StoryLine: Unsupervised Geo-event Demultiplexing in Social Spaces without Location Information

Shiguang Wang, Prasanna Giridhar, Hongwei Wang
University of Illinois
Urbana, IL 61801
[swang83,Giridhar2,hwang172]@illinois.edu

Lance Kaplan, Tien Pham
US Army Research Labs
Adelphi, MD 207843
lance.m.kaplan@us.army.mil,tien.pham1.civ@mail.mil

Aylin Yener
Penn State University
University Park, PA 16802
yener@engr.psu.edu

Tarek Abdelzaher
University of Illinois
Urbana, IL 61801
zaher@illinois.edu

ABSTRACT

Some of the most widely deployed IoT devices in urban areas are smartphones in the possession of urban individuals. Their proliferation has led to the emergence of crowdsensing/crowdsourcing services, where humans collect data about their environment (using phones), and servers aggregate the data for various application purposes of interest. With the emergence of social media, a common alternative form of human data entry has become media posts (e.g., on Twitter). This leads to the prospect of building crowdsensing services on top of social media content, exploiting humans as "sensors". In this paper, we develop one such service, called *StoryLine*. The service detects and tracks physical urban events of interest to the user, such as car accidents, infrastructure damage (in the aftermath of a natural disaster), or instances of civil unrest. It offers an interface to client-side software that allows browsing such events in real time, as well as an interface for software applications to a structured representation of the events and their related statistics. The service embodies novel algorithms for real-time detection, demultiplexing, and tracking of physical events using social media data. In our evaluation with Twitter feeds, we show that our service outperforms two state-of-the-art baselines in event detection and demultiplexing. We also conduct two case-studies to show the effectiveness of the real-time event detection capability and event tracking performance of our system.

CCS CONCEPTS

•**Computer systems organization** →*Embedded and cyber-physical systems;* Sensor networks;

KEYWORDS

Social Sensing, Event Tracking

ACM Reference format:
Shiguang Wang, Prasanna Giridhar, Hongwei Wang, Lance Kaplan, Tien Pham, Aylin Yener, and Tarek Abdelzaher. 2017. StoryLine: Unsupervised Geo-event Demultiplexing in Social Spaces without Location Information. In *Proceedings of The 2nd ACM/IEEE International Conference on Internet-of-Things Design and Implementation, Pittsburgh, PA USA, April 2017 (IoTDI'17),* 11 pages.
DOI: http://dx.doi.org/10.1145/3054977.3054992

1 INTRODUCTION

The proliferation of smart phones in urban spaces makes them some of the most commonly deployed devices in the emerging era of IoT. A common use of smart phones has been in data collection by exploiting phone sensors for various city-wide measurement tasks. This use is often termed *crowd-sensing*. While sensors offer a great opportunity for data collection on smart phones, a common alternative form of data entry is by the user via social media applications. This observation leads to the idea of building data collection/crowd-sensing services on top of real-time social media content. We shall henceforth call the idea of estimating physical state from social media posts, *social sensing* [32].

StoryLine is a novel social sensing (back-end) service that exploits real-time content posted on social media to detect, demultiplex, and track instances of physical events of interest to the user. The user may specify the category of events of interest, such as car accidents, road closures, concerts, or urban protests. The current version of the tool uses Twitter. It is intended to complement services that collect data from physical sensors. We leverage the intuition that Twitter posts (and, by implication, possibly similar microblogging media) can be exploited as a novel *sensing modality*, not unlike acoustic sensing, vibration sensing, or magnetic sensing. The analogy is straightforward as illustrated in Figure 1. Much the way physical objects induce distinguishable signals in their physical environment that can be detected by observing the physical medium, socially-relevant events (such as car accidents, attacks, natural disasters, parades, or protests) induce distinguishable signals in their social environment that can be detected by observing the social medium. The paper develops an IoT service that exploits this social modality of sensing, motivated by the proliferation of users who post in real-time to describe their surrounding world. The service offers a client-side interface and a programmers interface to browse and

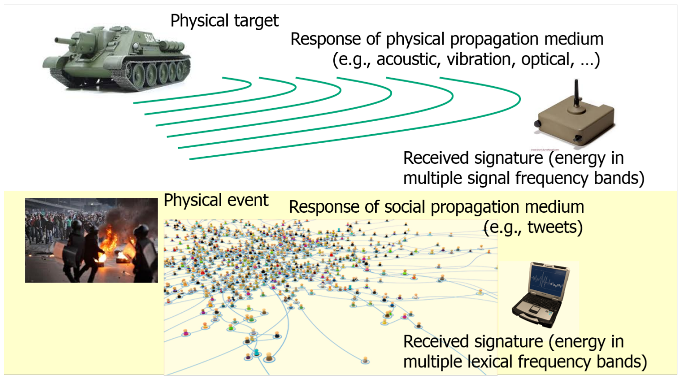
Physical target
Response of physical propagation medium (e.g., acoustic, vibration, optical, …)
Received signature (energy in multiple signal frequency bands)
Physical event
Response of social propagation medium (e.g., tweets)
Received signature (energy in multiple lexical frequency bands)

Figure 1: The Social Sensing Modality and its Analogy with Physical Sensing

retrieve detected events, receive alerts when certain events occur, and compute historical statistics.

StoryLine makes a fundamental contribution to event detection literature on Twitter; namely, to the authors' knowledge, it is the first paper that distinguishes between concurrent *instances* of a user-specified category of events (which we call event *demultiplexing*) in a manner that (i) does not need location information and (ii) is entirely unsupervised (i.e., does not need prior training or remote supervision techniques). None of the prior work offers event demultiplexing that has both of the above properties.

Demultiplexing is essential to our IoT service, where a city planner, for example, might want to know statistics of events occurrence over time, which implies knowing how many events (say, car accidents) occured. Not relying on location metadata means we can identify more events, since more than 98% of tweets are not geotagged [29, 35]. Not using language features and related training means the service can be deployed internationally at little or no additional cost, regardless of local language. It will demultiplex events described in most languages[1] (although will not be able to merge descriptions of the same event across different languages).

The idea that social media posts collectively constitute a form of sensing is not new. It dates back to the beginning of the decade. In their pioneering work, Sakaki et al. [27] proposed an algorithm to detect and track natural disasters, such as earthquakes and hurricanes. The work exploited the spatio-temporal footprint of media posts to detect and localize events. Since then, a large volume of literature on event detection was published. Surveys of these techniques recently appeared both for Twitter-based detection specifically [6], and for detection from social media in general [14].

Work on Twitter-based event detection generally falls into three categories. First, some algorithms do detection but not demultiplexing [3, 16, 21, 26]. Demultiplexing is a somewhat different problem from mere *detection* in that one needs to distinguish one concurrent event instance (e.g., a car accident) from another. An algorithm that does not do demultiplexing can detect, for example that a major traffic accident occurred, and can separate traffic accidents from other types of events, such as floods, but cannot easily differentiate between two concurrent traffic accidents. Many papers in this category do a form of burst detection and text-similarity-based clustering on tweets. Hence, for instance, tweets containing words related to traffic accidents end up in the same cluster (but can include descriptions of multiple accidents).

A second category of work does demultiplexing (separation of concurrent events of the same type) by clustering tweets based on time and location [9, 22, 31]. They often use some notion of coherence (increased frequency of keywords that are *semantically related*) at a given location as an indicator that an event occured at that location [40]. Unfortunately, on Twitter, less than 2% of tweets are geotagged [29, 35], so this approach can easily miss small events. While user account registration information commonly includes location (about 25% of accounts have it), it is course-grained (city-scale only), and hence cannot help distinguish different local events.

Finally, some papers indeed do demultiplexing without location metadata [29, 35]. However, they use natural-language processing or machine learning, and thus are language-specific and/or need prior training. For example, some papers use shallow analysis of text to identify location keywords (e.g., references to specific streets, cities, or landmarks) [13, 18, 29, 35], and cluster tweets based on locations referred to in the text. In contrast, our approach is unsupervised and hence does not require classifier training [19, 27, 37, 41], bootstrapping [4], or significant pre-processing [10, 30].

This paper thus opens up a new category of event detection methods that can *demultiplex* events, *without use of location* information, in an entirely *unsupervised* NLP-free fashion. We demonstrate the effectiveness and efficiency of our algorithm in the evaluation section by comparing with state-of-the-art baselines using four real Twitter feeds.

The rest of this paper is organized as follows. We define our problem more formally in Section 2. We propose our solution to unsupervised event detection, demultiplexing, and tracking in Section 3. The implementation of the resulting service is described in Section 4 and its evaluation is presented in Section 5. We discuss the related work in Section 6 and conclude the paper in Section 7.

2 PROBLEM STATEMENT

The purpose of *StoryLine* is to do for Twitter posts what back-end aggregation/fusion services do for crowd-sourced sensor data with the purpose of detecting and tracking physical events in urban spaces. We envision services like *StoryLine* complementing more traditional sensor data fusion services in IoT applications. Towards that end, *StoryLine* represents the monitored environment as a set of event instances, each given by an instance identifier, a general class label, and an observation summary that accumulates chronologically sorted posts (namely, Twitter messages, called *tweets*) regarding the event instance. While *StoryLine* stores the demultiplexed stream of tweets that describes each event instance, this stream – the story – is not interpreted by the service.

New events may be generated over time and old events are eventually removed. Each event has a finite lifespan during which the event is said to be *ongoing*. For the purposes of this paper, an event instance is broadly defined as an *incident, independently observable*

[1]Since we still rely on white space as word/token separators, it will not work well for languages with no spaces like Chinese.

by multiple humans within limited time and space. The term "independently observable" suggests that retweets be ignored, as they do not constitute independent observations. The term "multiple humans" suggests that a threshold could be used on the rate of reported observations, below which an event is of no interest for the purposes of this paper. Finally, "limited time and space" suggests that an event has a start time, an end time, and a location trajectory. Event locations described by a single point in space constitute a special case of a trajectory. Hence, vehicular traffic accidents, shootings, demonstrations, rallies, funeral processions, insurgent attacks, bombings, and sports events, are different examples that satisfy the definition of events used in this paper.

In this paper, we restrict our attention to the problem of demultiplexing of different *instances* of the same (user-specified) event category, together with related instance detection and instance tracking algorithms. To do so, we look for *co-occurrence surprises*; that is, spikes in keywords that *do not* commonly co-occur. An information gain metric is derived to detect such spikes in an unsupervised fashion. For example, in description of car accidents, a particular car accident involving a drunk driver who ran over a dog on a bridge, might be described by tweets containing such keywords as "drunk" and "dog". These words do not commonly co-occur in the same microblog post. Hence, if such an uncommon combination of words spikes today in the context of tweets about car accidents, it is an indication that a new event instance occurred. We show in the paper that co-occurrence surprise leads to better demultiplexing of event instances than techniques based on finding spikes in semantically related or commonly co-occurring words (e.g., "car accident").

In our problem, *StoryLine* discretizes time into slots, and abstracts the current state of the monitored environment at any discrete time instant, k, by a dynamically evolving set of ongoing event instances $\mathcal{E}(k)$, where an event instance E_i has a detection (or start) time, S_i, and a finish time, F_i. We say that $E_i \in \mathcal{E}(k)$ for $S_i \leq k \leq F_i$. Each event instance is further associated with a chronologically sorted list of all timestamped tweets that describe it up to the current time, called its cumulative observation summary, $Summary_i$ [k].

The social medium is said to emit a signal. The signal emitted in slot k (i.e., the slot ending at time instant, k) is the body of text emitted on the social medium in slot k. In the case of Twitter, this would be the set of tweets time-stamped in slot k. Our service uses the Twitter programming API to collect tweets in real time as they are emitted. The signal emitted on the social medium in slot k is denoted $Signal(k)$. Given the stream, $Signal(k)$, the problem addressed in this paper is to determine for each time slot, k, (i) the set of ongoing event instances, $\mathcal{E}(k)$, and (ii) the observation summary, $Summary_i[k]$, for each event instance, $E_i \in \mathcal{E}(k)$.

3 THE DESIGN OF STORYLINE

In this section, we present informal intuitions, followed by descriptions of our unsupervised detection, demultiplexing and tracking algorithms. To use *StoryLine*, the user issues a *StoryLine* query such as "traffic" and "accident".[2] This query is like a subscription to a newsfeed that filters content specific to the query terms. A process is started that repeatedly uses Twitter API to obtain the latest tweets (subject to Twitter rate limits) that contain the specified keywords (i.e., match the filter). The resulting real-time stream of arriving tweets is then demultiplexed to separate descriptions of different events (e.g., different accidents), which is the focus of the discussion below. The process continues indefinitely until terminated by the user. At any given time, multiple such queries may be ongoing, depending on the categories of events that the user is interested in following. In principle, other work in current literature can be used to help the user select appropriate keywords for each query to better filter the desired event category. A substantial amount of work, for example, exists on *topic modeling* [39] that can be leveraged for help with topic-specific queries. This help is outside the scope of our paper. In this paper, we start at the point where a query has been formulated and a stream of tweets matching the query filter has started arriving, and needs to be demultiplexed.

3.1 Design Intuitions

Perhaps the most important contribution of our demultiplexing approach is its *simplicity*. It is indeed based on a very simple intuition. The intuition underlying the approach lies in a *sparsity argument*; specifically, we find the simplest sparse feature space in which (by virtue of sparsity) event instances are sufficiently separated. To illustrate what this means, consider the lexicon of commonly used words in a language, such as English. Such a domain may contain around 10,000 words. We may want to distinguish 1000s of concurrent event instances, each described by multiple characteristic words. In this case, the set of event instances populate the space of words rather densely. (That is to say, there may be partial overlap between sets of words commonly used in describing *different* event instances.) The same is not true, however, of word pairs (i.e., the "second power" or Cartesian product of the lexical domain). In a language of 10,000 words, there are 100 million possible word pairs. This is several orders of magnitude larger than the number of event instances we might need to demultiplex within any given time slot. Hence, within a given time slot, the set of word pairs that characterize ongoing event instances populate *very sparsely* the feature space of all possible word pairs. The probability of overlap (i.e., different event instances being characterized by the same word pair) is negligible.[3] Two caveats must be understood regarding our sparsity observation.

First, the validity of the sparsity observation in the feature space of keyword pairs hinges on the lack of strong correlations between keywords used in the chosen pairs. The probability of seeing two words, W_1 and W_2, on the medium is $P(W_1, W_2) = P(W_1)P(W_2|W_1)$. If these words often come together as a single term, such as "Dodgers Stadium" or "Angela Merkel", the probability $P(W_2|W_1)$ may be close to 1 and thus, $P(W_1, W_2) \approx P(W_1)$. In other words, the term should be considered as a single keyword. Hence, we remove from consideration keywords pairs, where the individual keywords co-occur with a much higher probability than the product of the probabilities of their occurrence individually.

[2]The query terms are presumably expressed in the user's language and hence are language-specific. The point we made earlier, however, is that none of our processing mechanisms use any language assumptions. Hence, they work regardless of the language in which the user expresses the query.

[3]In a prior literature [12], an empirical study was conducted analyzing tweets about car accidents in three major California cities. The study indeed showed that 2-keyword signatures tend to uniquely distinguish different car accidents. The above general argument presents a signal-sparsity justification of this phenomenon.

With that simple filtering, we ensure lack of strong correlations between keywords in a pair.

Second, sparsity ensures that if two event instances are different, their discriminative keyword pairs are different with high probability. The inverse is not always true. Given two different discriminative keyword pairs, they may or may not be of two different event instances. This will be the case, for example, if the event instance has more than two high frequency keywords, allowing for multiple alternative subsets of two keywords to uniquely characterize the event. Such subsets would have to be consolidated.

As tweets arrive, new spikes in keyword pairs are detected and "bins" are associated with spiking pairs, called *discriminative* pairs. Thereafter, subsequent tweets are inspected for discriminative keyword pairs they contain and placed into the corresponding bins. The words in the pair may apprear in any order within the tweet and need not be contiguous. A tweet may be placed in multiple bins if it contains multiple discriminative keyword pairs. Note that, identifying discriminative keyword pairs is *not* a quadratic problem in the number of words or tweets in a time slot. This is because the only candidate pairs are those that occur together somewhere in a tweet. Hence, the problem is quadratic in the number of words in a tweet, but *linear* in the number of tweets in a timeslot. Since tweets are of short bounded size, the former component can be bounded by a manageable constant. Accorrdingly, computationally efficient solutions (linear in the number of tweets) are possible. Importantly, *no prior training is needed.*

Two questions remain. First, how are discriminative keyword pairs selected? Second, how to consolidate bins pertaining to the same event instance? (The latter is needed because an event instance may give rise to multiple discriminative keyword pairs.) These questions are addressed below.

3.2 Discriminative Keyword Pair Selection

Information gain is a common measure for detecting discriminative features that we leverage here. When a new event occurs, keyword pairs characteristic to that event will be present disproportionately in the current window compared to the previous one. We thus compute information gain of a keyword pair in a window as the amount of information gained in distinguishing this window from previous windows if we were told whether or not the given keyword pair occurred in that window. Clearly, pairs that occur more disproportionately in the current window offer more information gain. These are pairs of words that *do not normally co-occur*. Hence, information gain is a measure of co-occurrence surprise.

Let X_j denote the event whether a tweet contains the keyword pair s_j, where $X_j = 1$ means it contains s_j and $X_j = 0$ denotes it does not. For simplicity, we omit the script j when it is clear from the context. Let Y_k denote the event whether a tweet is posted in the current time slot k, where $Y_k = 1$ means it is posted in the current time slot, and $Y_k = 0$ means it is posted in the previous time slot $k-1$. Again, we omit the script k for simplicity. The tuple (X, Y) thus denotes whether a tweet contains the keyword pair s_j, and whether it is posted in the current time slot k. It can have four distinct values $(0, 0)$, $(0, 1)$, $(1, 0)$, $(1, 1)$ that have the straighforward physical meaning respectively.

$H(W)$ dentoes the entropy of the variable W and is defined as:

$$H(W) = -\sum_{w \in \mathcal{W}} p(w) \log p(w),$$

where $\mathcal{W}$ is the value set of variable W.

More specifically, let there be w_k distinct tweets emitted in window k, and w_{k-1} distinct tweets emitted in window $k-1$. Hence, the probability of a tweet (taken at random from the tweets in either window) to be present in the current window, k, is $p(k) = w_k/(w_k + w_{k-1})$. Similarly, the probability of a tweet (taken at random from the tweets in either window) to be present in the previous window, $k-1$, is $p(k-1) = w_{k-1}/(w_k + w_{k-1})$.

Let some keyword pair, s_j, be present in w_k^j distinct tweets in window k, and w_{k-1}^j distinct tweets in window $k-1$. Hence, the probability of a tweet that contains the pair s_j (taken at random from those containing that pair in either window) to be from the current window, k, is $p_j(k) = w_k^j/(w_k^j + w_{k-1}^j)$. Similarly, the probability of a tweet that contains s_i (taken at random from those containing that pair in either window) to be from the previous window, $k-1$, is $p_j(k-1) = w_{k-1}^j/(w_k^j + w_{k-1}^j)$.

Let the entropy of the variable referring to window identity, Y, be denoted $H(Y)$, where Y is either k or $k-1$. By definition, $H(Y)$ is given by:

$$\begin{aligned} H(Y) &= -p(k) log_2 p(k) - p(k-1) log_2 p(k-1) \\ &= -\frac{w_k}{(w_k + w_{k-1})} log_2 \frac{w_k}{(w_k + w_{k-1})} \\ &\quad -\frac{w_{k-1}}{(w_k + w_{k-1})} log_2 \frac{w_{k-1}}{(w_k + w_{k-1})} \end{aligned} \tag{1}$$

Similarly, the conditional entropy of Y, given that we know whether pair s_j occurred, is denoted $H(Y|s_j)$. By definition, $H(Y|s_j)$ is given by:

$$\begin{aligned} H(Y|s_i) &= -p_i(k) log_2 p_i(k) - p_i(k-1) log_2 p_i(k-1) \\ &= -\frac{w_k^i}{(w_k^i + w_{k-1}^i)} log_2 \frac{w_k^i}{(w_k^i + w_{k-1}^i)} \\ &\quad -\frac{w_{k-1}^i}{(w_k^i + w_{k-1}^i)} log_2 \frac{w_{k-1}^i}{(w_k^i + w_{k-1}^i)} \end{aligned} \tag{2}$$

Finally, the information gain, IG_j, associated with pair s_j, is given by:

$$IG_j = H(Y) - H(Y|s_j) \tag{3}$$

Equation (3) can be used to compute information gain for each keyword pair, s_j, in each time slot k. In computing information gain we do not count retweets, since they do not offer additional first-hand information on events. This helps remove rumors, opinion tweets and slogans that propagate primarily by retweeting, as opposed to descriptions of independently observable events. Only the keyword pairs with information gain greater than a threshold would be selected as discriminative keyword pairs.

The above discussion focused on detection of discriminative keyword pairs; those with high information gain. Remember that high information gain indicates that the words in the pair do not normally co-occur. We show that this insight allows us to find new event instances.

Besides detecting new discriminative pairs in the current window, the system also continues demultiplexing based on discriminative pairs found in previous windows. Those correspond to events detected earlier. Therefore, in each time slot k, we first inherit all discriminative keyword pairs used in the previous slot whose clusters were still growing, (i.e., the cumulative number of tweet containing that pair by time slot $k-1$ is greater than that by slot $k-2$). We then augment that inherited set with new keyword pairs found discriminative in the current window.

3.3 The Consolidation Algorithm

Events may contain more than one discriminative keyword pair. Therefore, it is important to be able to consolidate different bins when their tweets are about the same event. Consider the set of discriminative keyword pairs used in slot k. Each such pair, s_j, is associated with a bin of tweets, C_j, in which the pair occurs. Our approach for consolidating bins referring to the same physical event lies in detecting similarity between their respective data clusters. In our drunk driver example, presented earlier, a cluster of tweets about an accident involving a drunk driver killing a dog on a bridge might be distinguished by discriminative keyword pairs ("drunk", "dog"), ("drunk", "bridge") and ("bridge", "dog"). Each pair might end-up associated with a bin that contains largely the same tweets. A distance metric can thus be defined between content of different bins based on the statistical distribution of words in the bins. The distance between two bins will decide if they are about the same event. Four common distance metrics between statistical distributions of words are compared. Namely, the *Jaccard Distance*, the *Term Frequency Difference Ratio*, the *Cosine Similarity Distance*, and the *KL Divergence*. For the detailed definitions of the distance functions, please refer to the appendix.

We observed that Jaccard distance performs consistently the best among the four, and is also the simplest metric since it is the only one that does not consider word frequency (this empirical comparison is shown in the evaluation section). Interestingly, the metric that depends most heavily on the distribution of words, the KL divergence metric, performed the worst. The reason, we believe, is that the tweet clusters (the individual bins) are small enough that it is inaccurate to estimate the true probability of each keyword solely based on its frequency of occurrence in a bin. Hence, the more we depend on having to know a true probability distribution, the less accurate is the resulting consolidation.

Another benefit of applying the Jaccard distance is that the resulting consolidation threshold was found to be largely insensitive to the different types of events, due to its simplicity and discreteness. Other lexical distance functions do not have this property. Hence, in our system, we use the Jaccard distance for bin consolidation and pre-configure the threshold as a *static parameter*. New installations of the system need no further "training". The result of consolidation in slot k is the set of event instances, $\mathcal{E}(k)$, where each event $E_i \in \mathcal{E}(k)$ is associated with a set of tweets.

3.4 Event Tracking

Event tracking extends the consolidation algorithm in a straightforward manner by applying bin consolidation across successive time slots. That is, after consolidating the bins in the current time slot k, we consolidate the bins between the time slot k and $k-1$. One challenge in event tracking is that the event signature, defined by the corresponding consolidated keywords, might evolve due to the evolution of the event and thus the way people describe it.

To catch that change, we use an *overlapping sliding window*. It smoothes out the changes in the lexical frequency distribution of fast developing events over time, as illustrated in Figure 2. With overlapped windows, some part of the event signature remains the same across the two slots. (Note that, the compared slots are overlapping here as in Figure 2.) Therefore, by selecting a proper overlap, we can track the event smoothly and be able to consolidate relevant clusters properly, even as its signature changes gradually over time.

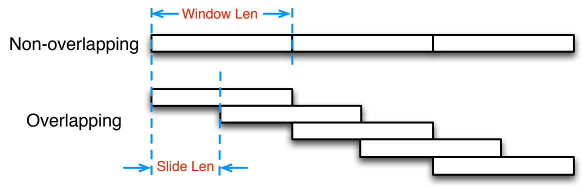

Figure 2: Illustration of the non-overlapping sliding window and the overlapping sliding window (with 50% overlapping).

4 SYSTEM IMPLEMENTATION

In this section, we present the architecture of our social event tracking system as shown in Figure 3. The targeted social medium of our system is Twitter [1]. The system is implemented in Python27 and integrated into an existing social sensing tool, Apollo [4], developed by a subset of the authors. StoryLine provides four interfaces, CREATE, PULL, KILL, and STATS. CREATE enables the user to start an event-tracking task, and PULL enables the user to get the real-time event tracking results. The key parameters of CREATE are (i) a list of keywords for crawling tweets, for example `[protests, confrontation]`, and (3) a user-customized window length (with default value of 24 hours). After the user creates a tracking task, a task ID is returned, which is used in PULL to get the real-time tracking results and in KILL to terminate the existing tracking task. Finally, STATS allows retrieval of a set of statistics about the event type, such as the frequency of occurrence of event instances over time.

Once the tracking task is created, the crawling parameters are passed to the crawler that uses the Twitter API to crawl tweets that satisfy the conditions defined in the parameters in real time. For the tweets returned, we first filter out the redundant tweets, such as the retweets, and then the filtered tweets are fed to our event detection module, where the event signature detection and consolidation are performed. The text clusters are then passed to the event-tracking module. When the user calls the PULL function with the task ID,

[4]http://apollo3.cs.illinois.edu

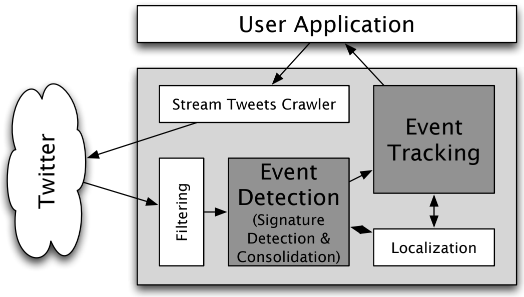

Figure 3: Event Tracking System Architecture

the most recent tracking results are returned encoded using the JSON format. An optional localization module is included (to pin the events on a map, for example, by Giridhar et al. [13]), but it is not relevant to this paper. Please note that unless the user calls KILL, the tracking task keeps working.

5 EVALUATION

In this section, we report the experience of using our tool on event detection and tracking on four datasets crawled from Twitter. We first describe the statistical details of the four datasets, and then discuss the performance of our event signature consolidation for the selected Jaccard distance metric. Next, we study the performance of event detection compared with the state-of-the-art baselines. Finally, we conduct two case studies of Earthquake events and show the real-time event detection capability and event tracking performance of our proposed StoryLine system.

5.1 Twitter Datasets

For repeatability, we collected four data sets from Twitter using the API described in the previous section. These were then replayed as the feeds used in the subsequent experiments to enable fair comparisons across multiple algorithms and conditions. We summarize data collected by the four tasks we created, labeled by (i) *Disaster*, (ii) *Protest*, (iii) *Traffic*, and (iv) *Armed Conflict* below.

- Disaster The dataset is collected with keywords "disaster", "humanitarian", "earthquake". In this dataset, 1, 800, 952 tweets were collected after filtered out retweets, and the time span is from Apr. 19th 19:41:08 UTC, 2015 to Feb. 03rd 06:07:15 UTC, 2016.
- Protest The dataset is collected with keywords "protest", "confrontation". In this dataset, 1, 211, 920 tweets were collected after filtered out retweets, and the time span is from Oct. 16th 05:41:02 UTC, 2015 to Feb. 01st 11:15:43 UTC, 2016.
- Chicago Traffic The keywords used here include "traffic", "accident", "chicago". And all tweets in the Chicago area were also collected in this dataset. In this dataset, 8, 013, 649 tweets were collected after filtered out retweets, and the time span is from May. 15th 13:58:09 UTC, 2015 to Feb. 19th 17:33:43 UTC, 2016.
- Armed Conflict The keywords used here include "rebels", "attack", "bombing". In this dataset, 2, 739, 363 tweets were collected after filtered out retweets, and the time span is from Oct. 16th 05:52:28 UTC, 2015 to Mar. 07th 02:27:03 UTC, 2016.

In the evaluation, each dataset is fed into our StoryLine system in real-time (i.e., we discretize the time into slots and in each slot the tool only considers the current data or that of the past slots but never in the future). Here, each time slot (i.e. window) spans 6 hours, and slides 1 hours in each step.

5.2 Event Signature Consolidation

We test the performance of event signature consolidation based on each of the four lexical frequency domain distance functions introduced earlier, namely *Jaccard distance (Jaccard)*, *Term Frequency Difference Ratio (Tfreq)*, *Cosine Distance (Cosine)*, and *KL Divergence (KL)*. The consolidation error rate is defined as the ratio between the number of incorrectly grouped 2-keyword signature pairs to the total number of signature pairs. Note that, a 2-keyword signature pair is said to be incorrectly grouped if two signatures corresponding to the same event are put into different groups or if two signatures corresponding to different events are put into the same group. Ground truth labeling is done manually.

Figure 4 shows the results, from which we observe that the Jaccard distance function consistently performs the best for all the four datasets, which corroborates our selection of Jaccard distance as the lexical frequency domain distance in Section 3.3. The error

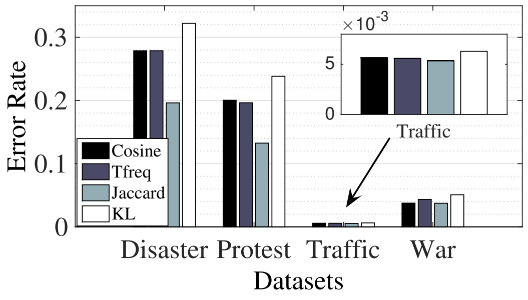

Figure 4: The consolidation error rate.

rate of signature consolidation for the Traffic dataset is the smallest among the four datasets. This is because traffic accidents have a relatively small social media footprint. Often a single 2-keyword signature is associated with the traffic event, therefore only a very small amount of consolidation occurs for this specific event class. We expect that urban events of interest to IoT applications will mostly have small footprints. Examples may be urban fires, shootings, traffic accidents, or road closures. It is therefore encouraging to see that the algorithm is better at detecting and demultiplexing such small-footprint events. The error rate of consolidation of Jaccard for the war dataset is less than 4%. For the protest dataset and the disaster dataset, the error rates are 14% and 20%, respectively.

5.3 Event Demultiplexing

In this subsection, we first eliminate geotagging-based demultiplexing techniques based on recall. We then include in the comparison those techniques that do not need location information, illustrating an advantage in precision and purity of demultiplexing (i.e., correct separation of instances).

Table 1 shows the percentage of tweets in our data sets that are geo-tagged. We also cluster the tweets into events and show the number of event clusters that carry zero, one, or more geo-tagged tweets. We consider fine-grained events here. For example, a war event might refer to a cluster of tweets discussing a single explosion. The table clearly shows that dependence on location information can render most of the events invisible, as they contain no geotagged tweets.

Next, we study the precision of event detection and demultiplexing in our StoryLine system. We compare our StoryLine with the following baselines:

(1) *ET* [26]: In this work, an event is detected using common bi-grams, where the bi-grams are selected from among adjacent pairs of tokens, which is an example of techniques that do not demultiplex well. The reason is that in looking for adjacent bi-grams that have a high chance of co-occurence (for example, "traffic alert" or "crime scene") one often ends up with bi-grams characteristic of a whole *category* of events. In contrast, in our solution, we look for unusual (i.e., rarely co-occurring) pairs of keywords. Results will confirm that those are more characteristic of an event instance.

(2) *TopicModel* [20]: This work proposes an online variation of LDA (Latent Dirichlet Allocation) [8], a famous topic modelling technique. Events are defined and detected by a topic model. This work is a representative event detection solution based on training a text coherence metric (around a topic).

(3) *GeoTag*: In this baseline, we only consider the geo-tagged tweets, and cluster them by physical Euclidean distance. If two tweets are posted within 30 miles, then we cluster them together. A limit is imposed on cluster size to prevent formation of geographically diffuse clusters. This baseline is an example of demultiplexing approaches based on location information.

We randomly selected one week data from our dataset, and compare the precision of event detection/demultiplexing. Here, precision is defined by the ratio between the number of true events output by the algorithm and the total number of events output by the algorithm. Note that, some of the text that the algorithm bins as a separate event might in fact be a false positive. For example, tweets such as "Can you recommend anyone for this #job?" or "these rumors about louis coming to chicago are making me stressed" do not constitute legitimate (geo-)events as defined in this paper.

Table 2 summarizes the precision results of all the algorithms. From this table, we can observe that our algorithm has the highest average performance rank of 1.25 (i.e. it ranks first in Traffic, Disaster, and Armed Conflict datasets and second in Protest dataset), whereas ET has average performance rank of 2.5, TopicModel has 2.75 and GeoTag has only 3.5. In the Protest dataset, most of the events are related to some protests. The number of tweets increases greatly when the protest starts, and at the same time, the total number of tweets also increases. Therefore, the increase of the percentage of the event related tweets and the total tweets is not that significant, thus some true events were not detected by our information gain based approach. But some noisy events were not affected, thus the precision of our algorithm is not the best. ET is based on the absolute increase of the number of event related tweets, therefore, it beats our algorithm. We also notice that geo-tagging does not perform well. We therefore drop it from further comparison.

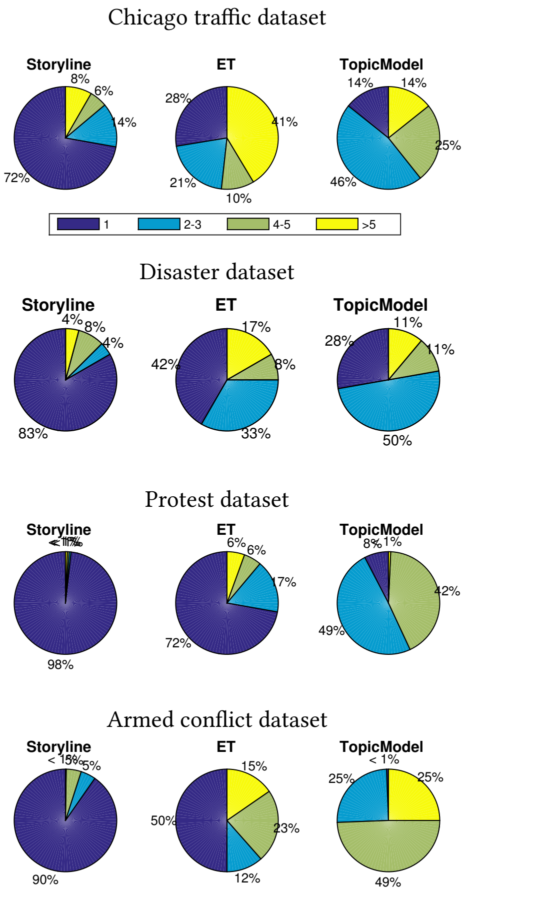

Figure 5: The purity pie charts.

Figure 5 shows the results of purity comparisons for the remaining algorithms, for all the datasets. Purity is a measure of demultiplexing quality into different event instances. Sometimes, the algorithm will output one event that might contain multiple instances. For example, three instances of traffic accidents were output by the TopicModel algorithm: (1) "I 70 now reporting 2 INJURY ACCIDENTS near OH 37", (2) "When things go BOOM on the US 60 @ArizonaDOT #12News", and (3) "@WKYTTraffic tracking an ongoing closure along I-75 near the TN stateline." The purity is defined by a vector, that is the percentage of output events that contain only (1) one event instance, (2) two to three instances, (3) four to five instances and (4) greater than five instances. Ground truth is labelled manually by two different people and conflicts are resolved by a third one.

Table 1: Prevalnce of Geotags in Tweets and Events

Metric	Traffic	Disaster	Protest	Armed Conflict
Total tweets	8013649	1800952	1211920	2739363
Geotagged tweets	726663 (9%)	6068 (0.33%)	2323 (0.19%)	6381 (0.23%)
Events with no Geotagged tweet	90.7%	99.4%	99.6%	99.6%
Events with 1 Geotagged tweet	3.9%	0.5%	0.4%	0.4%
Events with multiple Geotagged tweets	5.4%	0.05%	0	0

Table 2: Event Detection Precision Comparison

Algorithm	Traffic	Disaster	Protest	Armed Conflict
StoryLine	**72.55%**	**76.92%**	80.95%	**88.24%**
ET	57.14%	36.36%	**86.36%**	61.90%
TopicModel	55.10%	60.87%	65.22%	69.57%
GeoTag	66.67%	23.33%	47.37%	41.38%

From the pie charts, we clearly observe that our algorithm has the highest percentage of output events that only contain one instance, which shows that our algorithm does better at *demultiplexing* event instances compared with the baselines.

5.4 Case Study – Real-time Earthquake Detection

In this subsection, we conduct a case study to evaluate the delay in event detection. Here, we select Earthquake events because it is easy to find out the exact (ground-truth) time at which they occurred.

Table 3: Real-time Earthquake Detection Summary

Earthquake Location	Happened Time	Detection Time	Delay
Midoro, Philippines	10/19/2015 13:50	10/19/2015 18:26	4:16
Vanuatu	10/20/15 21:52	10/21/2015 02:40	4:48
Afghanistan	10/26/15 09:09	10/26/15 10:17	1:08
Molucca islands	01/11/16 16:38	01/11/16, 20:41	4:03
Afghanistan	01/12/16 20:05	01/12/16 22:59	2:54
Alberta, Canada	01/12/16 17:30	01/12/16 22:59	5:29
Urakawa, Japan	01/14/16 03:30	01/14/16 04:09	0:39
Alaska	01/24/16 10:30	01/24/16 11:37	1:07
Morocco	01/25/16 04:22	01/25/16 10:44	6:22
Taiwan	02/06/16 19:57	02/06/16 21:39	1:42
Fiji	02/06/16 01:39	02/06/16 02:41	1:02
Indonesia	02/12/16 10:02	02/12/16 13:28	3:26
Oklahoma	02/13/16 17:07	02/13/16 22:37	5:30
NewZealand	02/14/16 00:13	02/14/16 04:38	4:25
Wasco, CA	02/24/16 00:02	02/24/16 00:37	0:35
Antarctica	02/23/16 18:08	02/24/16 00:37	6:29
Cebu, Phillipine	03/01/16 14:52	03/01/16 17:14	2:22
Sumatra, Indonesia	03/02/16 12:49	03/02/16 14:19	1:30

Table 3 shows a summary of the ground-truth occurrence time and detection time (in UTC) of recent earthquake event instances. From the table, we observe that for most of the instances, our algorithm can detect it within 4 hours. For earthquakes occurring in regions with large numbers of active Twitter users, like Japan and California, we can detect earthquakes within 1 hour. (Note that our window sliding length is just 1 hour, so 1 hour is the smallest delay feasible in this configuration.) The results confirm utility of the system for detection of urban events.

5.5 Case Study – Nepal Earthquake Tracking

Finally, we conduct a case study of the Nepal earthquake to help the readers intuitively understand the performance of the tracking functionality of our StoryLine system. The result is summarized in Table 4. An earthquake happened on April 24th 2015 that resulted on the death of more than $8,000$ people in Nepal. The event was detected due to the rise of tweets with new high-information-gain keyword pairs on the social medium. New keyword pairs were associated with the same event as it evolved. The table shows detected keyword pairs and example tweets from their clusters.

From the table, we observe that in the beginning of the earthquake, media posts focused more on the earthquake itself using keywords such as "earthquake" and "death" in tweets. As the earthquake developed, people switched their attention to relief efforts, using keywords such as "donations" and "humanitarian". Later, the discussion focused on survivors, using keywords such as "survivor" and "hospital". Neither the original occurrence of the event nor any of the above keyword pairs was known to our algorithms in advance. They were detected automatically and associated with the same event based on discussed distance metrics. The example shows the capability of our algorithm to tracking real-world events as they evolve.

6 RELATED WORK

The idea of using social networks as sensor networks was discussed in recent literature [33, 34]. While much work focused on analysis of reliability of crowd-sourced observations, this paper exploits social media (specifically, Twitter) to build an IoT service for event detection, demultiplexing, and tracking.

Event detection in social spaces is an active research topic in information retrieval. Some early work includes Allan et al. [4], in which they proposed an online event detection and tracking algorithm. Their algorithm exploits features based on term frequency (TF) and inverse document frequency (IDF), such that if the feature score for a new term is above a predefined threshold then a new event or topic is found. Some recent literature exploits TF-IDF-like features includes Shamma et al. [28] and Benharus et al. [7]. Shamma et al. [28] proposed a peakiness score to identify words that are salient in some time window that were used to detect new events. Since unigrams may not always be sufficient to describe complex events, Benharus et al. [7] proposed a different normalized frequency metric called the trending score for identifying event related n-grams instead of unigrams. These approaches are good at identifying event categories and topic. However, as shown in the evaluation, they are less efficient at separating individual event instances. Our work is also related to the text stream clustering literature [2]. An example is work utilizing optimizations

Table 4: Nepal Earthquake Tracking Summary

Date	New Detected Keywords	Sample Tweets
04/25/2015	nepal, earthquake, death	Powerful magnitude-7.8 earthquake that rocked Nepal triggered an avalanche on Mount Everest http://t.co/MULEuWhx3Q http://t.co/QeRKg8QgYp
		RT @BBCBreaking: At least 876 killed in Nepal #earthquake; deaths also reported in India, Tibet &amp; Bangladesh http://t.co/3BTo9l1QZ4 http://ž026
04/26/2015	help, nepalearthquake	RT @cnnbrk: At least 2,263 people have died in Nepal from massive #NepalEarthquake and aftershocks, official says. http://t.co/hCyjO7YyS7
		Anyone with information about my son Joseph Patrick please help #NepalEarthquake #Pray4Joe http://t.co/X2Kn7mOtRO http://t.což026
04/27/2015	surges, devastation, drone, thankyoupm, donations	Nepal #earthquake: Death toll surges to 3,218; four aftershocks felt in last 12 hours http://t.co/Njvru9k2kQ
		@cnni: New drone footage shows the extent of devastation from the #NepalEarthquake: http://t.co/7PiPjayQZ1https://t.co/phIGRkYoZQ
		#ThankYouPM for massive rescue and relief operation by India in Nepal after #earthquake
		Nepal Earthquake: Facebook to Match Donations Made for Victims http://t.co/aLooadYNxj Free Submission http://t.co/J90dT2qnXb
04/28/2015	salute2indianforces, koirala, sanjay, humanitarian	Thank you very much Indin Forces for being with us.It means alot.... #Salute2IndianForces
		CNN's Dr. Sanjay Gupta performs surgery on girl in Nepal: CNN's Dr. Sanjay Gupta performed a life-saving... http://t.co/4EtmH28EwC #tcot
		Live: Nepal earthquake kills 4,352, PM Sushil Koirala says death toll could reach 10,000: A high-intensity ear... http://t.co/A68VtR6hWK
04/29/2015	survivor,hours, hospital, field, miracle	Nepal earthquake survivor drank urine while trapped for 82 hours http://t.co/v9DHM5Jhnf #worldnews
		That is amazing, Nepal Army rescued a 4-month kid alive after 22 hours! ::http://t.co/KzJPJeZDCx https://t.co/HvTkvS0Ba0 via @sharethis
		RT @haaretzcom: Nepal earthquake updates / Israeli field hospital opens, to treat 200 people per day http://t.co/PMwRlRT6YO http://t.co/s9i
04/30/2015	pakistan, serves, masala, teenage, lydia	Pakistan serves 'beef masala' to earthquake-hit Nepal via /r/worldnews http://t.co/GoFJO09mJP
		Teenage boy pulled out of rubble alive five days after Nepal earthquake http://t.co/0kiAigYE7M #telegraph #news
		Lydia Ko donating earnings to Nepal relief effort: The 18-year-old Ko, ranked No. 1 in the world, successfully... http://t.co/2nquCITqJa

of k-means algorithms to cluster data streams, as proposed by Ordonez [25] and Zhong [42]. However, theirs need prior knowledge (such as the k), which is not always available in social streams for event detection and de-multiplexing. Our approach, in contrast, depends on detecting *co-occurrence surprise*; that is to say, new frequently co-occurring words in tweets that did not previously co-occur. Moreover, our calculations are conducted based on only two adjacent time windows, which is much more efficient than the TF-IDF approach that needs to consider the whole (or a large portion of) corpus.

Topic modeling is another common approach for event detection [17, 20, 43]. Lau et al. [20] proposed an online variation of Latent Dirichlet Allocation (LDA) [8]. In LDA, each topic is modeled as a multinomial distribution of words in a volcabulary, and each document is modeled as a multinomial distribution of k topics, where k is a predifined parameter denoting the total amount of topics. And these two classes of multinomial distributions have two Dirichlet priors respectively. (Dirichlet prior is chosen due to the fact that it is the conjugate prior of the multinomial distribution.) The idea in Lau et al. [20] is incrementally updating the priors in each time window based on previous calculated parameters, and maintaining the one-to-one correpondence of the topics in the current time window and the last one. If there is a sudden change in the topic word distribution, then a new event is supposed to have occurred, where the distance of the distributions is measured by the Jensen-Shannon divergence. Hu et al. [17] proposed ET-LDA (joint Event and Tweets LDA) that exploits a search engine and aligns tweets with corresponding text of events provided by traditional media. They showed that results are greatly improved. Zhou et al. [43] further expand LDA with time and location of the tweets, and proposed a new graphical model called location-time constrained topic (LTT). In their approach, the tweet content, timestamps and geo-tags are all considered. As with TF-IDF based approaches, the topic modeling based approaches also suffer when multiple event instances occur in parallel. Futhermore, on Twitter (which is our focus), reliance on geotags is not sufficient to distinguish different event instances due to the relative scarcity of geotagged tweets.

Previous work also exploits the features (metadata) of tweets in event detection. Chierichetti et al. [11] proposed an event detection algorithm that is purely based on communication pattern analysis in the tweet stream. In their solution, events are detected based only on tweet and retweet counts, via logistic regression. They also provide a model of communication to explain the rationale behind event detection. Similarly, Aggarwal and Subbian [3] considered the social network topology and proposed a clustering solution for event detection. The rationale behind such techniques is based on distinguishing shared human interests; namely, clustering text with similar retweet/communication patterns will isolate events with shared community interest. However, in such approaches, events that trigger a similar community response (such as different terror attacks in nearby locations, or different assaults on police in nearby towns) cannot be easily demultiplexed.

An entirely different line of event detection and demultiplexing techniques focus on location-based (or more generally, spatio-temporal) features [9, 22, 31]. These approaches use different forms of clustering by location metadata contained in tweets, which is indeed an effective means of separation of event instances if the location metadata is sufficiently fine-grained. Unfortunately, less than 2% of tweets are geotagged [29, 35]. While location of other tweets can be estimated from the registered account location of the source, the account metadata carries only city-level location information, which is not sufficiently fine-grained for demultiplexing events at sub-city scale, such as traffic accidents. An interesting approach in the category of location-based event detection techniques is Geoburst [40]. It floats a circle of a pre-specified radius and computes a measure of coherence of tweets originating within the circle. Coherence measures semantic distances between words used in these tweets. When coherence spikes (indicating shorter distances) an event is said to be detected. The rationale is that event occurrence focuses the discussion around fewer topics related to the event, leading to increased coherence of local tweets. Liang et al. [23] exploit a noise filtering approach for event detection and demultiplexing, where the temporal and spatial frequency of each token is treated as a signal and a band-pass filter is applied to filter out background noise as well as separate different event signals within a given locale. In contrast to the above approaches, ours does not depend on using location metadata.

Finally, like us, some recent papers indeed propose demultiplexing schemes that do not use location metadata [13, 18, 29, 35]. Instead, they use language-specific features to distinguish events. A common example of such processing is *isolation of location keywords within the text* of the tweets [13, 29, 35], then clustering by the extracted location information. To appreciate the disadvantage of these techniques, the reader is invited to extract the location information from each of the sentences in Figure 6. Our point is: an approach that does not depend on having language-specific extraction rules is much easier to port across languages, which is a big advantage when considering an international medium, such as Twitter.

Το γαλλικό πλήρωμα αναγκάστηκε σε προσγείωση στην Αθήνα, στην πορεία τους προς τη Μόσχα

フランスの乗組員は、モスクワへ向かう途中、アテネの緊急着陸を余儀なくされた

واضطر الطاقم الفرنسية إلى الهبوط اضطراريا في أثينا في طريقها إلى موسكو

।फ्रेंच चालक दल मास्को के लिए अपने रास्ते पर एथेंस में एक आपात लैंडिंग करने के लिए मजबूर किया गया

Figure 6: Tweets with Location Information.

Our technique, in fact, often finds location keywords automatically as part of the detected signature keyword pairs. Imortantly, however, it does so based on statistical analysis alone, and not linguistic analysis of data. Unlike other event detection techniques that rely on clustering [3, 16, 21, 26, 28, 36], ours looks for frequent pairs that did not usually co-occur. In contrast, much of the prior work looks for burstiness of keywords that are *semantically related* or *frequently co-occur* is some context, as a way of detecting events that feature the indicated semantics or context. This distinction, as we have shown, makes our solution better at event demultiplexing, which is the main contribution of the paper.

Finally, target detection and tracking with physical sensors have been extensively studied in other communities such as sensor networks [15, 24, 38]. A particularly relevant sensor model is that of binary sensors [5], since it closely corresponds to twitter posts that either indicate an event or not. We hope that such literature will inform event detection and demultiplexing algorithms on Twitter.

7 CONCLUSIONS

In this paper, we presented a novel service for IoT applications that augments physical sensor data aggregation and fusion with social media data processing for purposes of physical event detection and demultiplexing. We argued that the social modality of sensing is not unlike other sensing modalities, such as magnetic, acoustic, or seismic. In each case, a useful practice is to transform the signal received from the environment into an appropriate feature domain, and then perform signal processing on that domain. This paper described an exercise in applying the above approach to Twitter text. A specific contribution was the development of an event demultiplexing algorithm that allows separation of (text pertaining to) different instances of a given user-defined category of urban events (e.g., car accidents) in a manner that (i) is entirely unsupervised and (ii) needs no location information. In turn, this separation allows computing various statistics about the events in question, such as their frequency over time. Evaluation results show that the approach is successful at detecting, demultiplexing, and tracking physical events. The success of the approach is analytically attributed to a sparsity argument that enables one to use a very simple feature space to demultiplex instances of events.

The paper is an example of IoT services that go beyond physical sensing. Indeed, in future applications, such as smart cities, data from physical sensors will be fused with data from social media in order to better understand events in the city. Such physical and social fusion offers interesting directions for future work. The paper is a first step towards the envisioned novel cyber-physical architectures. The authors are in the process of investigating follow-up ideas that jointly exploit combinations of physical sensors and social media.

ACKNOWLEDGMENTS

We sincerely thank Dr. Octav Chipara for shepherding this paper and the anonymous reviewers for their valuable comments that significantly enhanced the final manuscript. Research reported in this paper was sponsored in part by the Army Research Laboratory under Cooperative Agreement W911NF-09-2-0053 and in part by NSF under grants CNS 16-18627 and CNS 13-20209. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory, NSF, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

REFERENCES

[1] Twitter inc. http://www.twitter.com.
[2] C. C. Aggarwal. A survey of stream clustering algorithms., 2013.
[3] C. C. Aggarwal and K. Subbian. Event detection in social streams. In *SDM*, volume 12, pages 624–635. SIAM, 2012.
[4] J. Allan, R. Papka, and V. Lavrenko. On-line new event detection and tracking. In *SIGIR*. ACM, 1998.
[5] J. Aslam, Z. Butler, F. Constantin, V. Crespi, G. Cybenko, and D. Rus. Tracking a moving object with a binary sensor network. In *SenSys*. ACM, 2003.
[6] F. Atefeh and W. Khreich. A survey of techniques for event detection in twitter. *Comput. Intell.*, 31(1):132–164, Feb. 2015.
[7] J. Benhardus and J. Kalita. Streaming trend detection in twitter. *International Journal of Web Based Communities*, 9(1):122–139, 2013.
[8] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.
[9] A. Boettcher and D. Lee. Eventradar: A real-time local event detection scheme using twitter stream. In *Proceedings of the 2012 IEEE International Conference on Green Computing and Communications*, GREENCOM '12, pages 358–367, Washington, DC, USA, 2012. IEEE Computer Society.
[10] M. Brenner and E. Izquierdo. Social event detection and retrieval in collaborative photo collections. In *Proceedings of the 2nd ACM International Conference on Multimedia Retrieval*, page 21. ACM, 2012.
[11] F. Chierichetti, J. M. Kleinberg, R. Kumar, M. Mahdian, and S. Pandey. Event detection via communication pattern analysis. In *ICWSM*, 2014.

[12] P. Giridhar, M. T. Amin, T. Abdelzaher, L. Kaplan, J. George, and R. Ganti. Clarisense: Clarifying sensor anomalies using social network feeds. In *PERCOM Workshops*. IEEE, 2014.

[13] P. Giridhar, S. Wang, T. F. Abdelzaher, J. George, L. Kaplan, and R. Ganti. Joint localization of events and sources in social networks. In *Proceedings of the 2015 International Conference on Distributed Computing in Sensor Systems*, DCOSS '15, pages 179–188, Washington, DC, USA, 2015. IEEE Computer Society.

[14] A. Goswami and A. Kumar. A survey of event detection techniques in online social networks. *Social Network Analysis and Mining*, 6(1):107, 2016.

[15] C. Gui and P. Mohapatra. Power conservation and quality of surveillance in target tracking sensor networks. In *MobiCom*. ACM, 2004.

[16] A. Guille and C. Favre. Mention-anomaly-based event detection and tracking in twitter. In *Advances in Social Networks Analysis and Mining (ASONAM), 2014 IEEE/ACM International Conference on*, pages 375–382. IEEE, 2014.

[17] Y. Hu, A. John, D. D. Seligmann, and F. Wang. What were the tweets about? topical associations between public events and twitter feeds. In *ICWSM*, 2012.

[18] T. Hua, F. Chen, L. Zhao, C.-T. Lu, and N. Ramakrishnan. Automatic targeted-domain spatiotemporal event detection in twitter. *Geoinformatica*, 20(4):765–795, Oct. 2016.

[19] M. Jäger, C. Knoll, F. Hamprecht, et al. Weakly supervised learning of a classifier for unusual event detection. *Image Processing, IEEE Transactions on*, 17(9):1700–1708, 2008.

[20] J. H. Lau, N. Collier, and T. Baldwin. On-line trend analysis with topic models:\# twitter trends detection topic model online. In *COLING*, pages 1519–1534, 2012.

[21] C. Li, A. Sun, and A. Datta. Twevent: segment-based event detection from tweets. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 155–164. ACM, 2012.

[22] R. Li, K. H. Lei, R. Khadiwala, and K. C.-C. Chang. Tedas: A twitter-based event detection and analysis system. In *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering*, ICDE '12, pages 1273–1276, Washington, DC, USA, 2012. IEEE Computer Society.

[23] Y. Liang, J. Caverlee, and C. Cao. A noise-filtering approach for spatio-temporal event detection in social media. In *Advances in Information Retrieval*, pages 233–244. Springer, 2015.

[24] C.-Y. Lin, W.-C. Peng, and Y.-C. Tseng. Efficient in-network moving object tracking in wireless sensor networks. *IEEE TMC*, 5(8):1044–1056, 2006.

[25] C. Ordonez. Clustering binary data streams with k-means. In *ACM SIGMOD workshop*. ACM, 2003.

[26] R. Parikh and K. Karlapalem. Et: events from tweets. In *Proceedings of the 22nd international conference on World Wide Web companion*, pages 613–620. International World Wide Web Conferences Steering Committee, 2013.

[27] T. Sakaki, M. Okazaki, and Y. Matsuo. Earthquake shakes twitter users: Real-time event detection by social sensors. In *WWW*, 2010.

[28] D. A. Shamma, L. Kennedy, and E. F. Churchill. Peaks and persistence: modeling the shape of microblog conversations. In *Proceedings of the ACM 2011 conference on Computer supported cooperative work*, pages 355–358. ACM, 2011.

[29] I. Tien, A. Musaev, D. Benas, and C. Pu. Detection of damage and failure events of critical public infrastructure using social sensor big data. In *Proceedings of International Conference on Internet of Things and Big Data*, April 2016.

[30] K. N. Vavliakis, A. L. Symeonidis, and P. A. Mitkas. Event identification in web social media through named entity recognition and topic modeling. *Data & Knowledge Engineering*, 88:1–24, 2013.

[31] M. Walther and M. Kaisser. Geo-spatial event detection in the twitter stream. In *Proceedings of the 35th European Conference on Advances in Information Retrieval*, ECIR'13, pages 356–367, Berlin, Heidelberg, 2013. Springer-Verlag.

[32] D. Wang, T. Abdelzaher, and L. Kaplan. *Social Sensing: Building Reliable Systems on Unreliable Data.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2015.

[33] D. Wang, T. Amin, S. Li, T. A. L. Kaplan, S. G. C. Pan, H. Liu, C. Aggrawal, R. Ganti, X. Wang, P. Mohapatra, B. Szymanski, and H. Le. Humans as sensors: An estimation theoretic perspective. In *IPSN*, 2014.

[34] D. Wang, L. Kaplan, H. Le, and T. Abdelzaher. On truth discovery in social sensing: a maximum likelihood estimation approach. In *IPSN*, 2012.

[35] 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 *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, pages 2541–2544, New York, NY, USA, 2011. ACM.

[36] J. Weng and B.-S. Lee. Event detection in twitter. *ICWSM*, 11:401–408, 2011.

[37] Y. Yang, J. G. Carbonell, R. D. Brown, T. Pierce, B. T. Archibald, and X. Liu. Learning approaches for detecting and tracking news events. *IEEE Intelligent Systems*, (4):32–43, 1999.

[38] Y. Yao and J. Gehrke. The cougar approach to in-network query processing in sensor networks. *ACM Sigmod Record*, 31(3):9–18, 2002.

[39] C. Zhai and S. Massung. *Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining.* Association for Computing Machinery and Morgan &#38; Claypool, New York, NY, USA, 2016.

[40] C. Zhang, G. Zhou, Q. Yuan, H. Zhuang, Y. Zheng, L. M. Kaplan, S. Wang, and J. Han. Geoburst: Real-time local event detection in geo-tagged tweet streams. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, SIGIR 2016, Pisa, Italy, July 17-21, 2016*, pages 513–522, 2016.

[41] D. Zhang, D. Gatica-Perez, S. Bengio, and I. McCowan. Semi-supervised adapted hmms for unusual event detection. In *IEEE CVPR*. IEEE, 2005.

[42] S. Zhong. Efficient streaming text clustering. *Neural Networks*, 18(5):790–798, 2005.

[43] X. Zhou and L. Chen. Event detection over twitter social media streams. *The VLDB journal*, 23(3):381–400, 2014.

APPENDIX: DISTANCE FUNCTIONS IN EVENT SIGNATURE CONSOLIDATION

Let S_i be the set of key words of the tweet cluster C_i. For each word, $w \in S_i$, let $f_i(w)$ denote its frequency. For notation simplicity, if $w \notin S_i$, we define $f_i(w) = 0$. We investigate four broadly applied distance metrics described below for defining the lexical frequency domain distance between clusters.

- **Jaccard Distance:** The Jaccard similarity (JS) between clusters C_i and C_j is defined by
$$JS(i,j) = |S_i \cap S_j| / |S_i \cup S_j|,$$
where $|S|$ denotes the cardinality of the set S. The Jaccard distance is defined by $1 - JS(i,j)$.
- **Term Frequency Difference Ratio:** The term frequency difference (FD) between clusters C_i and C_j is defined by:
$$FD(i,j) = \sum_{w \in S_i \cup S_j} |f_i(w) - f_j(w)|,$$
where $abs(X)$ denotes the absolute value of X. The term frequency difference ratio is the normalized term frequency difference, i.e. $\frac{FD(i,j)}{\sum_{w \in S_i} f_i(w) + \sum_{w \in S_j} f_j(w)}$.
- **Cosine Distance:** Cosine Similarity (CS) is defined by
$$CS(i,j) = \frac{\sum_{w \in S_i \cap S_j} f_i(w) \times f_j(w)}{\sqrt{\sum_{w \in S_i} (f_i(w))^2} \times \sqrt{\sum_{w \in S_j} (f_j(w))^2}}.$$
CS measures the cosine of the angle between two vectors whose elements are f_i and f_j. The more similar the two vectors, the smaller the angle between them. The Cosine distance is defined by $1 - CS(i,j)$.
- **KL Divergence:** The KL divergence, KL, is a non-symmetric measure of the difference between two probability distributions, defined as follows in our case:
$$KL(i,j) = \sum_{w \in S_i \cup S_j} p_i(w) \ln \frac{p_i(w)}{p_j(w)},$$
$$p_i(w) = \frac{f_i(w)}{\sum_{w \in S_i \cup S_j} f_i(w)}, p_j(w) = \frac{f_j(w)}{\sum_{w \in S_i \cup S_j} f_j(w)}.$$
Note that, when $f_i(w) = 0$ or $f_j(w) = 0$, $KL(i,j)$ is malformed. To avoid this problem, we add 1 to $f_i(w)$ and to $f_j(w)$ for all w.