supports the systematic study and comparison of such techniques. In our work, we use this platform in order to gain a better understanding of how different parameter settings affect the trade-off between processing time and result quality in existing event detection techniques. In Section 4, we outline how this empirical research will contribute to building event detection techniques that can adapt to content and volume changes in the social media data stream. Finally, we give concluding remarks in Section 5.

2 Event Detection Techniques

In recent years, numerous techniques to detect events in social media data streams and, in particular, Twitter have been proposed. Rather than presenting a comprehensive survey of event detection techniques, we introduce five examples in this section. The first three examples are existing approaches that we studied in detail in previous work [18, 19]. The remaining two examples are approaches that we proposed ourselves in an effort to develop techniques that process their input in a fully streaming and incremental manner. For a more detailed discussion of the state of the art, we refer the interested reader to one of the existing surveys on this subject. For example, the survey of Nurwidyantoro and Winarko [14] summarizes 11 techniques to detect disaster, traffic, outbreak, and news events. The survey of Madani *et al.* [12] 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 presented by Bontcheva and Rout [7]. Finally, Farzindar and Khreich [10] conducted an extensive survey of techniques that are specifically intended to detect events in the Twitter social media data stream.

EDCoW (Event Detection with Clustering of Wavelet-based Signals) [21] is one of the most-cited event detection techniques. In the first step, this algorithm applies a time-based tumbling window of size s to the stream to partition it into non-overlapping segments. For each window instance, it then builds the DF-IDF signals for each distinct term in the segment. The DF-IDF is similar to the TF-IDF that is commonly used in information retrieval to measure the importance of a word (term). Since multiple occurrences of the same term in one document (tweet) are typically associated with the same event, the DF-IDF only counts the number of documents that contain the term. On each of these signals, a discrete wavelet analysis is performed in order to build a second signal in which each data point summarizes a sequence of values of length Δ from the first signal. Trivial terms are filtered out in the next step by checking the corresponding signal auto-correlations against a threshold γ. A modularity-based graph partitioning technique is then applied to the remaining terms in order to form events by clustering them. Finally, another threshold ϵ is used to filter out insignificant events. In the original paper, EDCoW is evaluated on a month's worth of Twitter data that was gathered in June 2010 by collecting the tweets from the top 1000 Singapore-based users and their friends within two hops. The initial window size s was set to a whole day.

The *WATIS* (Wavelet Analysis Topic Inference Summarization) [8] event detection technique is similar to *EDCoW* in that it first segments the stream into time-based windows of size s and then builds the DF-IDF signals for each distinct term. However, before these signals are further analyzed, they are smoothed using an Adaptive Kolmogorov-Zurbenko (KZA) [22] low-pass filter that calculates a moving average with i_{kz} iterations over n intervals. Based on these smoothed signals a time-frequency representation is constructed using continuous wavelet transformation. On this representation two wavelet analyses are performed in order to detect unexpected shifts in the frequency of a term: the tree map of the continuous wavelet extrema and the local maxima detection. Finally, Latent Dirichlet Allocation (LDA) [6] with i_{lda} iterations is used to enrich event terms with co-occurring terms. The technique is evaluated by applying it to a dataset consisting of 13.6 million tweets, which were gathered over a period of eight days. In this evaluation, the technique was used to process the entire dataset at once, *i.e.*, the initial window has a size s of 192 hours.

As the previous approaches, *enBlogue* [4] uses a time-based tumbling window of size s to segment the stream before processing it.[2] For each window, so-called "seed tags" are identified based on their popularity, which is computed as the relative frequency of a term in a window. Topics are modeled as pairs of tags, which are formed by measuring the correlation between two documents that contain the tags using the Jaccard coefficient. A topic is considered to be an emergent event if its current behavior is different from its previous behavior, *i.e.*, if there is an unexpected shift in its popularity. All topics are then ranked according to their degree of emergence and the top k topics are reported as events. In the original evaluation, the size s of the initial window is set to one hour and the result quality of the detected events is assessed based on a user study.

To conclude this section, we present two simple event detection techniques that we developed in previous work. The goal of both techniques is to reduce the latency with which events can be reported, but each technique follows a different approach to do so. In contrast to the techniques described above, *LLH* [20] reduces the processing required to detect events. It simply calculates a log-likelihood measure for the frequency of all distinct terms in the current time-based tumbling window ($s = 1$ hour) against their frequency in the previous window. For the top N terms ranked according to this ratio, the corresponding top four most co-occurring terms are computed and the resulting term set is reported as an event. Our second technique, *Shifty* [17], aims to reduce latency by using both shorter and sliding windows to segment the stream. It detects events by monitoring the IDF values of distinct terms in successive sliding windows. For each term in a (tumbling) window of size $s = 1$ minute, Shifty computes the IDF value and filters out terms with an IDF value above the window average. In order to calculate the IDF shift for each remaining term from one window to the next, a window with size s_1 that slides with range r_1 is built in the next step. Only terms with a shift above the average shift are retained. In the last step, another sliding window with size s_2 that slides with range r_2 is built. This window is used to calculate the total shift value as the sum of all shift values of the sub-windows. Terms with a total shift value greater than Ω are detected as events and reported together with their top four co-occurring terms.

[2] In their original paper, Alvanaki *et al.* [4] state that *enBlogue* uses sliding windows. However, only the value for the size of the window is given, while the value for the slide range is never mentioned. Personal communication with one of the authors confirmed that indeed a tumbling window is used.

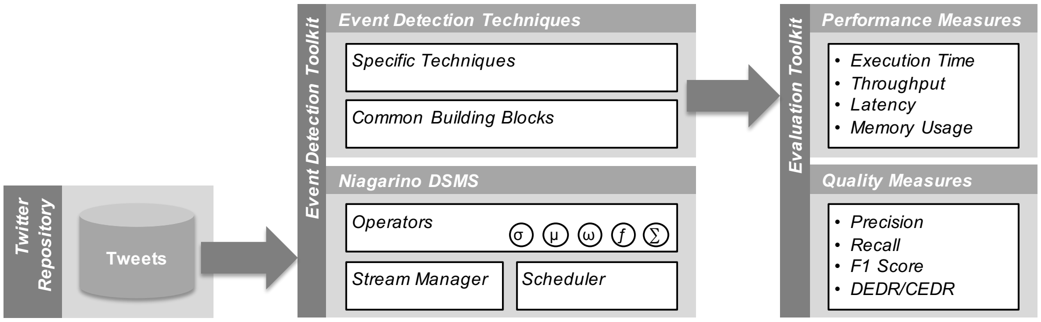

Figure 1: Overview of the evaluation platform for Twitter event detection techniques.

3 Evaluation Platform

In order to understand how our own approaches compete with the current state of the art, we designed and developed an evaluation platform for event detection techniques. Figure 1 gives a schematic overview of this platform and its components. The first component is a tweet repository that we host on our servers, which contains a randomly sampled 10% sub-stream of the public live stream of Twitter. The repository is continuously updated with new tweets that we have been gathering since 2012 using the Twitter Streaming API[3] with the so-called "Gardenhose" access level. At the moment, the repository contains about 10 TB of data, which corresponds to over 50 billion tweets at an average rate of 2.5 million tweets/hour.

The next component of our evaluation platform is a toolkit that can be used to experiment with existing and new event detection techniques in a controlled environment. In order to obtain reliable performance measurements that can be compared fairly, we propose to realize all studied event detection techniques in a DSMS. For this purpose, we currently use *Niagarino*[4], a lightweight and extensible DSMS that we develop and maintain in our research group. The main purpose of Niagarino is to serve as an easy-to-use research platform for streaming applications such as the ones presented in this article. Many of its concepts can be traced back to a series of pioneering data stream management systems, such as Aurora [2], Borealis [1], and STREAM/CQL [5]. In particular, Niagarino is an offshoot of NiagaraST [11], with which it shares the most common ground. The representation of event detection techniques as query plans is one of the key benefits of our approach. Using Niagarino's textual plan description format or the graphical plan builder that we are currently developing, new techniques can be easily developed by modifying existing plans or by creating new ones. In order to further simplify this task, our toolkit already provides a number of building blocks that are common to many event detection techniques, such as operators to tag tweets with their languages, to filter tweets that contain spam, and to remove terms that are considered noise or stop-words. Finally, additional operators that cannot be assembled from already existing ones can be added to our toolkit with limited programming overhead due to Niagarino's modular architecture.

The last component of our platform is a toolkit to evaluate event detection techniques. By providing this toolkit, we address two shortcomings of the current state of the art. First, very few authors of existing event detection techniques have evaluated the performance of their approach in comparison to other techniques. Nevertheless, factors such as throughput, latency, and memory usage are particularly crucial to the feasibility of an approach in a highly volatile streaming setting such as Twitter. Our toolkit therefore provides a number of measures that can be used to study and compare these performance characteristics of event detection techniques. Second, the quality of the results, *i.e.*, the validity of the detected events, is another factor that is paramount to the usefulness of an approach. While some authors of previous approaches have evaluated the results of their technique using a manually crafted ground truth or based on a user study, very few have compared their results

[3] `https://dev.twitter.com` (November 24, 2015)

[4] `http://www.informatik.uni-konstanz.de/grossniklaus/software/niagarino/` (November 24, 2015)

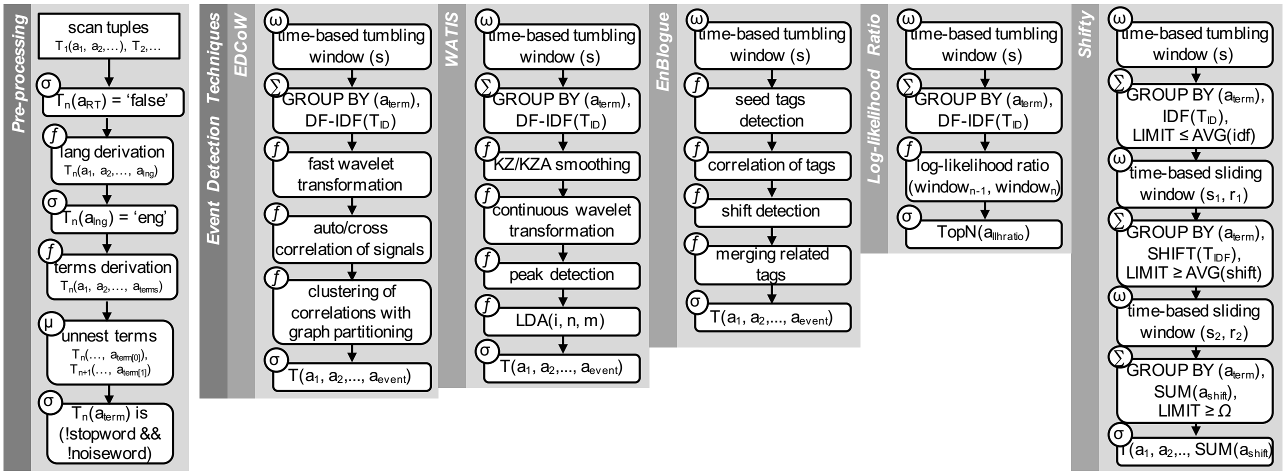
Pre-processing
scan tuples
T1(a1, a2,...), T2,...
σ Tn(aRT) = 'false'
f lang derivation
Tn(a1, a2,..., alng)
σ Tn(alng) = 'eng'
f terms derivation
Tn(a1, a2,..., aterms)
μ unnest terms
Tn(..., aterm[0]),
Tn+1(..., aterm[1])
σ Tn(aterm) is (!stopword && !noiseword)
Event Detection Techniques
EDCoW
ω time-based tumbling window (s)
Σ GROUP BY (aterm), DF-IDF(TID)
f fast wavelet transformation
f auto/cross correlation of signals
f clustering of correlations with graph partitioning
σ T(a1, a2,..., aevent)
WATIS
ω time-based tumbling window (s)
Σ GROUP BY (aterm), DF-IDF(TID)
f KZ/KZA smoothing
f continuous wavelet transformation
f peak detection
f LDA(i, n, m)
σ T(a1, a2,..., aevent)
EnBlogue
ω time-based tumbling window (s)
f seed tags detection
f correlation of tags
f shift detection
f merging related tags
σ T(a1, a2,..., aevent)
Log-likelihood Ratio
ω time-based tumbling window (s)
Σ GROUP BY (aterm), DF-IDF(TID)
f log-likelihood ratio (windown-1, windown)
σ TopN(allhratio)
Shifty
ω time-based tumbling window (s)
Σ GROUP BY (aterm), IDF(TID), LIMIT ≤ AVG(idf)
ω time-based sliding window (s1, r1)
Σ GROUP BY (aterm), SHIFT(TIDF), LIMIT ≥ AVG(shift)
ω time-based sliding window (s2, r2)
Σ GROUP BY (aterm), SUM(ashift), LIMIT ≥ Ω
σ T(a1, a2,.., SUM(ashift)

Figure 2: Niagarino query plans for the preprocessing and the five example event detection techniques.

to competing approaches. One reason for this lack of comparative and systematic evaluation is that crafting a ground truth manually does not scale to the volume of the Twitter data stream and conducting user studies is time-consuming and expensive. In our work [18, 19], we have therefore focused on quality measures that can be applied automatically. For example, we propose to measure precision by matching detected events to a combination of Web search-engine results and knowledge bases such as DBpedia[5]. We follow a similar approach to measure recall by crawling the daily headlines of new archives such as Bloomberg and the New York Times. Based on precision and recall, we are able to calculate the F_1 score for a studied technique. It is important to note that values computed by these measures cannot be used to support any absolute conclusions about a single technique. However, they can be used to draw relative conclusions by comparing different techniques or multiple configurations of the same technique.

We have used this platform to conduct an extensive study of the event detection techniques introduced in the previous section. Figure 2 shows the corresponding Niagarino query plans as well as the preprocessing subplan that is common to all approaches. As a complete discussion of the results is out of the scope of this article, we refer the interested reader to our previous work. Weiler *et al.* [18] presents the evaluation measures that we defined. In order to demonstrate that these measure are useful, we apply them to both well-known event detection techniques and baseline approaches. The comparison of the results clearly show that our measures can discriminate between actual event detection techniques and approaches that, for example, simply select random or most frequently occurring terms. In Weiler *et al.* [19], we use these measures to study a number of event detection techniques in terms of performance and result quality. With respect to result quality (F_1 score) our study confirms that the status of both *EDCoW* and *WATIS* as frequently cited event detection techniques is well-deserved as they detect events more reliably than other techniques. However, this result quality comes at the price of lower throughput (tweets/second). In particular, *WATIS* would not be capable of handling the full 100% stream of Twitter on current server hardware, owing to the expensive LDA operator towards the end of the query network. In contrast, our own techniques, *LLH* and *Shifty*, score very well with respect to this performance measure. While *LLH* scores quite low in terms of result quality, *Shifty* is a close runner-up behind the more complex event detection techniques. We therefore conclude that *Shifty* represents an interesting trade-off between performance and result quality that we will investigate further in the future.

[5] `http://dbpedia.org` (November 24, 2015)

4 Future Work

Building on the work presented in this article, we are currently conducting research to address the need for *adaptive* event detection techniques for the Twitter social media data stream. In order to do so, we follow two lines of work.

First, we are studying methods to automatically determine the parameter settings of event detection techniques. As outlined in Section 2, current techniques depend on a number of parameters that directly affect performance and result quality of an approach. The ability to determine and adjust these parameters automatically is important for several reasons. On the one hand, it is unrealistic to assume that such parameter values can be determined based on a small sample of the stream during the design of the technique. This assumption has often been criticized before, for instance by Farzindar and Khreich [10]. On the other hand, the social media data stream may undergo qualitative and quantitative changes, which require parameter adjustments. Using our implementations of existing techniques that we described in this article, we study the effects of different parameter settings for each technique on a number of segments of the real-life Twitter data stream. The goal of this initial empirical study is to develop quality-of-service models for selected techniques that describe the relationship between performance and result quality. Based on these quality-of-service models, we envision that adaptive techniques can trade-off result quality for performance in case of changes in the volume of tweets that need to be processed. In the past, quality-of-service models have been used successfully to control load shedding [16]. Rather than shedding load, we are interested in using such models to shed processing time, *i.e.*, to dynamically reconfigure techniques to perform, for example, fewer LDA iterations or low-pass filter steps.

Our second line of work researches new forms of content-based stream segmentation for event detection techniques. All existing techniques use (large) time-based windows to process the unbounded stream of tweets. In previous and ongoing work [13], we criticized the use of simple time and tuple-based windows in today's complex data-stream applications and instead proposed data-driven windows, so-called frames. We are interested in studying whether frames as a method to segment streams can contribute to better result quality of event detection techniques. The quality improvements that can be obtained with frames stem from the fact that frames adapt the segmentation of the stream to the observed data rather than segmenting it into predefined intervals as windows do. Therefore, in order to use frames in the setting of streaming social media data analysis, the data that can drive the framing of the stream need to be identified. Since a portion of the Twitter stream contains GPS coordinates, it could, for example, make sense to use a position grid to segment the stream to track how (information about) an event spreads geographically.

5 Summary and Conclusion

Since their inception, DSMSs have been used to realize ever more complex stream processing applications, which often demanded new or extended functionality at the system level. In this article, we focussed on event detection in social media data streams, a relatively new application domain for DSMSs. Unfortunately, most existing event detection techniques have been developed without the support of a DSMS, which makes it difficult to reason about their practical feasibility, in particular with respect to their performance. Therefore, we introduced some well-known event detection techniques in this article and showed how they can be realized as query plans in a DSMS. This representation is one of the key benefits of our approach as it greatly simplifies the creation and modification of event detection techniques. In order to further promote the use of DSMSs in researching such techniques, we have designed and developed a platform that provides toolkits for both the implementation and evaluation of existing and novel approaches. Finally, we outlined open research challenges in this area, such as the need for fully streaming and adaptive event detection techniques. We believe that tackling these challenges will again require new DSMS concepts as, for example, new methods to deal with changes in data volume or to segment the stream in a more flexible manner.

Acknowledgments

The research presented in this article is funded in part by the Deutsche Forschungsgemeinschaft (DFG), Grant No. GR 4497/4: "Adaptive and Scalable Event Detection Techniques for Twitter Data Streams". We would also like to thank our students Christina Papavasileiou, Harry Schilling, and Wai-Lok Cheung for their contributions to the implementations of the *WATIS*, *EDCoW*, and *enBlogue* event detection techniques in Niagarino.

References

[1] Daniel J. Abadi, Yanif Ahmad, Magdalena Balazinska, Ugur Çetintemel, Mitch Cherniack, Jeong-Hyon Hwang, Wolfgang Lindner, Anurag Maskey, Alex Rasin, Esther Ryvkina, Nesime Tatbul, Ying Xing, and Stanley B. Zdonik. The Design of the Borealis Stream Processing Engine. In *Proc. Intl. Conf. on Innovative Data Systems Research (CIDR)*, pages 277–289, 2005.

[2] Daniel J. Abadi, Don Carney, Ugur Çetintemel, Mitch Cherniack, Christian Convey, Sangdon Lee, Michael Stonebraker, Nesime Tatbul, and Stand Zdonik. Aurora: A New Model and Architecture for Data Stream Management. *The VLDB Journal*, 12(2):120–139, 2003.

[3] James Allan. *Topic Detection and Tracking: Event-based Information Organization*. Kluwer Academic Publishers, 2002.

[4] Foteini Alvanaki, Sebastian Michel, Krithi Ramamritham, and Gerhard Weikum. See What's enBlogue: Real-time Emergent Topic Identification in Social Media. In *Proc. Intl. Conf. on Extending Database Technology (EDBT)*, pages 336–347, 2012.

[5] Arvind Arasu, Shivnath Babu, and Jennifer Widom. The CQL Continuous Query Language: Semantic Foundations and Query Execution. *The VLDB Journal*, 15(2):121–142, 2006.

[6] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet Allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003.

[7] Kalina Bontcheva and Dominic Rout. Making Sense of Social Media Streams through Semantics: A Survey. *Semantic Web*, 5(5):373–403, 2014.

[8] Mário Cordeiro. Twitter Event Detection: Combining Wavelet Analysis and Topic Inference Summarization. In *Proc. Doctoral Symposium on Informatics Engineering (DSIE)*, 2012.

[9] Aron Culotta. Towards Detecting Influenza Epidemics by Analyzing Twitter Messages. In *Proc. Workshop on Social Media Analytics (SOMA)*, pages 115–122, 2010.

[10] Atefeh Farzindar and Wael Khreich. A Survey of Techniques for Event Detection in Twitter. *Computational Intelligence*, 31(1):132–164, 2015.

[11] Jin Li, Kristin Tufte, Vladislav Shkapenyuk, Vassilis Papadimos, Theodore Johnson, and David Maier. Out-of-Order Processing: A New Architecture for High-Performance Stream Systems. *PVLDB*, 1(1):274–288, 2008.

[12] Amina Madani, Omar Boussaid, and Djamel Eddine Zegour. What's Happening: A Survey of Tweets Event Detection. In *Proc. Intl. Conf. on Communications, Computation, Networks and Technologies (INNOV)*, pages 16–22, 2014.

[13] David Maier, Michael Grossniklaus, Sharmadha Moorthy, and Kristin Tufte. Capturing Episodes: May the Frame Be with You (Invited Paper). In *Proc. Intl. Conf. on Distributed Event-Based Systems (DEBS)*, pages 1–11, 2012.

[14] Arif Nurwidyantoro and Edi Winarko. Event Detection in Social Media: A Survey. In *Proc. Intl. Conf. on ICT for Smart Society (ICISS)*, pages 1–5, 2013.

[15] Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. Earthquake Shakes Twitter Users: Real-time Event Detection by Social Sensors. In *Proc. Intl. Conf. on World Wide Web (WWW)*, pages 851–860, 2010.

[16] Nesime Tatbul, Ugur Çetintemel, Stanley B. Zdonik, Mitch Cherniack, and Michael Stonebraker. Load Shedding in a Data Stream Manager. In *Proc. Intl. Conf. on Very Large Data Bases (VLDB)*, pages 309–320, 2003.

[17] Andreas Weiler, Michael Grossniklaus, and Marc H. Scholl. Event Identification and Tracking in Social Media Streaming Data. In *Proc. EDBT Workshop on Multimodal Social Data Management (MSDM)*, pages 282–287, 2014.

[18] Andreas Weiler, Michael Grossniklaus, and Marc H. Scholl. Evaluation Measures for Event Detection Techniques on Twitter Data Streams. In *Proc. British Intl. Conf. on Databases (BICOD)*, pages 108–119, 2015.

[19] Andreas Weiler, Michael Grossniklaus, and Marc H. Scholl. Run-time and Task-based Performance of Event De-

tection Techniques for Twitter. In *Proc. Intl. Conf. on Advanced Information Systems Engineering (CAiSE)*, pages 35–49, 2015.

[20] Andreas Weiler, Marc H. Scholl, Franz Wanner, and Christian Rohrdantz. Event Identification for Local Areas Using Social Media Streaming Data. In *Proc. Workshop on Databases and Social Networks (DBSocial)*, pages 1–6, 2013.

[21] Jianshu Weng and Bu-Sung Lee. Event Detection in Twitter. In *Proc. Intl. Conf on Weblogs and Social Media (ICWSM)*, pages 401–408, 2011.

[22] Wei Yang and Igor G. Zurbenko. Kolmogorov-Zurbenko Filters. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(3):340–351, 2010.