BigEye: Detection and Summarization of Key Global Events From Distributed Crowdsensed Data

Abdulrahman Fahim, Ajaya Neupane, Evangelos E. Papalexakis, *Member, IEEE*, Lance Kaplan, *Fellow, IEEE*, Srikanth V. Krishnamurthy, *Fellow, IEEE*, and Tarek Abdelzaher, *Fellow, IEEE*

***Abstract*—Social media postings using smartphones (referred to as crowd-sensed data) can often facilitate real-time detection of key physical events in applications like disaster recovery or in smart cities. These postings also often contain visual content (e.g., images) that can be used to obtain zoomed-in views of such events. These crowd-sensed data are likely to be of large volume and distributed across a plurality of producers (e.g., cloudlets). Blindly transferring this large volume of raw data from the producers to a consumer will induce information overload and consume very high bandwidth. The problem is exacerbated in scenarios with limited bandwidth (e.g., after a disaster). In this article, we design `BigEye`, a novel framework that only transfers very limited data from the distributed producers to a central summarizer, and yet supports: 1) highly accurate detection and 2) concise visual summarization of key events of global interest. In realizing `BigEye`, we address several challenges including: 1) identifying events that have the highest global interest via the transfer of appropriate limited metadata from the producers to the summarizer; 2) reconciling metadata that could be inconsistent across the producers; and 3) the timely retrieval of visual summaries of the key events given bandwidth constraints. We show that `BigEye` achieves the same accuracy in detecting key events, as a system, where all data are available centrally while transferring only 1% of the raw data volume. Compared to the baseline approaches, `BigEye`'s parallelized transfer of visual content reduces the average delay by 67%.**

***Index Terms*—Crowd sensing, data summarization, distributed event detection, resource constrained networks, smart cities, social sensing, visual summaries.**

I. Introduction

PEOPLE share and disseminate postings on real-life events on social networks via smartphones (which can be considered the de facto most widely used IoT devices today). Such postings provide an inherent sensing capability [1] (defined as crowdsensing or social sensing in [2]). Specifically, as pointed out in prior papers [1], [3], [4], user posts relating to real-time information about events (e.g., protests and earthquakes) on social media networks, can be considered as IoT sensor outputs that provide knowledge of interest with regards those events. In fact, news reports suggest that Harvey storm victims used social media to communicate about critical events [5] during/after the disaster. We also point out that it is quite common for such user postings (e.g., using Twitter) to contain images that can be used for composing *zoomed-in* (or more informative) visual summaries of key events.

In scenarios like disaster aftermaths, one can visualize crowdsensed data to be dispersed across a set of geographically distributed "producers," because of the strained infrastructure. For example, one can envision the user postings (we also call them microblogs) within a local geographical region to be sent (streamed) to a common server (e.g., a cloudlet [6] or a local server [7], [8]), which can be considered to be a *producer*. While using microblogs as IoT-sensed outputs toward detecting physical events has received some recent attention (e.g., [3] and [4]), these prior efforts assume that all crowdsensed data are available centrally instead of being distributed across geographically spread producers; the latter is a more accurate representation of what happens in practice.

Blindly transporting all the raw data from such producers to either the consumer, or to a central entity for analytics will not only cause an information overload at that entity but also strain the network in terms of bandwidth consumption potentially resulting in network congestion. In addition, a large part of this voluminous data will have redundant information, and might not contribute to extracting useful information (e.g., detecting events of interest). The problem of data transportation is exacerbated in scenarios such as natural disasters wherein it is very common for network infrastructure to be damaged; in such cases, the available bandwidth is constrained significantly making it prohibitive to transfer raw data. Again, going back to the example of the tropical storm Harvey, the failure of several cell towers [9] strained the available bandwidth. Because of this, it becomes imperative to only retrieve small amounts of data from the distributed producers, and yet be able to detect events of global significance with high accuracy. Our work targets this important problem.

Specifically, in this article, we design an IoT service, we call `BigEye`, using which global events can be detected with very high accuracy, by only sharing very small amounts of information (metadata) between the distributed producers (with raw crowdsensed data) and a central entity (which we call summarizer). Once global events are detected, `BigEye`

Manuscript received August 31, 2021; revised December 17, 2021; accepted January 19, 2022. Date of publication February 1, 2022; date of current version June 7, 2022. This work was supported in part by the DEVCOM Army Research Laboratory under Agreement W911NF-09-2-0053, and in part by NSF CPS under Grant 1544969. *(Corresponding author: Abdulrahman Fahim.)*

Abdulrahman Fahim, Ajaya Neupane, Evangelos E. Papalexakis, and Srikanth V. Krishnamurthy are with the Department of Computer Science and Engineering, University of California at Riverside, Riverside, CA 92521 USA (e-mail: afahi002@ucr.edu).

Lance Kaplan is with the Signal and Information Processing Division, U.S. DEVCOM Army Research Laboratory, Adelphi, MD 20783 USA.

Tarek Abdelzaher is with the Department of Computer Science, University of Illinois at Urbana–Champaign, Urbana, IL 61820 USA.

Digital Object Identifier 10.1109/JIOT.2022.3148000

facilitates the transport of limited content in the form of images/videos to the summarizer, which then composes a visual summary from the same detected events to provide the appropriate information to gain insights with regards to these events of interest. These summaries can then be sent to a consumer (e.g., search and rescue personnel after a disaster). We implement BigEye using Twitter (which provides user postings of the type of interest) as a proxy, and showcase its benefits. Thus, without loss of generality, in the rest of this article, we refer to user postings as tweets.

Challenges: The challenges in designing BigEye are multifold. First, each producer only has a local view of events and, thus, is unable to by itself determine which events are of global significance. On the other hand, transferring all local data from all producers to the central summarizer is prohibitive and wasteful. The challenge then is "how do we identify key global events by only transferring limited metadata from each producer to the summarizer?" Second, multiple producers may detect the same global event via sensed triggers, but there is no easy way to determine that the triggers correspond to the same event. Unless we are able to make this determination, redundant content (e.g., unnecessary visual data) may be retrieved by multiple producers. Thus, "how do we reconcile (possibly inconsistent) metadata from multiple producers that correspond to a common event?" Finally, once the events are detected, we seek to retrieve a limited amount of visual content with minimum latency from the multiple producers to compose a summary. The challenge here is to identify "which producer will send what content in order to maximize the level of parallelization of retrieval, given that the bandwidths to the different producers could be different."

Approach in Brief: Briefly, with BigEye, each producer first individually identifies local events likely to have global importance (or interest) using a metric which is called the local information gain. The producer then *pushes* metadata pertaining to these local events to a central summarizer. The summarizer then assesses the need for any additional metadata and *pulls* the same from the proper subset of producers. As one of our contributions, we formally show that via an appropriate choice of the number of local events at each producer, we can guarantee that the detected key global events are identical to the ones detected if all the data are made available centrally. Once the global events are detected, BigEye uses a lightweight method to reconcile common events across a plurality of producers. Finally, using lightweight measurements of bandwidths to the various producers, it uses an intelligent algorithm to parallelize the transfer of visual content from the producers to compose visual summaries of all such events within a very short time (ideally, we want to do this in minimum time but we show this is NP-hard). This article is an extension of our work [10], where, in this work, we introduce all BigEye modules and show its overall merits.

Summary of Our Contributions: A summary of our contributions in this article are as follows.

1) We propose a method within BigEye that allows the detection of key global events when the crowdsensed data are distributed across multiple distributed producers. Via the transfer of just 1% of the data from the producers to a central summarizer (in the form of metadata), we show that BigEye is able to detect all key global events that would have been detected if all the data was available centrally.
2) BigEye includes a module that reconciles events across producers without having to transfer the entire content corresponding to these events to the central entity (Section V). We call this "consolidation" of events across producers. We show that consolidation further reduces the communication costs by 60% on average.
3) We show that we can map the problem of selecting visual objects with the highest scores to be sent from each producer to the summarizer, such that we ensure a certain number of visual objects per event while minimizing latency, to a multidimensional knapsack instance [11]; this is an NP-Hard problem. Given this, we design an online algorithm that: a) dynamically estimates bandwidth to each producer and b) fetches visual objects in parallel greedily from the producers given these estimates. Our experiments show that compared to baseline approaches, BigEye's parallelized transfer of visual content reduces delay by 67%.

II. Baseline Case: Centralized Detection

We first provide a description of a baseline method which allows global event detection when all the data are centrally available. The approach is largely based on the approach (called Storyline) by Wang *et al.* [4]. We start by describing the "information gain metric" used to detect global events in Storyline in this section. In Section IV, we show how we can transition from this global metric to an appropriate local metric that facilitates the detection of key global events in the distributed settings that we are interested in.

"Information gain" is a commonly used metric for detecting discriminative features [12]. In Storyline, this metric is used to measure the burstiness in the co-occurrence of pairs of uncommon words (keyword pair) in a stream of tweets (microblog objects are used as sensor outputs) between two time epochs. The keyword pairs are considered to be *discriminative features* and to be associated with physical events of interest. For example, given a key physical event where "a drunk driver kills a running dog on the bridge," the microblog data would experience a bursty surge in the keyword pair (driver, drunk) at a specific time epoch. The approach enables event demultiplexing, i.e., identifying separate events that belong to the same global scope. For example, in a disaster scenario, one might be interested in distinguishing between multiple areas with humans in distress (and not identify them as the same issue) so that the human responders can take proper actions. Formally, the information gain associated with a keyword pair s_z, across time windows $k-1$ and k is given by [4]

$$\text{infoGain} = H(Y) - H(Y|s_z). \tag{1}$$

In the above equation, $H(Y)$ and $H(Y|s_z)$ are computed as follows:

$$H(Y) = -\frac{N_k}{N_k+N_{k-1}}\log\frac{N_k}{N_k+N_{k-1}} - \frac{N_{k-1}}{N_{k-1}+N_k}\log\frac{N_{k-1}}{N_k+N_{k-1}} \quad \text{and} \tag{2}$$

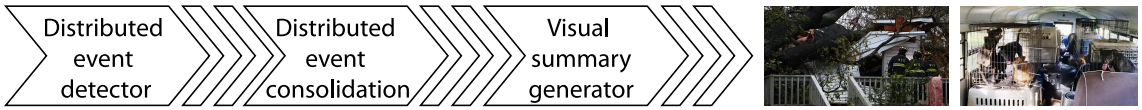

Fig. 1. High level depiction of BigEye with its modules.

$$H(Y|s_z) = -\frac{N_k^z}{N_k^z + N_{k-1}^z} \log \frac{N_k^z}{N_k^z + N_{k-1}^z} - \frac{N_{k-1}^z}{N_{k-1}^z + N_k^z} \log \frac{N_{k-1}^z}{N_k^z + N_{k-1}^z} \quad (3)$$

where N_k is the number of tweets in the current (kth) time window, and N_{k-1} is the number of tweets in the immediately previous time window, $k-1$. N_k^z and N_{k-1}^z are the number of tweets that contain the keyword pair s_z in the current and previous time window, respectively. Note that, roughly, these expressions characterize the entropy (conditional entropy) relating to the occurrence of tweet volume (number of tweets with the specific keyword pair) in consecutive time windows. More details can be found in [4] and are omitted in the interest of space. A threshold is chosen, and all keyword pairs whose information gains are higher than that threshold are considered to have associated physical events that are of interest.

III. Overview and Assumptions

We envision that BigEye is used in an architecture that comprises a set of producers, a summarizer, and one or more consumers (users). BigEye seeks to identify key events of global interest, occurring at specific time windows of fixed duration (we also call these windows epochs). A producer is an entity that collects sensed data from within a local region (e.g., a microblog repository). Let m be the number of producers; each producer is denoted by P_i, where $i = 1, 2, \ldots, m$. In BigEye, we assume that the time epochs are synchronized across all producers (we assume that protocols such as NTP can enable this [13]); without loss of generality, we denote the index of the time window of interest by k.

The summarizer is a central entity (e.g., a server) that receives appropriate data from all the producers, identifies key events, and composes a summary. We assume that all the producers are connected to the summarizer via a network of arbitrary topology. The transfer of all local data from m producers to the central summarizer is considered to be large and prohibitive in terms of bandwidth consumption (either because of limited bandwidth, congestion, or both). This will typically be the case in scenarios such as disaster recovery, wherein loss of infrastructure can induce significant constraints on the available bandwidth (e.g., fewer functional base stations or access points). Blindly sending the large volume of data can have significant consequences. First, because of the sheer volume, significant delays may be experienced. This, in turn, induces significant delay in aggregating the data of interest (or importance) from the multiple producers. This, in turn, delays the inferences relating to events detection, which can cause response delays (especially in disaster scenarios when fast response is critically). We point out that if instead, inferences are based on partially received data, suboptimal decisions may be made and the response resources may be directed at less than critical points of need.

Without fine-grained information relating to the events of interest, the summarizer may experience confusion with respect to multiple events, often categorizing them to be the same. This increases the rate of false negatives (some of these may be missed completely). It is also possible that the false-positive rates relating to events could increase due to decisions made with coarse grained information. In this work, we propose, BigEye, a system that deals with the aforementioned conditions and is able to detect events accurately with low latency. Furthermore, it has the ability to disambiguate events that occur across a wide reach (referred to as global events), thereby allowing their precise identification for proper response delivery.

In BigEye, each producer identifies a set of local events that are likely to be of global interest or importance. It generates appropriate metadata corresponding to these events, and pushes the metadata to the summarizer. While multiple producers may report the same global event albeit with inconsistent metadata, BigEye allows the summarizer to reconcile/consolidate such events. With BigEye, the summarizer also triggers the intelligent retrieval of visual content from the appropriate producers to compose visual summaries of all the key events within a very short time. These precise visual summaries will enable disambiguation across events, thereby allowing response delivery to all events that are of importance. A depiction of the modules and composition of BigEye is shown in Fig. 1.

A consumer or user is connected to the summarizer, and queries for summaries of key events. We assume that a query provided by a user belongs to a specific scope (e.g., protest, army conflicts, etc.) and the producers collect the data that match such queries.[1] Each producer observes postings composed of short descriptions of their context with attached multimedia (videos/images) (e.g., tweets). BigEye takes this information as an input and produces summaries that are composed of global events of interest plus a visual summary that provides additional context which provide detailed information and also allow disambiguation of similar but separate events.

Motivating Our Vision: To exemplify our vision, we describe an example where our framework can find application and help significantly. A rescue operator is interested in identifying/detecting the key global events in a disaster affected area. Let us recall the previously mentioned example relating to hurricane Harvey, wherein the victims posted tweets relating to their surroundings, with textual and visual content. The postings are sent to the nearest cell tower or cloudlet (we refer

[1]The communications between the user and the summarizer are out of the scope of this article.

TABLE I
KEY NOTATION

Symbol	Description
N_k	# of tweets in time window k
N_k^z	# of tweets in time window k that contain the pair s_z
p_s	$\frac{N_k^z}{N_k^z+N_{k-1}^z}$
k	index of the data stream window
$\in$	$H(Y) - threshold$
m	# of producers
i	Producer index
P_i	i^{th} producer in the the system
r	ratio of occurrence of keyword pair s_z in time window k to the corresponding occurrence in time window $k-1$
n	total # of multimedia objects
j	multimedia object index
w_j	size of the j^{th} multimedia object
f_j	score of the j^{th} multimedia object
B_i	Bandwidth between P_i and the summarizer
C_e	# of multimedia objects inquired from event e

to these as producers in this article); because of bandwidth constraints, carrying a possibly a large sets of tweets with visual content beyond this local infrastructure in near real-time will be problematic. Thus, a central controller where the rescue operator (user) resides might not be able to receive this information, or process it in a timely manner. Here, is where `BigEye` comes into place; it enables the rescue operator who only has limited connectivity to the producers, to identify the key global events without retrieving the raw generated data in its entirety, from the producers. For example, the global event may be a "major flooding in a specific street or neighborhood," "pets are in danger in a certain area" or "the 911 network has broken-down." The rescue operator can then take proper action, based on the detected "event-level" information.

The key notations used in this article are summarized in Table I.

IV. DISTRIBUTED EVENT DETECTION

Compared to prior approaches (in particular, the baseline described in Section II), `BigEye` distinguishes itself in that it applies in a realistic scenario where the crowdsensed data are dispersed across a plurality of geographically distributed producers. If one was to blindly transport all the raw data that are available at these producers to a central summarizer (which can then create summaries as discussed before), the bandwidth consumed will be prohibitive, especially in scenarios such as disaster recovery. Because of this constraint, when building `BigEye`, we try to answer the question "How can we determine the events that are of global significance without having to transfer all raw data to the central summarizer?"

The information gain metric previously discussed in Section II is based on an underlying assumption that the entire raw data are centrally available. However, this assumption does not hold and each producer only has a local view. Thus, needed is an approach wherein the significance of a local event can be estimated with respect to its the global importance (i.e., will a local event also be flagged as a key global event if the data was available centrally?). If all the producers were to simply report the "number of tweets" to the summarizer, in the two consecutive time windows of interest (say $k-1$ and k), $H(Y)$ can be computed. However, the challenge lies in computing $H(Y|s_z)$ globally since: 1) all keyword pairs and the associated tweets are not known centrally and 2) each producer will only see part of the data and can only compute $H(Y|s_z)$ based on its local data set. To address these issues, we first map a global requirement on information gain (and, thus, $H(Y|s_z)$) to a local requirement at each individual producer. Later, we reconcile inconsistencies by having the summarizer pull appropriate data from a subset of producers.

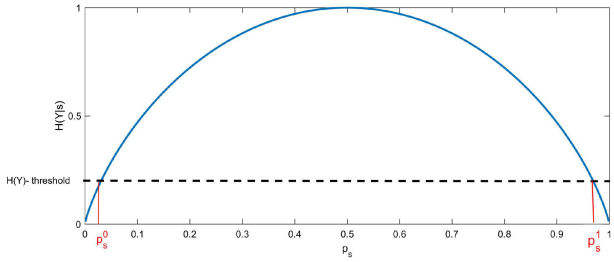

Fig. 2. $H(Y|s)$ with varying p_s. p_s^0 and p_s^1 are two intersecting points with $H(Y)-$ threshold.

What values should the global $H(Y|s_z)$ take to achieve high information gain? Before, we describe our approach in more detail, we first ask the above question. As pointed out above, $H(Y)$ only depends on the number of tweets in consecutive time windows. Thus, the discriminatory term that dictates the information gain with respect to a keyword pair (say s_z) is $H(Y|s_z)$. It is obvious that the lower the value of this term, the higher the information gain associated with the keyword. With reference to (3), let us define $p_s = [N_k^z/(N_k^z + N_{k-1}^z)]$. Then, $[N_{k-1}^z/(N_{k-1}^z + N_k^z)] = 1 - p_s$. In Fig. 2, we plot $H(Y|s_z)$ as a function of p_s. We see that the lowest values of $H(Y|s_z)$ (yielding the highest information gain) are achieved when p_s is very small or very large (approaches 1). Since p_s corresponds to the probability of having a high number of tweets in frame k relative to the previous frame, it must be large (not small) in order to reflect a new event of interest (otherwise, it indicates an event that was of interest in frame $k-1$ but has died down). In other words, the takeaways from the above discussion are: 1) $H(Y|s_z)$ must be small (say some small value ϵ) and 2) the corresponding probability p_s as defined above must be large (we require it to be > 0.5).

Let r be the ratio of occurrence of a keyword pair in the current time epoch to the corresponding occurrence in the previous time epoch. Let p_s^* be the value of p_s that makes $H(Y|s_z) = \epsilon$. Since we cannot directly obtain p_s^* in a closed form by solving $H(Y|s_z) = \epsilon$, we numerically solve it using the Newton method [14] and from among the results, choose the value that is > 0.5. From p_s^*, we compute (N_k^z/N_{k-1}^z) and denote it as r^*. r^* is the minimum (global) threshold with respect to the ratio of occurrences of a keyword pair in consecutive time epochs, that must be met if the associated keyword pair is to signify an event of interest. In other words, if for a keyword pair $r \geq r^*$, then that keyword pair is a discriminative pair. Next, we provide a formal proof of this claim.

Lemma 1: Any pair with $p_s \geq p_s^*$ has a ratio of occurrence, r greater than or equal to r^*.

Proof:

$$p_s = \frac{N_k^z}{N_k^z + N_{k-1}^z} \tag{4}$$

$$p_s N_k^z - N_k^z = -N_{k-1}^z p_s \tag{5}$$

$$\frac{N_k^z}{N_{k-1}^z} = \frac{p_s}{1-p_s}. \tag{6}$$

Note that r is nothing but (N_k^z/N_{k-1}^z). If we can show that $[p_s/(1-p_s)]$ is a nondecreasing function, i.e., $[p_s/(1-p_s)] \geq [p_s^*/(1-p_s^*)]$ if $p_s \geq p_s^*$, then we can infer that $r \geq r^*$ if $p \geq p_s^*$. Let us denote $[p_s/(1-p_s)]$ by $F(p_s)$. We show that $F(x) \leq F(y)$ for $0 < x \leq y < 1$. We need to show that

$$\frac{x}{1-x} \leq \frac{y}{1-y} \text{ or} \tag{7}$$

$$x(1-y) \leq y(1-x) \text{ or} \tag{8}$$

$$x - xy \leq y - xy \text{ or} \tag{9}$$

$$x \leq y \tag{10}$$

which is true by assumption. ∎

Choosing a Local Threshold: Given the global threshold r^*, we need to derive an appropriate local threshold; each producer would estimate r with respect to each keyword pair and if this r is lower than the local threshold one can deem that those keyword pairs are not of global interest. In order to retrieve all the discriminative pairs of global interest (those that would have been detected if all data was available centrally) we need to be conservative, i.e., the choice of the local threshold must account for the worst case scenario. By doing so, we can achieve the same precision and recall values with `BigEye`, compared to a centralized baseline (discussed in Section II). We point out here that one may experience an outlier case, where there are no (zero) tweets with a keyword pair in window $(k-1)$ but a significant number in window k; to avoid the divide by zero possibility, we assume that each pair appears at least once at each time window; this fix has almost no influence on the ratios that we are trying to compute.

Given the above threshold and based on the following theorem, we choose the local threshold to be (r^*/m) if there are m producers.

Theorem 1: If a keyword pair has a global ratio of occurrence r which is $\geq r^*$, the *local* ratio of occurrence of that keyword pair must be larger than or equal to (r^*/m) at one or more of m producers.

Proof: Let the number of occurrences of a keyword pair in the current and previous time windows be N_k^z and N_{k-1}^z, respectively. The ratio of occurrence of the pair at the global level $(N_k^z/N_{k-1}^z) \geq r^*$.

Case 1 ($[N_k^z/m]$ Is an Integer): Let the number of occurrences of the pair in the current time window (window k) at the local producers be the following: $(N_k^z/m) + c_1$, $(N_k^z/m) + c_2, \ldots, (N_k^z/m) + c_m$, where c_i and (N_k^z/m) are integers.

The summation of all the occurrences, across all producers should be equal to the global count N_k^z, i.e., $([N_k^z/m] + c_1) + ([N_k^z/m] + c_2) + \cdots + ([N_k^z/m] + c_m) = N_k^z$.

Thus, $m(N_k^z/m) + \sum_i c_i = N_k^z$; hence, $\sum_i c_i = 0$.

Case 1A (P_i With $c_i \geq 0$): The ratio of occurrence of the keyword pair of interest at P_i is $([(N_k^z/m) + c_i]/N_{k-1}^{z'})$, where $1 \leq N_{k-1}^{z'} \leq N_{k-1}^z$. Here, $N_{k-1}^{z'}$ is the number of occurrences of the keyword pair in the previous time window [(window $(k-1)$] at P_i; naturally, this is $\leq$ the global count in that window.

The next step shows that the theorem holds regardless of the value of $N_{k-1}^{z'}$

$$\frac{\frac{N_k^z}{m} + c_i}{N_{k-1}^{z'}} = \frac{N_k^z}{mN_{k-1}^{z'}} + \frac{c_i}{N_{k-1}^{z'}} \geq \frac{N_k^z}{mN_{k-1}^{z'}} \geq \frac{N_k^z}{mN_{k-1}^z}.$$

But $[N_k^z/(mN_{k-1}^z)] \geq (r^*/m)$ and hence, the local rate of occurrence at P_i is higher than (r^*/m).

Case 1B (P_i With $c_i < 0$): Since $\sum_i c_i = 0$, there must be at least one other producer with $c_l > 0, l \neq i$; Case 1A will now apply to that producer l.

Case 2 ($[N_k^z/m]$ Is Not an Integer): If all occurrences at local producers are $\lfloor (N_k^z/m) \rfloor$, their summation becomes smaller than N_k^z. Hence, the number of occurrences with respect to at least one of the producers, denoted as P_i, must be $\geq \lceil (N_k^z/m) \rceil$. In other words, the ratio of occurrence at P_i is $[\lceil (N_k^z/m) \rceil / N_{k-1}^{z'}]$, where $1 \leq N_{k-1}^{z'} \leq N_{k-1}^z$. Similar to the previous case, $[\lceil (N_k^z/m) \rceil / N_{k-1}^{z'}] \geq (r^*/m)$. ∎

Distributed Event Detection Algorithm: Based on the above findings, `BigEye` applies the following algorithm for distributed event detection.

1) Each producer computes the ratio of occurrences of keyword pairs available locally and transmits the pairs having ratios larger than or equal to (r^*/m) to the summarizer.
2) The summarizer sends a list of the received pairs to all the producers and inquires about the occurrences of those pairs at the producers. Any producer that had identified that keyword pair, but had not reported it (because it did not meet the threshold) now reports the number of occurrences of that pair. Once this information is available, the summarizer computes the global ratios of all pairs received in step (1).
3) The summarizer filters out pairs with the global ratios less than the global threshold, r^*.

Theorem 1 proved that any globally significant keyword pair "will" be reported by at least a single producer in the first step above. This proves the following lemma.

Lemma 2: `BigEye`'s distributed detection algorithm achieves 100% precision and recall, relative to centrally available data.

Discussion: The performance of our algorithm degrades when the local threshold is very small. When the local threshold becomes very small, the number of keyword pairs sent by the producers to the summarizer increases drastically. Specifically, this happens when the number of producers m is very large or the global threshold r^* is very small, or both. When the local threshold becomes very small (say has a value 1) each producer sends all the pairs; to avoid division by 0 we had implicitly set the ratio of occurrence of any pair

at a local producer to be greater than or equal to 1. However, in practice, these cases are not of interest. A very large set of producers will imply that the local data consists of small sets and, thus, it will be hard to detect events that are of global interest. A very small threshold will also fail in discriminating between key events of interest and others.

V. Event Consolidation

Different discriminative pairs detected by the summarizer (based on local reports from the producers) may refer to the same physical event. This is because a single event can be characterized by multiple discriminative keyword pairs. We demonstrate this phenomenon using an example. The discriminative keyword pairs (boy, drowning) and (boat, rescue) could refer to the same event where a drowning boy in a river, was rescued by a fishing boat. Because these allude to the same event, we need to merge the key word pairs and, thus, avoid the unnecessary retrieval of redundant visual summaries pertaining to this same physical event. For merging similar events, `BigEye` packs microblogs containing specific "keyword pairs" into clusters. To consolidate two keyword pairs representing the same event, the similarity between two clusters represented by these keyword pairs is computed. If the similarity score is larger than a given threshold both the clusters pertaining to the keyword pairs are consolidated.

Assessing this similarity in a distributed setting is challenging. Naively sending the entire cluster of words associated with a keyword pair to the summarizer for computing similarity scores defeats the purpose of reducing communication costs. Thus, `BigEye` consists of an approach to consolidate events across producers, in bandwidth constrained distributed environments. The approach consists of two steps described below.

Step I: In the first step, `BigEye` tries to consolidate keyword pairs representing similar events at the local producers. Specifically, it consolidates events corresponding to "keyword pairs," the clusters associated with which have content that are *very similar*. It uses the Jaccard distance [15] to measure the similarity between the two clusters (events). Previous work [4] has reported that the Jaccard distance outperforms other similarity metrics for event consolidation in this way.

Returning to our earlier example, "a drowning boy was rescued by a fishing boat in the river" has the discriminative keyword pairs (boy, drowning) and (boat, rescue). One can expect that the similarity score between the two local clusters of the corresponding discriminative keyword pairs (boy, drowning) and (boat, rescue) to be high (we find such scores to be consistent with what is observed in the case when all data are available centrally). Based on this, the similarity between clusters of microblogs is computed at each local producer. If the distance between any two clusters of events exceeds a certain threshold with regards to the Jaccard distance, the local producers notify the central summarizer that they should be consolidated.

Note here that such approaches (although not identical to what we propose) have been previously used in event detection [4], [16]. In our case, we start by having a number of clusters equal to the number of detected keyword pairs; after the consolidation phase, we end up having only C clusters of microblogs representing the physical events. Our algorithm is similar to the one used in [4] for consolidating similar events.

To assess the similarity between microblogs belonging to two clusters, we measure the similarity between the words belonging to the two sets of microblogs. Specifically, we measure the ratio of the number of unique words that are present in "both" sets of microblogs to the total number of unique words in both sets. This metric referred to as Jaccard similarity [15] has been reported to outperform other metrics in assessing the similarity of data sets, and in particular for event consolidation [4]. At summarizer, if the majority of the producers ($\geq$50%) indicate that two keyword pairs should be consolidated, the summarizer sends feedback to all the producers to merge the contents associated with these keyword pairs. This helps to consolidate highly similar events at individual producers, without sending the entire cluster contents to the summarizer.

Step II: In the second step, `BigEye` further tries to consolidate global events that do not have very high similarity locally at individual producers, and were not consolidated in step I. To achieve this objective, it seeks to only exchange minimal information with the summarizer to limit bandwidth consumption. Specifically, it employs minHash [17] and locality sensitive hashing (LSH) [18] functions at each producer, to convert a cluster of words represented by a discriminative keyword pair into a set of hash integers. minHash and LSH are techniques commonly used to measure the similarity of large documents within reasonable running times [18], [19]. The probability that hashes of two sets are similar is equivalent to the corresponding Jaccard similarity of the same sets [19], [20]. Each producer transmits the computed hash values to the summarizer. The summarizer compares the hash values across clusters to measure the similarity globally. The bandwidth consumed on sharing the value generated by minHash is significantly smaller than the bandwidth consumed on sharing the entire cluster to the summarizer (as will be shown in Section VII). At this point, `BigEye` has tried to reconcile the possibility that a single global event of interest was perhaps identified as different events because there were multiple keyword pairs from tweets that were used as discriminatory features for this event. While we are not able to completely eliminate a single event being wrongly classified as multiple events, this process drastically reduces the possibility.

VI. Composition of Visual Summaries

The final module of `BigEye` deals with the retrieval of a set of multimedia objects (e.g., images) from producers to visually summarize an event; the goal is to achieve this retrieval in the minimum amount of time. Without loss of generality, we assume that the number of objects (fixed) required to compose a visual summary for each detected event, is set *a priori* (e.g., by a consumer) and that this number is the same for all events of interest. Without *a priori* knowledge of what events occur, this ensures that every event is summarized fairly, with no priority of one event over others. However, by applying proper

weights it is easy to extend the approach to account for cases wherein events are to be prioritized.

A summary should include multimedia objects that best describe the corresponding event of interest. Thus, we assign a score to each object based on a set of features (e.g., retweet count, favorite count, etc.). We denote this set of selected features as $f_1, f_2, \ldots, f_F$ and the score associated with each item is equal to $f_j = \gamma_1 f_1 + \gamma_2 f_2 + \cdots + \gamma_F f_F$, where j is the item index. The summarizer then looks for a set of multimedia objects that maximizes $\boldsymbol{f} = \sum f_j$

$$\begin{aligned}
\text{maximize} \quad & \sum_{j=1}^{n} f_j x_j \\
\text{subject to} \quad & \sum_{j=1}^{n} w_{ij}^1 x_j \le B_i, \ i = 1, \ldots, m \\
& \sum_{j=1}^{n} w_{ej}^2 x_j \le C_e, \ e = 1, \ldots, E \\
& w_{ij}^1 = \begin{cases} w_j, & j \in P_i \\ 0, & \text{otherwise} \end{cases} \\
& w_{ej}^2 = \begin{cases} 1, & j \in C_e \\ 0, & \text{otherwise} \end{cases} \\
& x_j \in \{0, 1\}, \ j = 1, \ldots, n. \qquad (11)
\end{aligned}$$

The objects to be retrieved may be of heterogeneous sizes; w_j denotes the size of an object j. Let the bandwidth between a producer and the summarizer be B_i, where i is the producer index. We denote the number of images to be retrieved from each cluster (event) by C_e. The total number of events is denoted as E. Since there is a bandwidth constraint imposed by the network, retrieving the images with the highest quality (score) may induce large delays. Selecting what to send from each producer to maximize the objective function while respecting the bandwidth and the event coverage constraints is an NP-Hard problem [21]. This is because the problem can be mapped to a multidimensional knapsack instance as shown in (11). The optimization problem aims to maximize the objective function by picking up the multimedia objects with the highest scores subject to a set of constraints—the bandwidth between the producer and the summarizer, and the number of images belonging to a certain event should not exceed the limit that is imposed. This is akin to filling up a multidimensional knapsack with objects subject to constraints on the knapsack sizes.

Online Algorithm: Solving the above optimization problem is not trivial for two reasons. First, as mentioned above, it is known that knapsack is an NP-Hard problem [21] and, thus, obtaining the optimal solution would require an exponential running time. Second, in real-world scenarios, the bandwidth allocated to a producer is not known *a priori*. The estimated bandwidth in the idle state is different from the available bandwidth in real time, as the producers might share the same paths and, thus, affect the bandwidths available to each other (in addition to traffic dynamics). Thus, the optimization problem needs to be solved online. Generally, this problem is known in the literature as the knapsack of unknown capacity [22].

As evident from the problem formulation, there is a trade-off between the retrieval time and the quality of the retrieved items (the total score). We design our algorithm such that it is flexible to allow the operator to favor the quality over time or vice versa. First, each producer sends the metadata (size and score) of multimedia objects to the summarizer. Next, the summarizer sorts the objects based on a rank given by the ratio, $[(1 + Lf_j)/w_j]$. This ratio captures the relation between the score of the object and its size. L is a normalizing factor that can allow favoring one metric (e.g., score) over the other as desired by the deployer. For example, if $L = 0$, the algorithm retrieves the objects with the smallest sizes which lead to shorter retrieval times but can cause poor quality objects to be retrieved.

Initially, when no bandwidth estimates are available, `BigEye` obtains an initial estimate of the bandwidth between each producer and the summarizer, using the *iperf* utility [23]. Next, the algorithm updates this bandwidth estimate during execution as follows. It considers a fixed period of time (denoted by T), and the summarizer retrieves the maximum number of multimedia objects from each producer, which are retrievable within this period. Partially retrieved objects (complete retrieval not possible in T) are not counted. Based on the objects retrieved and their sizes, from a producer, the summarizer estimates the bandwidth to that producer (data retrieved divided by T).

At any given point, the summarizer has the sorted objects as described earlier, and the bandwidths to each of the producers. It then tries to pull up an object from producers in order from the ranked list. For each object, the summarizer first checks if the object is still relevant (meaning that the necessary number of objects have already been received for the corresponding event). If not, the object is discarded. Otherwise, it checks if the retrieval of the object violates the bandwidth constraints to the corresponding producer (i.e., can the object be retrieved within T seconds, given the estimate of the bandwidth to that producer). If there is no violation, the object is retrieved from the producer and the bandwidth to that producer is decreased by the size of the object. In particular, if the estimate of the available bandwidth to producer P_i was B_i and the object size was w_j, $B-i$ is updated to $B_i = B_i - w_j$. Furthermore, the number of objects that is needed for the summary corresponding to that event (say e) is decreased by 1 (i.e., $C_e = C_e - 1$).

For each new time period, the producer's bandwidths are implicitly updated during the above process. The summarizer repeats these steps until the total number of required objects is retrieved for all events ($\sum_e C_e = 0$). The procedure is formally summarized in Algorithm 1.

VII. Implementation and Evaluations

The implementation and experimentation environments are described first in this section. Subsequently, our evaluations of `BigEye` are provided. Each module is evaluated separately to showcase the benefits of each. We consider the holistic performance of our system at the end of the section. To emulate a network, we use Mininet [24], a popular software defined network (SDN) emulator. To show the realism of our

Algorithm 1: Multimedia Objects Retrieval

Function ***Calibrate*** *(ReceivedObjects[i])*
  Bandwidth[i] $\leftarrow \sum w_j, j \in$ *ReceivedObjects*[*i*]
  BestBw[i] $\leftarrow$ Bandwidth[i]
end
for *i in* ***I*** **do**
  MaxBW[i] $\leftarrow$ *iperf* (P_i, summarizer)
end
Bandwidth $\leftarrow$ MaxBw
BestBw $\leftarrow$ Bandwidth
objects $\leftarrow$ sorted (objects, $\frac{1+Lf_j}{w_j}$)
while $\sum_e C_e \neq 0$ **do**
  forall *j in objects* **do**
    $i \leftarrow j \in P_i$
    if *Bandwidth[i]-* $w_j \geq 0$ & C_e*- 1* ≥ 0 **then**
      Bandwidth[i]= Bandwidth[i]- w_j
      $C_e = C_e$ - 1
      RequestObject(j, P_i)
    end
  end
  wait (timeWindow)
  Bandwidth $\leftarrow$ BestBw
  for *i in* ***I*** **do**
    if *connection(summarizer,*P_i*) intercepted* **then**
      Calibrate(ReceivedObjects[i])
    end
  end
  if *set of transmitting producers changes* **then**
    Bandwidth $\leftarrow$ MaxBw
  end
end

approach, we use the NDN test-bed topology to represent the network [25]. This topology is used in a real deployment where distributed producers are connected. We change the default latencies and bandwidth per link in some experiments to showcase and demonstrate the scenarios of interest (bandwidth constrained environment).

We use the ONOS SDN controller [26] to manage the communication and routing between producers and the summarizer (and *vice versa*). Each producer receives microblogs (e.g., tweets) of interest, and by using `BigEye`, the necessary computations and communications are performed at both the producer and summarizer sides, as described in the aforementioned three stages of the system. Finally, `BigEye` outputs a summary of the global events of interest plus a concise visual summary (the holistic output of the approach is discussed in Section VII-G).

A. Data Set Collection and Distribution

We collect tweets using Apollo [27], a framework that retrieves information from Twitter using Twitter API. The framework allows its users to collect tweets that match keywords of interest such as "disaster" and "wild fires." The collected data sets are cleaned by removing: 1) retweets; 2) stop words and special characters [28]; and 3) URLs. Subsequently, we apply stemming [29] to the collected data. We note that this is a common practice in data mining applications. We use the Python-NLTK tokenizer [30] and the Porter stemmer [29], which are common tools used for these purposes. The collected data sets are summarized in the following:

1) *Protest:* This data set is collected using the keyword "protest." The data set has tweets related to protests and it was collected from *March 18, 2018* to *April 18, 2018*. It consists of approximately 300K tweets after applying the aforementioned filtering methods.
2) *Florence:* The data set contains information about the Hurricane that occurred in the Carolinas in 2018. The data set was collected using the keywords "hurricane" and "florence" from September 14, 2018 to September 24, 2018. It contains approximately 100K tweets after conducting the aforementioned filtering methods.

To emulate a scenario in real time, the data are streamed and fed to the producers with respect to the time stamps collected from the tweets. For the purpose of experimentation in this article, we choose a window size of 24 h. The term *instance* refer to the data corresponding to each such window size. Thus, for the above data sets, we have 40 instances in total. The data sets are distributed over multiple producers with two different scenarios that are described below.

1) Natural Distribution: In practice, the tweets are posted by Twitter users from different geographical locations. One can use those locations to cluster the tweets into different geographical zones, which can be later associated with the producers. However, not all tweets contain associated geolocation information; in fact less than 2% of tweets have this information [31]. This makes the problem of simulating exact real-word geographical distribution of tweets hard. So, the addresses of users (which can be obtained from users profiles) are used to mimic the geographical distribution of tweets (tweets from the same user are assumed to be made from his/her profile's geolocation information). Using this approach, we are able to retrieve 70% and 60% of this information, from users associated with the Florence and Protest data sets, respectively. We are unable to decipher the rest because some users had provided vague addresses (e.g., "space" and "floor"). We use an API provided by *HERE.com* to convert the provided addresses to geolocations (composed of latitude and longitude) [32].

As mentioned earlier, we used the NDN topology, which reflects a real-world network (real server locations); we assume each tweet is sent to the nearest physical producer (assuming that each server is a producer). Fig. 3 shows an example demonstrating the approach. We extend the same idea when a variable number of producers (different from the original number of servers in the NDN-topology) is considered, from as follows. First, each tweet is assigned to its nearest producer and among all producers we select the top m (ones that receive the most number of tweets) producers. Next, we assign tweets to the nearest producer from these top selected m producers. For tweets where we could not retrieve the geographical locations, they are assigned randomly to a producer.

2) Synthetic Distribution: To further evaluate the performance of `BigEye`, we consider the uniform and skewed distribution of tweets across multiple producers.

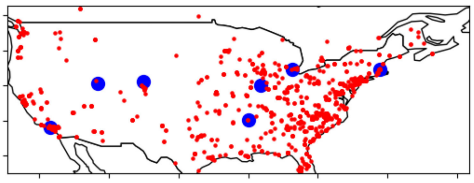

Fig. 3. Map of the United States, representing two hours of tweets collected from the Florence data set. Blue dots represent producers and red dots represent the locations of tweets. Each tweet (red dot) is sent the nearest producer (blue dot).

1) *Skewed Distribution:* The distribution of the data over the producers follows a Gaussian distribution with $\mu = (N_k/m)$ and $\sigma^2 = \beta * \mu$; we vary the β to control the skew.
2) *Uniform Distribution:* The distribution of the data over the producers follows a uniform distribution.

We also collect the images associated with the tweets and stream them on the producers according to the selected distribution. For tweets scores, we select three features (f_j as discussed in Section VI): 1) retweet count; 2) favorite count; and 3) follow ratio defined as (# of followers/# of following).[2] We also select L to be 1. Twitter allows users to post more than one multimedia object in a single tweet. We treat each as an independent object while giving each the same score as their associated tweet. Twitter API keeps multiple versions of the videos each with different quality. To avoid redundancy, we pick only a particular version randomly. We note that the higher the quality of a video or image, the higher its size.

B. Evaluation Parameters

Our evaluations of the performance of BigEye consist of a comparison with an approach wherein all data are made available centrally. Our evaluations are on the data sets that were discussed in the prior section. We stream each data set separately, in line with our assumption (streaming data are aligned with the scope of a realistic scenario) in Section III. The results that we obtain are then integrated before we discuss them for two reasons. First, we find that the performance with BigEye is consistent across both data sets, when the data are distributed over geographically separated producers, compared to when the data are centrally available. Second, the generality of BigEye ensures that its behavior with any crowdsensed data set, belonging to a particular scope, is consistent. As a consequence of these two reasons, an independent discussion of the details of the performance of each data set provides no new information, and is wasteful of space.

Our experiments are performed with all instances from all of the data sets (40 in total), considered one at a time. For statistical significance, experiments are repeated 50 times. BigEye's three components are integrated holistically; however, first we independently evaluate each component to provide microscopic views of the benefits of each. The following describe for these module specific set ups.

[2]Assessing how good the selected features are in practice is beyond the scope of this article.

Distributed Event Detection Module: As described in Section IV, we find that our method always yields the same precision and recall with respect to detection of key events, as that of an ideal system which considers that all data can be made available centrally. We experimentally validated this and do not further showcase the event detection accuracy in the interest of space.

To evaluate if the number of keyword pairs that are returned by BigEye to the summarizer is reasonable, we compare the number with the best case scenario. In particular, we consider an *oracle* that does a brute force search, considering all possible subsets of the keywords pairs sent (the ranked orders of those pairs are still maintained), and checks if any of those subsets yields the same precision and recall as our approach (and the central approach). We choose the smallest subset among these as the best possible scenario (we label it as oracle prediction in the results that we present). In other words, instead of choosing a threshold (r^*/m), we find a the smallest value $l \leq m$ such that choosing $(r * /l)$ results in the detection of all events of interest.

Consolidation of Events: Our proposed distributed consolidation is evaluated next. Specifically, BigEye's approach is compared with a case all the data associated with the clusters is sent to the summarizer which then applies a consolidation. We denote the baseline as *central consolidation*. We point out that this is the consolidation used in the prior work Storyline [4], where the data are centrally made available and similarity assessment is based on Jaccard distance.

The metrics of interest are accuracy (defined next) and the amount of data sent from the producers to summarizer for the purposes of consolidation. Accuracy is defined to be the ratio of the number of keyword pairs that are grouped correctly (the events are correctly consolidated) to the total number of keyword pairs. Two keyword pairs are incorrectly grouped if: 1) these pairs belong to the same event but are put in different groups and 2) they belong to different groups (events) but are consolidated into the same group. We also compare the amount of sent data from the producers to summarizer with our approach (in bytes), with the other approach.

Composition of Visual Summaries: We assess the "quality" from BigEye's visual summary module with respect to the centralized case (where no bandwidth constraints are present). In particular, if data are located centrally, the summary can be composed using the objects that have the highest associated scores. However, objects with the highest scores are not always retrieved in BigEye to meet timeliness constraints as discussed in Section VI. To evaluate the impact of this, we evaluate the quality which is defined as the ratio of the summation of the scores of the multimedia objects retrieved with BigEye, to the corresponding sum score achieved in the centralized case.

We also evaluate the delay in retrieving the visual objects. Formally, this delay is defined as the time it takes for the summarizer to receive the data used for the visual summary from the producers. Specifically, the metric we use is the average delay incurred in receiving 1 MB of data (we normalize this since the sizes of the visual summaries could be different for different events). We compare the delay of BigEye's

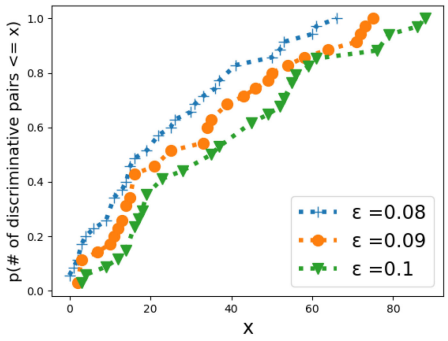

Fig. 4. CDF of the number of detected discriminative pairs with different global thresholds.

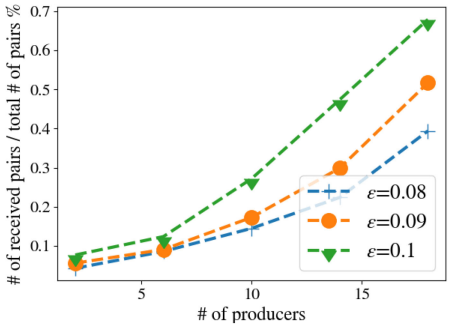

Fig. 5. Performance of our distributed event detection with varying number of producers.

visual summary module with the following baseline (denoted as *baseline*). Each producer is assigned the job of sending information related to a specific event. If the number of events are more than the number of producers, the events are evenly distributed across the producers. If a producer is assigned more than an event, the resources are shared equally between them with no priority given to one over the other. Our goal here is to showcase the efficacy of our parallelization approach with a second approach (the baseline) that also parallelizes transfers but is not bandwidth aware (does not take into account the different bandwidths to the different producers).

C. Results on Distributed Event Detection

First, we evaluate our distributed event detection module using the aforementioned metrics. We recall our discussion in Section IV [$H(Y|s_z)$ was to be a small value ϵ], and select ϵ to be 0.08, 0.09, and 0.1 ($r^* = 100$, 87, and 76, respectively). Here, we also refer the reader to Table I since the notation therein is used in the discussion.

Effect of r^* on the Total Number of Pairs Retrieved by the Summarizer: We plot the CDFs of the number of events (corresponding to identified discriminative key word pairs) detected with each value of r^* in Fig. 4. As one might expect, as r^* increases, the number of pairs retrieved decreases (with higher r^* only the most significant events are detected). This effect is also seen in Fig. 5.

Effect of Increasing Number of Producers: In Fig. 5, we plot the ratio of the number of retrieved keyword pairs to the total number of keyword pairs identified, versus the number

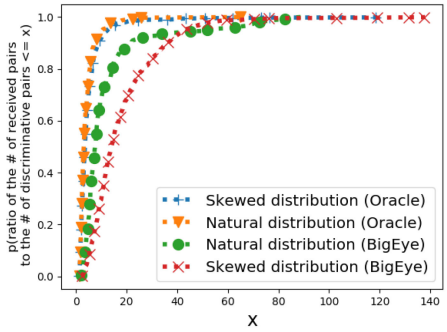

Fig. 6. Performance of our distributed event detection with different data distributions.

of producers. We assume that the data are distributed as per the *natural* distribution. As one might expect, the number of key-words pairs returned to the summarizer increases when as the number of producers, m, increases. This is because (r^*/m) decreases, i.e., a lower or more conservative (local) threshold is used at each of the producers. It is worth noting that with small $r^* = 76$, $(H(Y|s_z) = 0.1)$ and large $m = 18$, the total number of received pairs is less than 1% of the total number of keyword pairs considered globally.

Comparison With Oracle: Next we examine how the number of keyword pairs retrieved with `BigEye` compares to what is obtained by an oracle, when the data are spread across the producers as per the different data distributions (discussed in Section VII-A). For the skewed distribution, we choose $\mu = (N_k/m)$ and $\sigma^2 = 0.5 * \mu$; this ensures a high skew. We fix $\epsilon = 0.09$ ($r^* = 87$), and m to be 10.

In Fig. 6, we plot the CDF of the ratio of the number of pairs received at the summarizer to the number of global discriminative pairs with both `BigEye` and the oracle-based approach described earlier. We see that the performance of `BigEye` is similar to that of the *oracle* when data are dispersed as per the natural distribution. However, when the distribution is skewed, the performance of the `BigEye` degrades compared to the oracle. This is because the producers with large numbers of tweets have a large number of keyword pairs that pass the conservative threshold selected by `BigEye`; thus, they end up sending a large number of pairs that are not useful in detecting key events.

D. Results With Regards to Consolidation

Next, we evaluate the benefits from `BigEye`'s consolidation module. In our evaluations we use the same ϵ values mentioned earlier in Section VII-C. Recall that `BigEye` consolidates events over two steps. In the first step, we consider a similarity requirement of 0.99 (Jacquard distance) to consolidate events at individual producers. For the second step, we consider three minHash signatures of length 64, 128, and 256 integers. We consider different minHash signatures as it has been reported that the length of generated minHash signatures affects the similarity scores [19]. We also vary the consolidation thresholds that are used centrally from 0.4 to 0.9 with a stepsize of 0.1.

In Fig. 7, we compare `BigEye`'s consolidation approach with different minHash lengths. We observe that the similarity

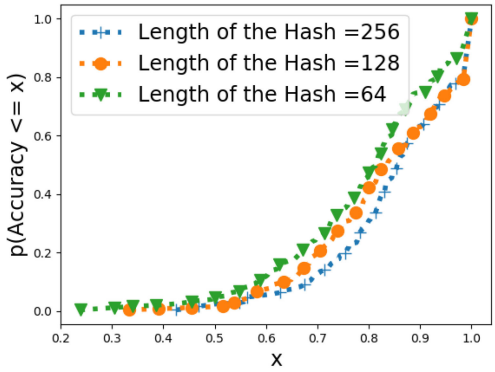

Fig. 7. Distributed consolidation accuracy with respect to centralized consolidation (Storyline [4]).

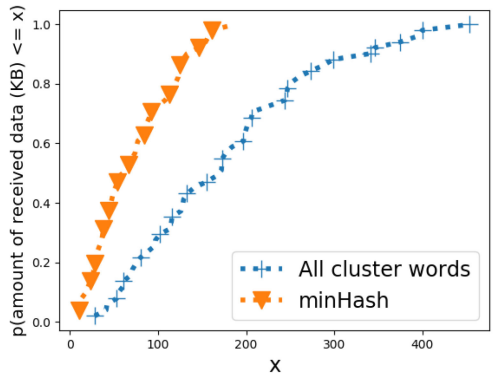

Fig. 8. Bandwidth savings from BigEye's distributed consolidation approach in terms of total amount of data sent from producers to summarizer.

estimation improves as the length of the minHash signature increases (less collisions). Similar phenomena have also been reported in the previous literature [19]. We observe that for a minHash signature of length 128, almost 85% of instances show a consolidation accuracy of larger than 70% of what is achieved if all data were available centrally. BigEye provides consistent consolidation accuracies irrespective of the number of producers.

We show next the consolidation benefits in terms of the reduction in the volume of data sent from producers to summarizer in bytes, compared to sending all keywords in the clusters (needed for central consolidation); Fig. 8 shows that our approach reduces the communication costs significantly compared to that baseline approach, and in particular the average cost by 60% (fewer bytes).

E. Results on Visual Content Retrieval

Next, we evaluate the performance of BigEye's visual summary module with respect to the baseline approach discussed in Section VII-B.

Delay Performance With Different Numbers of Producers: We evaluate the performance of BigEye's final module with different numbers of producers (we consider two, six, and ten producers). We assume that the data are distributed uniformly over these producers. We plot the CDFs of the average delay (over the instances) with BigEye and the baseline in Fig. 9. First, we observe that our approach outperforms the

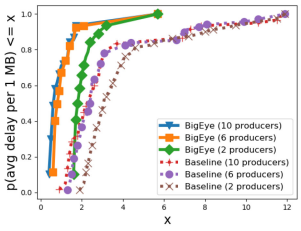

Fig. 9. Delay in visual content retrieval with different number of producers.

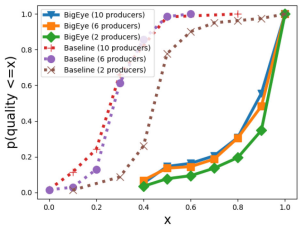

Fig. 10. Summarization quality of visual summary with different number of producers.

baseline by approximately 67% on average. Furthermore, we also see that the delay decreases as the number of producers increases initially. This is because of the increased parallelization in retrieval that is possible due to this. However, the benefits reach a point of diminishing gains. Specifically when the number of producers increases beyond a certain number (e.g., above 6 and 10 in Fig. 9) because of limitations in the NDN network structure very little additional parallelization is possible (the producers share common paths). We have manually constructed a tree network where each producer is connected to the summarizer with a dedicated link. In this case, we do observe significant performance enhancements between the cases of six and ten producers since parallelization is now viable. We omit showing these results to conserve space.

Impact on Summary Quality: We plot the CDF of the quality of the visual summaries obtained with different numbers of producers versus the baseline approach in Fig. 10. We see that the best quality is achieved when there are only two producers. Typically, if all data are at a single producer, it is natural to pick the objects with highest scores (f_j/w_j) according to the global sorted list shown in Algorithm 1. However, when the number of producers increases and we try to achieve parallelization of transfers, the highest ranked objects in the global list are not always retrieved. In particular, if a single producer has most of the highest ranked objects, we will retrieve objects of inferior quality from other producers. In other words, the summarizer pulls other objects (with lower rank) from other producers, which, in turn, affects the achieved quality (the tradeoff is the reduction in terms of the delay).

Data Requirement to Compose Summaries: Next, we evaluate the performance of BigEye when the user imposes different requirements on the number of objects that are needed in the visual summary. Specifically, we impose requirements of (25%, 50%, and 75%) of the total number of multimedia objects (corresponding to each detected event) to compose the summary. We use the skewed data distribution with $\beta = 1$ (very skewed distribution). This means that a small

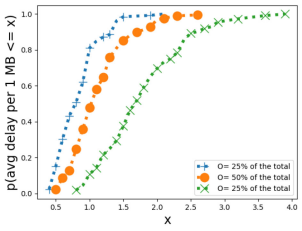

Fig. 11. Delay in visual content retrieval with different number of objects required to compose summary.

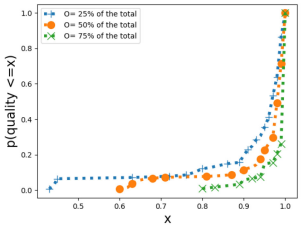

Fig. 12. Summarization quality of visual summary with different numbers of required objects.

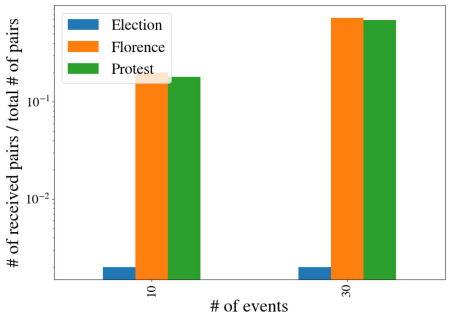

Fig. 13. Performance of the distributed event detection module with multiple data sets (scalable setting).

subset of the producers has a large number of multimedia objects, while others have only few. We fix the number of producers to be 6. As shown in Fig. 11, when the required number of objects is only 25% of the total, the lowest delay is achieved. As evident, as this requirement increases, the delay increases. In fact, because of the skewed distribution, when we need 75% of the objects, the summarizer is forced to abandon parallelization and retrieve all objects from the producers with the large number of objects. This drastically affects the delay.

Fig. 12 shows, as one might expect, that the quality of the composed summary improves as the number of required objects to compose the summary increases. However, this improvement is not drastic. This is because the first objects that are retrieved have the highest scores; later objects contribute less and less to the quality of the summary.

F. Scalability

To showcase BigEye's scalability where the generated data are much larger, we use a data set collected on election day in the United States. It consists of around six million tweets and we clean it up as we discussed earlier in Section VII-A. We point out here that the data sets used in the results shown earlier (Florence and Protest) are small in comparison to the

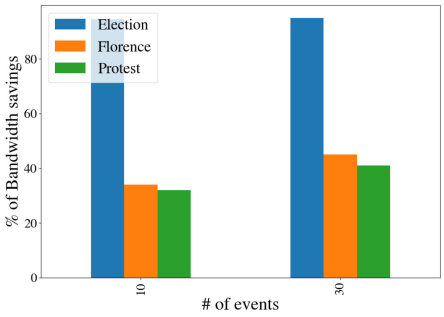

Fig. 14. Bandwidth savings from BigEye's distributed consolidation with multiple data sets (scalable setting).

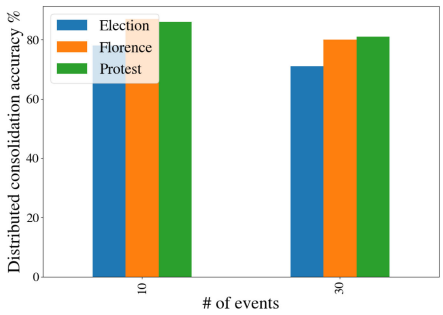

Fig. 15. BigEye's distributed consolidation accuracy with multiple data sets (scalable setting).

Election data set (by a factor of about ≈ 20). We select global thresholds such that an approximate constant number of top events (10 and 30) are retrieved, and assess the performance of our modules. Specifically, we examine how the results scale with the data set size (in comparison to the smaller ones). We point out here that tuning the global threshold to get the exact same number of events across all data sets is hard; thus we choose thresholds such that the ratio of events retrieved across each pair of data sets varies between 0.9 and 1.1 (i.e., the numbers are approximately 10 or 30 but not exact). We select the number of producers m to be 10 and a hash of length 128. The accuracy of our distributed event detection module remains at 100% like in the case where all the data are available centrally (similar to the results reported earlier) even when the data set is large. The ratio of the received keyword pairs (after processing by BigEye) to the total number of keyword pairs is drastically smaller with the Election data set (only 0.002%) in comparison with the Florence hurricane and Protest data sets as shown in Fig. 13. One might expect this to be the case since the total number of keyword pairs is much larger in comparison to the candidate discriminative keyword pairs [with a local ratio of occurrence larger than (r^*/m), the local threshold]; recall that only these candidates are sent from the producers to the central controller. In large data sets (e.g., the Election data set), the total number of keyword pairs is much larger than the corresponding number in smaller data sets (e.g., Florence); since we choose a target number of top events to be detected, the ratio of candidate discriminative key word pairs to the total number of keyword pairs becomes drastically smaller. Thus, the takeaway is that one can expect even further reductions in overhead with BigEye as the data set sizes grow, because of large

TABLE II
FLORENCE HURRICANE SUMMARIZATION

Date	Detected keywords	Textual summary
Sept 15	**Event I:** sympathy, hurricane, deepest, hurricane	”My deepest thoughts and prayers are with those in North and South Carolina, and Virginia affected by Hurricane Florence”
Sept 16	**Event I:** ’abandon’, ’hero’, ’six’, ’dog’	“Six dogs have been rescued from rising flood waters, after they were locked in a cage and abandoned by their owners”
	Event II: ’dog’, ’64’, ’rescue’, ’hurricane’	“As Florence loomed, a pet lover escaped South Carolina with 64 dogs and cats on a school bus”

TABLE III
PROTEST SUMMARIZATION

Date	Detected keywords	Textual summary
Mar 29	**Event I:** ’viral’, ’plung’, ’california’, ’cliff’	”Washington state family famed for protest photo died when SUV goes off California cliff”
	Event II: ’NBA’, ’office’	“Police, DSS Invade Ikeja NBA Office To Foil Protest Coinciding With Buhari’s Visit”
Mar 30	**Event I:** ’bill’, ’school’, ’teacher’, ’kentucki’	“Kentucky schools close as teachers protest GOP-passed pension overhaul”
	Event II: ’gaza’, ’border’, ’palestinian’, ’kill’, ’israel’	“7 Palestinians killed, dozens injured as Israel suppresses massive protest in Gaza”
Mar 31	**Event 1:** ’Stephon’ , ’clark’, ’car’, ’hit’	“Sheriff’s Car Knocks Down Activist During Stephon Clark Shooting Protest”
	Event II: ’student’ ’Howard’, ’protest’	“Howard University students stage sit-in to protest financial aid scandal”
		”Rihanna’s “Bitch Better Have My Money” becomes protest anthem for Howard University students”
	Event III: ’deadly’, ’day’, ’15’, ’palestinian’, ’kill’, ’israel’	“15 Palestinians reported killed by Israeli fire as Gaza border protest builds”

reductions (if we seek to detect a certain target number of top events).

We next evaluate our second module (consolidation) with a large data set in terms of both bandwidth savings and consolidation accuracy. With the new larger data set, the number of words that mapped on to each event increases significantly; thus, `BigEye`’s approach of sending a fixed hash representation of the event contents (as compared to sending the raw words corresponding to the event) reduces the overall proportion of transmitted data in comparison to the smaller data sets as shown in Fig. 14. However, as shown in Fig. 15, we see that with the Election data set, we have a slightly lower consolidation accuracy in comparison with the results obtained with the other data sets. This is because the used hash length is small (only 128) and, hence, it is not sufficient to capture the similarity between the detected events in this large data set. As indicated earlier, a possible way to enhance the consolidation accuracy is to increase the hash length (see Fig. 7). This is because the similarity estimation improves as the length of the hash increases [19]. To verify that this is applicable on large data sets, we increase the hash length to 256 and run `BigEye` with the Election data set and we achieve consolidation accuracy of 90% when the number of events equal to 30.

G. Holistic Output of BigEye

Next, we show the holistic output of `BigEye`. We select random instances from three data sets. We use the previously mentioned Protest and Florence data sets, and a new data set (called the disaster data set [33]) that is available for public use to further show the merits of our proposed framework. The summaries are composed of: 1) the discriminative key word pairs; 2) textual summaries (complete tweets) describing the observed events; and 3) the visual summaries (we select the best images to avoid cluttering). The discriminative pairs and the textual summaries of the three data sets are shown in Tables II–IV, and the corresponding visual summaries are shown in Figs. 16–18. To elaborate, we will go through a sample of the results associated with Protest data set. On March 30,

TABLE IV
DISASTER SUMMARIZATION

Date	Detected keywords	Textual summary
Jan 14	**Event I:**'Australia', 'forest' 'fire'	"13 years ago we predicted that the worst fire seasons would be directly observable in australia by the year 2020"
	Event II: 'volcano', 'Philippine', 'Taal', 'erupt'	"Taal volcano, located 70 km from Manila, Philippines. In addition to the eruption itself, the so-called volcanic tsunami"

Event I on Sept 15

Event I (left) and Event II (right) on Sept 16

Fig. 16. Florence data set.

Event I (left) and Event II (right) on March 29

Event I (left) and Event II (right) on March 30

Event I on March 31

Event II on March 31

Event III on March 31

Fig. 17. Protest data set.

two major protests were detected. The first related to teachers in Kentucky schools, who were protesting against a pension bill which forced schools to close (an image of the protest is shown). The second event is related to a protest in Gaza where seven people were killed and dozens were injured. One can easily understand and differentiate between these by following the textual summaries of the events and their corresponding visual summaries.

VIII. RELATED WORK

Postings on social networks have been recently used as sensor outputs [34], [35] that can be used to discern events. There exist some prior studies on detecting events from such sensors' data (with goals aligned with those of `BigEye`). Allan *et al.* [36] used the term frequency (tf) and inverse document frequency (idf) features to build a query representation for content from news stories and identified an event, when the similarity score of new news story was less than a given threshold in comparison to any previous news query in memory. Similarly, Shamma *et al.* [37] used a normalized tf to identify peaky topics, the terms which are particular to a time window, to detect highly localized events of interest. Benhardus and Kalita [38] also used tf–idf analysis and relative normalized tf analysis, on twitter documents to identify trending topics. However, these approaches were

Event I on Jan 14

Event II on Jan 14

Fig. 18. Disaster data set.

reported to be inefficient in differentiating between separate event instances [4]. Moreover, unlike tf–idf, BigEye works by only computing information gain over two consecutive time windows.

Text stream clustering has also been applied for event detection. Ordonez [39], Zhong [40], and Aggarwal and Yu [41], used optimizations of *k*-means algorithms to cluster data streams for events detection. Similarly, communication patterns [42], social network topological features [43], language specific features [44]–[46], and location of tweets [47]–[49] have also been used by researchers for clustering data to detect events. Nevertheless, precisely defining the number of clusters (k) for online streaming data is not feasible. Researchers have also used topic modeling for event detection [50]–[52]. However, topic-based approaches have been reported to be inefficient in identifying events happening in parallel instances [4]. Unlike these methods, BigEye detects events by measuring the temporal bursts in the word-pairs that do not co-occur frequently. BigEye's event detection approach is closely related to Storyline that was proposed by Wang *et al.* [4]. However, unlike BigEye, Wang *et al.* only focused on event detection when data are centrally located.

A different line of work considers the problem of truth finding in social sensing blogs [53], [54]; however, these only work when all the data are made available centrally unlike BigEye. In contrast, BigEye is targeted for a distributed setting, wherein the data are distributed across multiple producers that are geographically separate.

BigEye centers around the detection of global events by only sharing minimal amounts of information between distributed producers and a central summarizer. There have been some prior works on selectively sending information to a central entity [55]–[57]. There are also related works focusing on calibrating distributed sensors with the objective of finding global measurements from those sensors (e.g., measuring urban air pollution) [58], [59]. However, unlike BigEye, these approaches do not focus on event detection. Closely relevant to our study is the study by McCreadie *et al.* [60]. Unlike BigEye, they do not consider bandwidth constraints and only try to minimize the event detection time by distributing the computational costs of processing documents across multiple machines.

IX. Conclusion

In this article, we addressed the important problem of detecting global events from crowdsensed data. Toward this, we designed and implemented BigEye, a system that enables: 1) the detection of key global events based on distributed crowdsensed data that exist at geographically spread out producers and 2) the crafting of visual summaries that provide concise zoomed-in views of such events. BigEye distinguishes itself in that it is extremely thrifty in terms of the bandwidth that it consumes, i.e., very little of the raw crowdsensed data from the producers needs to be transferred to a central entity for both event detection and the subsequent visual summarization. Despite its thriftiness, it is able to achieve 100% precision and recall compared to approaches, where all crowdsensed data are made available centrally. Via emulations of realistic scenarios, we showed that BigEye only consumes 1% of the crowdsensed data for detecting key global events, and its parallelization of visual content retrieval reduces the average delay by 67% compared to baseline approaches.

Acknowledgment

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 DEVCOM Army Research Laboratory 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] D. Wang, T. Abdelzaher, and L. Kaplan, *Social Sensing: Building Reliable Systems on Unreliable Data*. Waltham, MA, USA: Morgan Kaufmann, 2015.

[2] B. Guo *et al.*, "Mobile crowd sensing and computing: The review of an emerging human-powered sensing paradigm," *ACM Comput. Surveys.*, vol. 48, no. 1, p. 7, 2015.

[3] M. I. Ali *et al.*, "Real-time data analytics and event detection for IoT-enabled communication systems," *J. Web Semantics*, vol. 42, pp. 19–37, Jan. 2017.

[4] S. Wang *et al.*, "StoryLine: Unsupervised geo-event demultiplexing in social spaces without location information," in *Proc. 2nd Int. Conf. Internet Things Des. Implement.*, 2017, pp. 83–93.

[5] "Harvey Victims are Using Social Media When 911 Fails." Nypost. [Online]. Available: https://nypost.com/2017/08/28/harvey-victims-are-using-social-media-when-911-fails/ (accessed Sep. 2018).

[6] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[7] "Twitter Adding More Data Center Space (Again)." DataCenter Knowledge. Sep. 2011. [Online]. Available: https://www.datacenterknowledge.com/archives/2011/09/19/twitter-adding-more-data-center-space-again (accessed Oct. 11, 2018).

[8] "Moscow Says Twitter Ready to Store Data of Users on Russian Servers Despite Concerns Over Surveillance." Telegraph. Nov. 2017. [Online]. Available: https://www.telegraph.co.uk/news/2017/11/08/moscow-says-twitter-ready-store-data-users-russian-servers-despite/ (accessed Oct. 11, 2018).

[9] "Tropical Storm Harvey Takes out 911 Centers, Cell Towers, and Cable Networks." Ars Technica. [Online]. Available: https://tinyurl.com/yu6ven52 (accessed Aug. 2018).

[10] A. Fahim, A. Neupane, E. Papalexakis, L. Kaplan, S. V. Krishnamurthy, and T. Abdelzaher, "Edge-assisted detection and summarization of key global events from distributed crowd-sensed data," in *Proc. IEEE Int. Conf. Cloud Eng. (IC2E)*, Prague, Czech Republic, 2019, pp. 76–85.

[11] S. Hanafi and A. Freville, "An efficient tabu search approach for the 0–1 multidimensional knapsack problem," *Eur. J. Oper. Res.*, vol. 106, nos. 2–3, pp. 659–675, 1998.

[12] Y. Yang and J. O. Pedersen, "A comparative study on feature selection in text categorization," in *Proc. ICML*, vol. 97, 1997, pp. 412–420.

[13] D. Mills *et al.*, "Network time protocol," IETF, RFC 958, 1985.

[14] D. G. Luenberger and Y. Ye, *Linear and Nonlinear Programming*, vol. 116. New York, NY, USA: Springer, 2008.

[15] S. Niwattanakul, J. Singthongchai, E. Naenudorn, and S. Wanapu, "Using of Jaccard coefficient for keywords similarity," in *Proc. Int. MultiConf. Eng. Comput. Sci.*, vol. 1, 2013, pp. 1–5.

[16] W. Feng *et al.*, "STREAMCUBE: Hierarchical spatio-temporal hashtag clustering for event exploration over the Twitter stream," in *Proc. IEEE 31st Int. Conf. Data Eng.*, Seoul, South Korea, 2015, pp. 1561–1572.

[17] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher, "Min-wise independent permutations," *J. Comput. Syst. Sci.*, vol. 60, no. 3, pp. 630–659, 2000.

[18] T. Haveliwala, A. Gionis, and P. Indyk, "Scalable techniques for clustering the Web (extended abstract)," in *Proc. 3rd Int. Workshop Web Databases (WebDB)*, 2000. [Online]. Available: http://ilpubs.stanford.edu:8090/445/

[19] J. Leskovec, A. Rajaraman, and J. D. Ullman, *Finding Similar Items*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 2014, pp. 68–122.

[20] A. Arasu, V. Ganti, and R. Kaushik, "Efficient exact set-similarity joins," in *Proc. 32nd Int. Conf. Very Large Data Bases*, 2006, pp. 918–929.

[21] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack Problems*. Berlin, Germany: Springer, 2004.

[22] Y. Disser, M. Klimm, N. Megow, and S. Stiller, "Packing a knapsack of unknown capacity," *SIAM J. Discrete Math.*, vol. 31, no. 3, pp. 1477–1497, 2017.

[23] J. Ahrenholz, C. Danilov, T. R. Henderson, and J. H. Kim, "CORE: A real-time network emulator," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, San Diego, CA, USA, 2008, pp. 1–7.

[24] "Mininet." Mininet.org. [Online]. Available: http://mininet.org/ (accessed Jun. 2018).

[25] "Named Data Networking." NDN-Testbed. [Online]. Available: https://named-data.net/ndn-testbed/ (accessed Jun. 2018).

[26] "ONOS SDN." ONOS Project. [Online]. Available: https://onosproject.org/ (accessed Jun. 2018).

[27] "Moscow Says Twitter Ready to Store Data of Users on Russian Servers Despite Concerns Over Surveillance." Apollo. Nov. 2014. [Online]. Available: http://apollo2.cs.illinois.edu/index.html (accessed Oct. 11, 2018).

[28] A. Rajaraman and J. D. Ullman, *Data Mining*. Cambridge, U.K.: Cambridge Univ. Press, 2011, pp. 1–17.

[29] M. Porter. "The Porter Stemming Algorithm." Jan. 2016. [Online]. Available: https://tartarus.org/martin/PorterStemmer/ (accessed Oct. 11, 2018).

[30] "NLTK 3.3 Documentation." NLTK Tokenizer. [Online]. Available: https://www.nltk.org/api/nltk.tokenize.html (accessed Oct. 11, 2018).

[31] Z. Cheng, J. Caverlee, and K. Lee, "You are where you tweet: A content-based approach to geo-locating Twitter users," in *Proc. 19th ACM Int. Conf. Inf. Knowl. Manag.*, 2010, pp. 759–768.

[32] "Location Data Processing." Here.com. [Online]. Available: https://www.here.com/ (accessed Jun. 2018).

[33] Jan. 2021, "Disaster Tweets Dataset," Kaggle. Accessed: Jan. 5, 2021. [Online]. Available: https://www.kaggle.com/vstepanenko/disaster-tweets

[34] D. Wang *et al.*, "Using humans as sensors: An estimation-theoretic perspective," in *Proc. 13th Int. Symp. Inf. Process. Sens. Netw. (IPSN)*, Berlin, Germany, 2014, pp. 35–46.

[35] D. Wang, L. Kaplan, H. Le, and T. Abdelzaher, "On truth discovery in social sensing: A maximum likelihood estimation approach," in *Proc. 11th Int. Conf. Inf. Process. Sens. Netw.*, Beijing, China, 2012, pp. 233–244.

[36] J. Allan, R. Papka, and V. Lavrenko, "On-line new event detection and tracking," *ACM SIGIR Forum*, vol. 51, no. 2, pp. 185–193, 2017.

[37] D. A. Shamma, L. Kennedy, and E. F. Churchill, "Peaks and persistence: Modeling the shape of microblog conversations," in *Proc. ACM Conf. Comput. Supported Cooperative Work*, 2011, pp. 355–358.

[38] J. Benhardus and J. Kalita, "Streaming trend detection in Twitter," *Int. J. Web Based Communities*, vol. 9, no. 1, pp. 122–139, 2013.

[39] C. Ordonez, "Clustering binary data streams with K-means," in *Proc. 8th ACM SIGMOD Workshop Res. Issues Data Min. Knowl. Discov.*, 2003, pp. 12–19.

[40] S. Zhong, "Efficient streaming text clustering," *Neural Netw.*, vol. 18, nos. 5–6, pp. 790–798, 2005.

[41] C. C. Aggarwal and P. S. Yu, "A framework for clustering massive text and categorical data streams," in *Proc. SIAM Int. Conf. Data Min.*, 2006, pp. 479–483.

[42] F. Chierichetti, J. M. Kleinberg, R. Kumar, M. Mahdian, and S. Pandey, "Event detection via communication pattern analysis," in *Proc. ICWSM*, 2014, pp. 1–10.

[43] C. C. Aggarwal and K. Subbian, "Event detection in social streams," in *Proc. SIAM Int. Conf. Data Min.*, 2012, pp. 624–635.

[44] P. Giridhar, S. Wang, T. F. Abdelzaher, J. George, L. Kaplan, and R. Ganti, "Joint localization of events and sources in social networks," in *Proc. IEEE Int. Conf. Distrib. Comput. Sens. Syst. (DCOSS)*, 2015, pp. 179–188.

[45] I. Tien, A. Musaev, D. Benas, A. Ghadi, S. Goodman, and C. Pu, "Detection of damage and failure events of critical public infrastructure using social sensor big data," in *Proc. IoTBD*, 2016, pp. 435–440.

[46] 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 *Proc. 20th ACM Int. Conf. Inf. Knowl. Manag.*, 2011, pp. 2541–2544.

[47] A. Boettcher and D. Lee, "EventRadar: A real-time local event detection scheme using Twitter stream," in *Proc. IEEE Int. Conf. Green Comput. Commun.*, Besancon, France, 2012, pp. 358–367.

[48] C. Li, A. Sun, and A. Datta, "Twevent: Segment-based event detection from tweets," in *Proc. 21st ACM Int. Conf. Inf. Knowl. Manag.*, 2012, pp. 155–164.

[49] M. Walther and M. Kaisser, "Geo-spatial event detection in the Twitter stream," in *Proc. Eur. Conf. Inf. Retrieval*, 2013, pp. 356–367.

[50] J. H. Lau, N. Collier, and T. Baldwin, "On-line trend analysis with topic models: #Twitter trends detection topic model online," in *Proc. COLING*, 2012, pp. 1519–1534.

[51] Y. Hu, A. John, D. D. Seligmann, and F. Wang, "What were the tweets about? Topical associations between public events and Twitter feeds," in *Proc. ICWSM*, 2012, pp. 1–8.

[52] X. Zhou and L. Chen, "Event detection over Twitter social media streams," *VLDB J. Int. J. Very Large Data Bases*, vol. 23, no. 3, pp. 381–400, 2014.

[53] Y. Li *et al.*, "A survey on truth discovery," *ACM SIGKDD Explorations Newslett.*, vol. 17, no. 2, pp. 1–16, 2016.

[54] H. Shao *et al.*, "Truth discovery with multi-modal data in social sensing," *IEEE Trans. Comput.*, vol. 70, no. 9, pp. 1325–1337, Sep. 2021.

[55] K. Khalil, A. Aqil, S. V. Krishnamurthy, T. Abdelzaher, and L. Kaplan, "NEST: Efficient transport of data summaries over named data networks," in *Proc. IFIP Netw. Conf. (IFIP Networking) Workshops*, Zurich, Switzerland, 2018, pp. 1–9.

[56] Y. Ma, Y. Guo, X. Tian, and M. Ghanem, "Distributed clustering-based aggregation algorithm for spatial correlated sensor networks," *IEEE Sensors J.*, vol. 11, no. 3, pp. 641–648, Mar. 2011.

[57] H. Gupta, V. Navda, S. Das, and V. Chowdhary, "Efficient gathering of correlated data in sensor networks," *ACM Trans. Sens. Netw.*, vol. 4, no. 1, p. 4, 2008.

[58] C. Xiang, P. Yang, C. Tian, H. Cai, and Y. Liu, "Calibrate without calibrating: An iterative approach in participatory sensing network," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 2, pp. 351–361, Feb. 2015.

[59] S. Moltchanov, I. Levy, Y. Etzion, U. Lerner, D. M. Broday, and B. Fishbain, "On the feasibility of measuring urban air pollution by wireless distributed sensor networks," *Sci. Total Environ.*, vol. 502, pp. 537–547, Jan. 2015.

[60] R. McCreadie, C. Macdonald, I. Ounis, M. Osborne, and S. Petrovic, "Scalable distributed event detection for Twitter," in *Proc. IEEE Int. Conf. Big Data*, Silicon Valley, CA, USA, 2013, pp. 543–549.

Abdulrahman Fahim received the B.Sc. degree in electrical engineering from Nile University, Giza, Egypt, in 2014, and the M.Sc. degree in wireless communications from the Wireless Intelligent Networks Center, Nile University, in 2016. He is currently pursuing the Ph.D. degree in computer science with the University of California at Riverside, Riverside, CA, USA.

In 2014, he joined the Wireless Intelligent Networks Center, Nile University as a Research Assistant. His research interests are networks, network security, machine learning applications, and RFID networks.

Ajaya Neupane received the Ph.D. degree in computer science from the University of Alabama at Birmingham, Birmingham, AL, USA, in 2017.

He is currently a Senior Staff Researcher with Palo Alto Networks, Santa Clara, CA, USA. In this role, he is building robust machine learning models to detect malicious activities and prevent network attacks. He was a Postdoctoral Researcher with the University of California at Riverside, Riverside, CA, USA, before joining Palo Alto Networks. He has published more than 20 peer-reviewed papers, with a "Distinguished Paper Award" at NDSS'14. Besides machine learning and cybersecurity, he is interested in sports and hiking.

Dr. Neupane received the Outstanding Doctoral Student Award for his Ph.D. degree.

Evangelos E. Papalexakis (Member, IEEE) received the Diploma and M.Sc. degrees in electronic and computer engineering from the Technical University of Crete, Chania, Greece, in February 2010 and July 2011, respectively, and the Ph.D. degree from the School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA, in August 2016.

He is an Associate Professor with the CSE Department, University of California at Riverside, Riverside, CA, USA. Broadly, his research interests span the fields of data science, machine learning, artificial intelligence, and signal processing. His research involves designing interpretable models and scalable algorithms for extracting knowledge from large multiaspect data sets, with specific emphasis on tensor factorization models, and applying those algorithms to a variety of real-world problems, including detection of misinformation on the Web, explainable AI, and gravitational-wave detection.

Dr. Papalexakis's work has appeared in top-tier conferences and journals and has attracted a number of distinctions, including the 2017 SIGKDD Dissertation Award (runner-up), the Number of Paper Awards, the National Science Foundation CAREER Award, and the 2021 IEEE DSAA Next Generation Data Scientist Award.

Lance Kaplan (Fellow, IEEE) received the B.S. degree (Distinction) in electrical engineering from Duke University, Durham, NC, USA, in 1989, and the M.S. and Ph.D. degrees in electrical engineering from the University of Southern California, Los Angeles, CA, USA, in 1991 and 1994, respectively.

From 1987 to 1990, he worked as a Technical Assistant of the Georgia Tech Research Institute, Atlanta, GA, USA. He held a National Science Foundation Graduate Fellowship and the USC Dean's Merit Fellowship from 1990 to 1993 and worked as a Research Assistant with the Signal and Image Processing Institute, University of Southern California, from 1993 to 1994. Then, he worked as a Staff with the Reconnaissance Systems Department, Hughes Aircraft Company, Los Angeles, from 1994 to 1996. From 1996 to 2004, he was a Faculty Member with the Department of Engineering and a Senior Investigator with the Center of Theoretical Studies of Physical Systems, Clark Atlanta University, Atlanta. He is currently a Team Leader of the Context Aware Processing Branch, DEVCOM Army Research Laboratory, Adelphi, MD, USA. His current research interests include information/data fusion, reasoning under uncertainty, network science, resource management and signal and image processing.

Dr. Kaplan was a three time recipient of the Clark Atlanta University Electrical Engineering Instructional Excellence Award from 1999 to 2001. He has been serving for the VP Publications of the IEEE Aerospace and Electronic Systems (AES) Society since 2021 and as for VP Conferences of the International Society of Information Fusion (ISIF) since 2014. Previously, he served as the Editor-in-Chief for the IEEE TRANSACTIONS ON AEROSPACE AND ELECTRONIC SYSTEMS from 2012 to 2017, on the Board of Governors for the IEEE AES Society from 2008 to 2013 and from 2018 to 2020, and on the Board of Directors of ISIF from 2012 to 2014. He is a Fellow of MSS and ARL.

Srikanth V. Krishnamurthy (Fellow, IEEE) received the Ph.D. degree in electrical and computer engineering from the University of California at San Diego, San Diego, CA, USA, in 1997.

From 1998 to 2000, he was a Research Staff Scientist with the Information Sciences Laboratory, HRL Laboratories, LLC, Malibu, CA, USA. He is currently a Professor of Computer Science with the University of California at Riverside, Riverside, CA, USA. His research interests are in network, computer system and ML security, and computer and wireless networks.

Prof. Krishnamurthy was a recipient of the NSF CAREER Award from ANI in 2003. He was the Editor-in Chief of ACM MC2R from 2007 to 2009 and is currently the Associate Editor-in-Chief of the IEEE TRANSACTIONS ON MOBILE COMPUTING.

Tarek Abdelzaher (Fellow, IEEE) received the Ph.D. degree from the University of Michigan at Ann Arbor, Ann Arbor, MI, USA, in 1999.

He is currently a Professor and a Willett Faculty Scholar with the Department of Computer Science, University of Illinois at Urbana–Champaign, Champaign, IL, USA. He has authored/coauthored more than 300 refereed publications in real-time computing, distributed systems, sensor networks, and control. His research interests lie broadly in understanding and influencing performance and temporal properties of networked embedded, social and software systems in the face of increasing complexity, distribution, and degree of interaction with an external physical environment.

Prof. Abdelzaher served as an Editor-in-Chief for the JOURNAL OF REAL-TIME SYSTEMS and an Associate Editor of multiple journals, including the IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, *ACM Transaction on Sensor Networks*, *ACM Transactions on Internet of Things*, *ACM Transactions on Internet Technology*, and *Ad Hoc Networks*. He is a Fellow of ACM.