Towards A Unified Framework for Event Detection Applications

Rami A. Alghamdi
University of Minnesota, Twin Cities
Minneapolis, MN, USA
algha017@umn.edu

Amr Magdy
University of California, Riverside
Riverside, California, USA
amr@cs.ucr.edu

Mohamed F. Mokbel
Qatar Computing Research Institute
Doha, Qatar
mmokbel@hbku.edu.qa

ABSTRACT

Event detection applications have attracted significant attention with the rise of user-generated spatio-temporal data over the past decade. However, building event detection applications still encounter high cost and effort due to lack of system support. This paper envisions a holistic system approach to support efficient and easy-to-use system infrastructures for building event detection applications. We outline our vision for representing event applications as a set of layered abstractions and discuss potential pathways to realize these abstractions at the system level. Also, We highlight different open problems in different system components.

1 INTRODUCTION

Detecting events from user-generated data has received a significant attention over the past decade from data management and analysis researchers [1–5, 7–11, 14–21, 23–26, 29, 30, 32–38] as well as major corporations such as Thomson Reuters that detects events from news data [20, 21], and governmental units, such as the US Department of Health and Human Services that tracks health-related events [22] and the US Geological Survey that monitors earthquake events [6] from social media data. This is attributed to the availability of massive event-related data that has started to dramatically increase since 2008, the year when Internet connectivity has coupled with mobile devices, as it became much easier for users to contribute data to online platforms. In fact, we are currently witnessing ~48% of Internet traffic through mobile devices worldwide where 21% of such traffic on social media platforms [27]. These percentages are even significantly higher in some localities, e.g., UAE and Saudi Arabia encounter 96% and 88%, respectively, of mobile Internet users [28]. As a result, tens of millions of users can post data that is associated with the location and temporal information around the clock from mobile devices, which has led to unprecedented rates of user-generated spatio-temporal data.

Such an unprecedented explosion of user-generated data, along with its inherent spatio-temporal nature, has motivated a wide variety of event-related applications, ranging from critical and life-saving applications to entertainment and leisure applications. For example, several efforts have successfully designed an early earthquake detection approaches from Twitter feeds, where up to 75% of earthquake detections occurred within the first two minutes from the initial impact and alerts were disseminated earlier than official authorities' first warnings in several cases [10, 11, 26]. Another example of critical applications is detecting criminal and riot activities [3, 19]. Less critical applications include detecting traffic jams events and road accidents to alarm commuters [1, 16, 29, 30, 34] up to discovering breaking news faster than traditional reporting tools [2, 18, 20, 21, 32] and detecting leisure events such as local music concerts, festivals, and celebrations [5, 17].

Despite the plethora of research techniques that investigated supporting effective event detection functionality for different types of events [1–5, 7–11, 14–21, 23–26, 29, 30, 32–38], that are studied in recent surveys [13, 31], it is still labor intensive for developers to build event applications on top of the available rich data sources. In fact, addressing challenges in end-to-end data-to-knowledge pipelines is identified as one of the major challenges in the Beckman report on database research [12]. Quoting the report "*it is still an extremely labor-intensive journey from raw data to actionable knowledge*", thirty top-notch database researchers concluded. This challenge is apparent in event detection applications, where none of the existing systems support scalable infrastructure for ease of building event detection tasks, from data acquisition and preparing to deploying and monitoring. As a result, whoever builds an event detection need to develop major components from scratch, which limits both usability and efficiency and hinders the widespread of using event detection techniques in real-life applications.

In this paper, we envision a declarative system interface that provides SQL-like language to define event detection pipelines. The language allows users to specify high-level details of different phases of the pipeline from preprocessing of different input data sources to producing event summaries for visualization. Under the hood, the system will encapsulate efficient end-to-end modules for building event detection applications. Towards realizing this vision, researchers need to address several challenges. The first challenge is representing event detection with a set of abstract modules that exploit the existing rich literature of event detection techniques. These abstractions should include the common utilities to support: (a) a wide variety of events types, e.g., earthquakes, crimes, traffic jams, and music festivals, (b) diverse data sources and formats, e.g., social media and news feeds from different sources, (c) operating in online and offline modes to detect new events from live streams and historical data, and (d) different levels of time-sensitivity of detected events, e.g., crime events are more time-sensitive than traffic jams. Meeting all such requirements in a unified framework is a major research contribution from a system perspective. Such abstract modules will serve as building blocks specialized for event applications, similar in spirit to SELECT-PROJECT-JOIN building blocks for relational database queries. The second challenge is realizing the developed abstractions in existing data management systems. This realization by itself will require several major system

SSTD, 2019

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

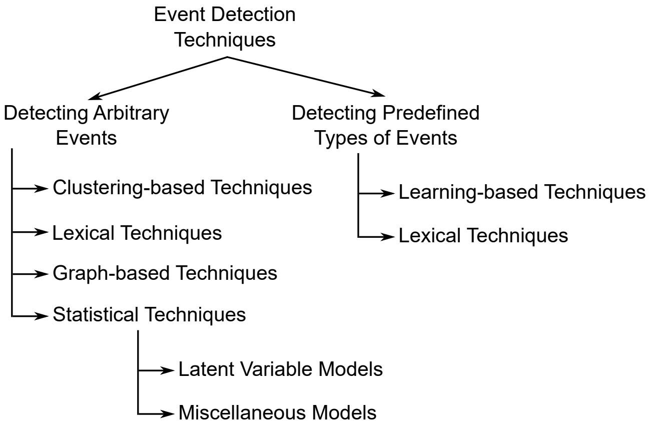
Event Detection Techniques
Detecting Arbitrary Events
Detecting Predefined Types of Events
Clustering-based Techniques
Lexical Techniques
Graph-based Techniques
Statistical Techniques
Latent Variable Models
Miscellaneous Models
Learning-based Techniques
Lexical Techniques

Figure 1: Bird's-eye View of Event Detection Literature

contributions that include developing optimization algorithms to support such abstractions efficiently at a system-level. In analogy with the previous example, plenty of algorithms have been developed to efficiently support join operations and ordering of different relational operators in database systems.

The rest of the paper outlines our vision for the system abstractions and potential pathways to realize them in existing data management systems. Section 2 gives an overview of the existing literature of event detection techniques. Then, Sections 3 and 4 outlines our envisioned framework and its realization in existing systems. Finally, Section 5 concludes the paper.

2 LITERATURE OVERVIEW

Towards our vision for a unified framework for event detection applications, we have extensively surveyed the landscape of existing event detection techniques. The scope of this paper is not providing a detailed review of this rich literature. In fact, recent surveys have already provided such a detailed review [13, 31]. However, we give a summarized overview of this literature as a foundation for our envisioned abstractions for the unified framework. The rest of this section gives an bird's-eye view of the existing literature.

Figure 1 shows an overview of the event detection literature. The literature has two major types of techniques: **(1) type 1 techniques** for *detecting arbitrary events* and **(2) type 2 techniques** for *detecting predefined types of events*. Each of these two types has several sub-categories as depicted in Figure 1. For type 1 techniques, the main objective is detecting any arbitrary event without any prior information about the event type or context. Therefore, the heart of this type is grouping similar data records into cohesive groups to generate a set of candidate events for further processing. As a result, the sub-categories of type 1 techniques are categorized based on the grouping type, which is dominated with clustering algorithms but still includes lexical, statistical, and graph-based techniques. For type 2 techniques, there is prior information about the type of events to be detected, e.g., earthquakes, crimes, or traffic jams. The event type is used to induce contextual information, e.g., crime-related keywords, that can be used to classify data records whether or not they are relevant to this type of events. Thus, the heart of type 2 techniques is a classification technique that decide on the relevance of different data records. This classification could be learning-based or lexical-based as depicted in Figure 1.

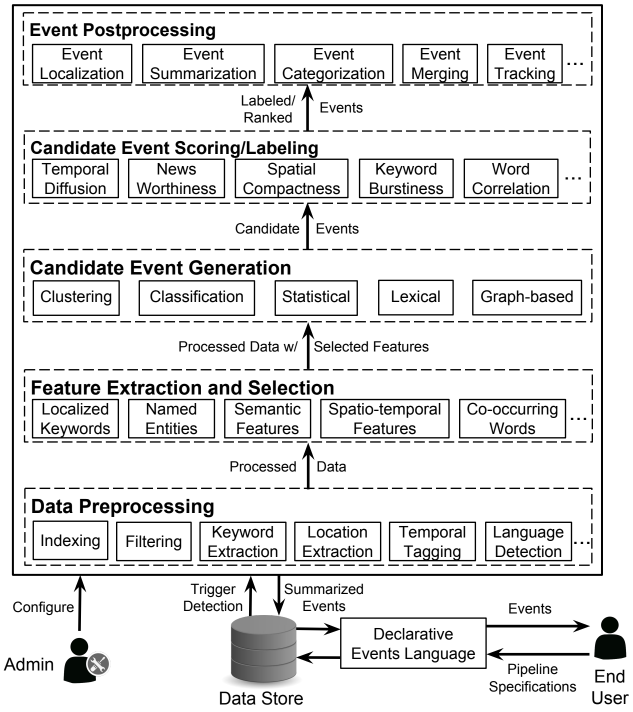

Figure 2: Envisioned Unified Framework Architecture

3 A UNIFIED FRAMEWORK FOR EVENT DETECTION

In this section, we present our vision for a unified end-to-end framework for event detection. The envisioned framework consists of high-level abstractions that are can be supported at a system level through a declarative language interface. The declarative interface will enable a wide variety of end users with different levels of expertise, e.g., SQL developers, to easily build efficient and scalable event detection applications. Based on our extensive survey of the literature, Figure 2 depicts our envisioned unified framework. The framework takes a variety of input data sources that are organized into a scalable data store. The end users can post declarative configurations to define the high-level details of different pipeline stages, e.g., specific preprocessing modules, certain clustering algorithm, and certain scoring function. Then, the relevant data are filtered out from the data store and pipelined into five major sequential layers, namely, *data preprocessing*, *feature extraction and selection*, *candidate event generation*, *candidate event scoring*, and *event postprocessing*. Finally, the output events are returned to end users for visualization and further analysis. The rest of this section briefly outlines the abstractions of each of the five layers in Figure 2.

(1) Data Preprocessing Layer. This layer will be responsible for raw data acquisition and preparation from different data sources to prepare input data records for subsequent layers of processing. The preprocessing layer will include a diverse set of tasks such as filtering, indexing, keyword extraction, geotagging, temporal tagging, and language detection. The specific set of needed preprocessing tasks depend on the input data source. For example, news items come with known languages, no need for language detection that might be needed with social media free text. We next briefly highlight the main preprocessing tasks.

The preprocessing filtering task discards non-event data, e.g., spams, chit-chats, and advertisements. This process is necessary to have a reliable and high accurate result. Moreover, other filtering predicates are used in specific techniques; for example, spatial and temporal predicates are used to filter out data of geographic regions and time periods that are not within the range of interest.

A handful of detection techniques take advantage of spatial, temporal, and spatio-temporal indexes to effectively access data within certain spatial and temporal ranges of interest. The use of indexes is because event-related data are spatial and temporal by nature. For example, many techniques process incoming data in a sliding window where all the data in that window are processed when the window ends, while other techniques employ a simple grid index of the space, and so on.

Keyword, location, and temporal information extraction are key operations for event detection applications. These three types of features represent the significant features that are used in both grouping and classification for both type 1 and type 2 techniques that are introduced in Section 2. Various techniques are used for this extraction including natural language processing, gazetteer-based, and temporal-tagging techniques.

(2) Feature Selection and Extraction Layer. This layer will encapsulate several techniques that take preprocessed data records and associate each of them with various features to be used in subsequent layers. The feature extraction will be mostly through automated techniques while feature selection will be defined within the pipeline specifications through the declarative language interface. Examples of major features are keywords, named entities, spatial locations, temporal information, and semantic features. Some of these features are prepared during the preprocessing stage, and others are not. For example, spatial locations are mostly extracted in preprocessing, while named entities, semantic features, and local keywords that are associated with certain regions are not. Techniques that are used in feature extraction are diverse and include text mining, natural language processing, tokenization, spatial correlation. In general, this layer involves finding data that exhibit different types of high-level characteristics for events generation.

(3) Event Candidate Generation Layer. The input to this layer is the data records associated with different features that are extracted in previous stages. The output is a set of candidate events that are generated through (a) grouping techniques for detecting arbitrary events (type 1 techniques as in Section 2), or (b) classification techniques for detecting predefined types of events (type 2 techniques), e.g., earthquakes, crimes, and traffic jams. The grouping modules will include clustering, lexical-based, graph-based, and statistical techniques. However, clustering will represent the dominated type of grouping techniques in this layer due to its popularity and effectiveness in many existing research techniques. On another hand, the classification techniques will be either learning-based or lexical-based techniques according to the existing literature. However, learning-based classification, e.g., SVMs and regression, will represent the vast majority of encapsulated techniques.

(4) Event Candidate Scoring/Labeling Layer. This layer further enhances the set of generated candidate events from the previous layer through scoring and labeling. For detecting arbitrary events, the generated candidate events usually have a lot of noisy groups that do not represent actual events. Thus, the set of candidate groups are scored or labeled based on their group features, e.g., temporal diffusion, spatial compactness, newsworthiness, word co-occurrences, keyword burstiness, or named entities correlation. Then, top scored candidates or most confidently labeled groups are selected as output events to the next layer while the rest of candidates are discarded as noisy groups. The declarative interface will allow end users to specify certain scoring or labeling methods to define this layer of the pipeline. This specification will depend on the underlying supported application. For example, news event discovery application will prefer newsworthiness ranking measure while localized event detection will prefer spatial compactness.

(5) Event Postprocessing. In the last phase of the event detection pipeline, output events are postprocessed to be readily usable for end users and their analysis tools, e.g., visualization tools. For example, events are summarized or categorized into topics, e.g., politics or cultural, or being attached a spatial location to locate them on maps. At this phase, an event is mostly represented as a group of raw data records collectively are about the event plus all the meta-data extracted from previous phases, e.g., features, score, or label. The postprocessing tasks help end users to interpret and analyze the event, or even following up on tracking the event updates or expiring obsolete events. In some applications, events expire or even evolve to become similar to other events after more data arrive. So, it is essential to allow the user to expire or follow up on events after a while to distinguish it from new coming and similar events. Events also can be merged if the similarity between them is significant which indicate they are about the same event.

4 POTENTIAL REALIZATION PATHWAYS

In this section we briefly discuss three possible pathways to realize the envisioned framework in existing data management systems highlighting their potential advantages and disadvantages.

(1) On-top Approach. One approach to realize the envisioned framework is to build a standalone library on top of the existing systems, e.g., Apache Spark pipelines. In that case, the framework will treat the underlying system as a black box. This mean that the different framework layers will be completely decoupled from the internal operations of the underlying data management system that works as a data store and runtime engine. In this approach, the envisioned framework lives outside the codebase of the core data management engine giving the main advantage of relative simple realization as the complexity of the system internals are hidden. However, one disadvantage in this approach is that it will not fully utilize the optimization opportunities compared to realizing parts of its module inside the system body. For example, performing early pruning based on system indexes could avoid a significant irrelevant data transfer cost to upper level layers. In addition, reordering operations for query optimization will not be an option for an on-top approach realization.

(2) Built-in Approach. Another approach to realize the envisioned detection framework is to tightly couple its different layers with the core data management engine whenever possible. For example, data preprocessing is coupled with the system indexing, which will injected with new operations such as location extraction and language detection. Feature extraction and selection layer will be realized as a new intermediate layer between the indexing and machine learning pipelines, and so on. The expected performance gains of this approach is significant as it has access to all internal system resources, so it can fully utilize all potential optimization opportunities. However, it requires high realization cost to inject the framework layers in the codebase of the underlying system. In addition, managing any changes to the framework layers will also require certain level of expertise and efforts, which is expected to limit the system extensibility and community contributions.

(3) Centrist Approach. A third approach is to realize some of the low-level operations, e.g., filtering, indexing, and keyword extraction, as built-in internal operations and derive and append other operations on-top of these basic operations. This approach will combine half ways of both simplicity and efficiency of the previous two approaches. Although this approach may sound ideal, it will require a careful selection for the underlying data management system as its capabilities will highly affect the realization simplicity and the performance gains. In addition, it requires a thoughtful and non-straightforward design of the built-in and on-top operations to cover a wide variety of use cases while still maintains the framework extensibility.

5 CONCLUSION

This paper has envisioned a unified framework to support a wide variety of event detection techniques in existing data management systems. The main goal is significantly improving the impact of the existing rich literature of event detection techniques through easing building such applications for end users. To this end, the paper has proposed a set of abstractions that are organized in five main layers. These abstractions work as building blocks for event detection techniques, so an entire event detection pipeline can be defined through specifying the configuration details of such layers. These layered abstractions are envisioned to be incorporated with existing data management systems, providing a variety of configuration options through a declarative language interface. With realizing such a layered approach, the cost and efforts for building a new event detection application will be dramatically reduced, which will broaden the impact of such critical applications in different domains. The paper also discussed three potential realization pathways to couple the proposed framework with the existing data management systems highlighting the pros and cons of each pathway.

REFERENCES

[1] H. Abdelhaq, C. Sengstock, and M. Gertz. Eventweet: Online localized event detection from twitter. *Proceedings of the International Conference on Very Large Data Bases, VLDB*, 6(12):1326–1329, 2013.

[2] J. Allan, R. Papka, and V. Lavrenko. On-line new event detection and tracking. In *SIGIR*, pages 37–45, 1998.

[3] N. Alsaedi, P. Burnap, and O. F. Rana. Can we predict a riot? disruptive event detection using twitter. *ACM Transactions on Internet Technology*, 17(2):18:1–18:26, 2017.

[4] N. Avudaiappan, A. Herzog, S. Kadam, Y. Du, J. Thatche, and I. Safro. Detecting and Summarizing Emergent Events in Microblogs and Social Media Streams by Dynamic Centralities. In *IEEE Big Data*, 2017.

[5] E. Benson, A. Haghighi, and R. Barzilay. Event discovery in social media feeds. In *ACL: Human Language Technologies*, pages 389–398, 2011.

[6] blog.twitter.com. How the usgs uses twitter data to track earthquakes, 2019.

[7] L. Chen and A. Roy. Event detection from flickr data through wavelet-based spatial analysis. In *CIKM*, pages 523–532, 2009.

[8] A. Cui, M. Zhang, Y. Liu, S. Ma, and K. Zhang. Discover Breaking Events with Popular Hashtags in Twitter. In *CIKM*, 2012.

[9] N. D. Doulamis, A. D. Doulamis, P. C. Kokkinos, and E. M. Varvarigos. Event Detection in Twitter Microblogging. *IEEE Transactions Cybernetics*, 46(12):2810–2824, 2016.

[10] P. Earle, M. Guy, R. Buckmaster, C. Ostrum, S. Horvath, and A. Vaughan. Omg earthquake! can twitter improve earthquake response? *Seismological Research Letters*, 81(2):246–251, 2010.

[11] P. S. Earle, D. C. Bowden, and M. Guy. Twitter earthquake detection: earthquake monitoring in a social world. *Annals of Geophysics*, 54(6), 2012.

[12] D. A. et. al. The beckman report on database research. *Communications of the ACM*, 59(2):92–99, Jan. 2016.

[13] A. Farzindar and W. Khreich. A survey of techniques for event detection in twitter. *Computational Intelligence*, 31(1):132–164, 2015.

[14] W. Feng, C. Zhang, W. Zhang, J. Han, J. Wang, C. Aggarwal, and J. Huang. STREAMCUBE: Hierarchical Spatio-temporal Hashtag Clustering for Event Exploration Over the Twitter Stream. In *ICDE*, 2015.

[15] J. Kalyanam, S. Velupillai, M. Conway, and G. Lanckriet. From Event Detection to Storytelling on Microblogs. In *ASONAM*, 2016.

[16] J. Krumm and E. Horvitz. Eyewitness: identifying local events via space-time signals in twitter feeds. In *SIGSPATIAL*, pages 20:1–20:10, 2015.

[17] R. Lee and K. Sumiya. Measuring geographical regularities of crowd behaviors for twitter-based geo-social event detection. In *Proceedings of the International Workshop on Location Based Social Networks*, pages 1–10, 2010.

[18] Q. Li, A. Nourbakhsh, S. Shah, and X. Liu. Real-time novel event detection from social media. In *ICDE*, pages 1129–1139, 2017.

[19] R. Li, K. H. Lei, R. Khadiwala, and K. C. Chang. TEDAS: A twitter-based event detection and analysis system. In *ICDE*, pages 1273–1276, 2012.

[20] X. Liu, Q. Li, A. Nourbakhsh, R. Fang, M. Thomas, K. Anderson, R. Kociuba, M. Vedder, S. Pomerville, R. Wudali, R. Martin, J. Duprey, A. Vachher, W. Keenan, and S. Shah. Reuters tracer: A large scale system of detecting & verifying real-time news events from twitter. In *CIKM*, pages 207–216, 2016.

[21] X. Liu, A. Nourbakhsh, Q. Li, S. Shah, R. Martin, and J. Duprey. Reuters tracer: Toward automated news production using large scale social media data. In *IEEE BigData*, pages 1483–1493, 2017.

[22] nowtrending.hhs.gov. Following disease trends, 140 characters at a time, Apr. 2019.

[23] A. Ritter, O. Etzioni, S. Clark, et al. Open Domain Event Extraction from Twitter. In *SIGKDD*, 2012.

[24] K. Rudra, P. Goyal, N. Ganguly, P. Mitra, and M. Imran. Identifying Sub-events and Summarizing Disaster-Related Information from Microblogs. In *SIGIR*, 2018.

[25] A. M. Sainju and Z. Jiang. Grid-based colocation mining algorithms on GPU for big spatial event data: A summary of results. In *SSTD*, 2017.

[26] T. Sakaki, M. Okazaki, and Y. Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In *WWW*, pages 851–860, 2010.

[27] statista.com. Mobile internet - statistics & facts, 2019.

[28] statista.com. Mobile internet user penetration rate in selected countries as of 3rd quarter 2017, 2019.

[29] K. Watanabe, M. Ochi, M. Okabe, and R. Onai. Jasmine: a real-time local-event detection system based on geolocation information propagated to microblogs. In *CIKM*, pages 2541–2544, 2011.

[30] H. Wei, H. Zhou, J. Sankaranarayanan, S. Sengupta, and H. Samet. Detecting latest local events from geotagged tweet streams. In *SIGSPATIAL*, pages 520–523, 2018.

[31] A. Weiler, M. Grossniklaus, and M. H. Scholl. Editorial: Survey and experimental analysis of event detection techniques for twitter. *The Computer Journal*, 60(3):329–346, 2017.

[32] Y. Yang, T. Pierce, and J. G. Carbonell. A study of retrospective and on-line event detection. In *SIGIR*, pages 28–36, 1998.

[33] C. Zhang, D. Lei, Q. Yuan, H. Zhuang, L. Kaplan, S. Wang, and J. Han. GeoBurst+: Effective and Real-Time Local Event Detection in Geo-Tagged Tweet Streams. *ACM TIST*, 9(3):34, 2018.

[34] C. Zhang, L. Liu, D. Lei, Q. Yuan, H. Zhuang, T. Hanratty, and J. Han. Triovecevent: Embedding-based online local event detection in geo-tagged tweet streams. In *KDD*, pages 595–604, 2017.

[35] C. Zhang, G. Zhou, Q. Yuan, H. Zhuang, Y. Zheng, L. Kaplan, S. Wang, and J. Han. Geoburst: Real-time Local Event Detection in Geo-tagged Tweet Streams. In *SIGIR*, 2016.

[36] Y. Zhang, C. Szabo, Q. Z. Sheng, and X. S. Fang. SNAF: Observation Filtering and Location Inference for Event Monitoring on Twitter. *WWW Journal*, 21(2):311–343, 2018.

[37] D. Zhou, L. Chen, and Y. He. An Unsupervised Framework of Exploring Events on Twitter: Filtering, Extraction and Categorization. In *AAAI*, 2015.

[38] D. Zhou, T. Gao, and Y. He. Jointly Event Extraction and Visualization on Twitter via Probabilistic Modelling. In *ACL*, volume 1, 2016.