Towards Scalable and Dynamic Social Sensing Using A Distributed Computing Framework

Daniel (Yue) Zhang, Chao Zheng, Dong Wang, Doug Thain, Chao Huang, Xin Mu, Greg Madey
Department of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN, USA
yzhang40@nd.edu, czheng2@nd.edu, dwang5@nd.edu, dthain@nd.edu, chuang7@nd.edu, xmu@nd.edu, gmadey@nd.edu

***Abstract*—With the rapid growth of online social media and ubiquitous Internet connectivity, social sensing has emerged as a new crowdsourcing application paradigm of collecting observations (often called claims) about the physical environment from humans or devices on their behalf. A fundamental problem in social sensing applications lies in effectively ascertaining the correctness of claims and the reliability of data sources without knowing either of them a priori, which is referred to as *truth discovery*. While significant progress has been made to solve the truth discovery problem, some important challenges have not been well addressed yet. First, existing truth discovery solutions did not fully solve the *dynamic* truth discovery problem where the ground truth of claims changes over time. Second, many current solutions are not *scalable* to large-scale social sensing events because of the centralized nature of their truth discovery algorithms. Third, the heterogeneity and unpredictability of the social sensing data traffic pose additional challenges to the resource allocation and system responsiveness. In this paper, we developed a Scalable Streaming Truth Discovery (SSTD) solution to address the above challenges. In particular, we first developed a dynamic truth discovery scheme based on Hidden Markov Models (HMM) to effectively infer the evolving truth of reported claims. We further developed a distributed framework to implement the dynamic truth discovery scheme using Work Queue in HTCondor system. We also integrated the SSTD scheme with an optimal workload allocation mechanism to dynamically allocate the resources (e.g., cores, memories) to the truth discovery tasks based on their computation requirements. We evaluated SSTD through real world social sensing applications using Twitter data feeds. The evaluation results on three real-world data traces (i.e., Boston Bombing, Paris Shooting and College Football) show that the SSTD scheme is scalable and outperforms the state-of-the-art truth discovery methods in terms of both effectiveness and efficiency.**

***Index Terms*—Crowdsourcing, Socical Sensing, Truth Discovery, Distributed Computing, Control Theory, Hidden Markov Model**

I. Introduction

This paper presents a scalable streaming truth discovery scheme for social sensing applications. Social sensing has emerged as a new paradigm of crowdsourcing applications where humans are used as ubiquitous, versatile and inexpensive sensors to report their observations (often called claims) about the physical world. This paradigm is motivated by the proliferation of portable data collection devices (e.g., smartphones), the wide adaptation of online social media (e.g., Twitter, Facebook) and the ubiquitous Internet connectivity (e.g., WiFi, 4G/5G). Examples of social sensing include obtaining real-time situation awareness in disaster and emergency response scenarios [30], intelligent transportation system applications using location based social network services [1], geotagging and urban sensing applications using inputs from common citizens [29]. A critical challenge in social sensing is referred to as *truth discovery* where the goal is to identify the reliability of the sources and the truthfulness of claims they make without the prior knowledge on either of them.

Consider a campus attack scenario (e.g., OSU attack in Nov. 2016) as an example. A significant amount of reports about the current situation of the attack (e.g., the number of casualties, the escape path of suspects and safety alerts) are available from the social sensors (e.g., news reporters and common citizens on social media). However, those social sensors may not always generate reliable claims and some of their claims may even contradict with each other. Table I shows some example tweets collected in the OSU attack. We observe the first two tweets report that there was a shooting happening at OSU campus while the third one report it was false news. In general, it is very challenging to identify the truthfulness of the claims without knowing the reliability of the individual sources who make them *a priori*. Additionally, sources could also intentionally or unintentionally propagate the misinformation through their social networks [28]. All these complexities make the truth discovery in social sensing a non-trivial task to accomplish.

TABLE I
Example Tweets on Contradicting Claims in OSU Campus Attack, November, 2016

Tweet	Timestamp
OSU POSSIBLE SHOOTING: I am on campus near @OSUengineering TONS of police.	28 Nov 2016, 7:23 AM
There was a shooting at Ohio state please pray for people's safety #osu	28 Nov 2016, 7:47 AM EST
Liberals putting out fake claims about the terrorist attack. 1st not a shooting, 2nd not an American, 3rd not nazi but Islamic #osushooting	28 Nov 2016, 11:37 AM EST

A rich set of solutions have been proposed in network sensing, data mining, machine learning communities to solve the truth discovery problem [2], [6]–[8], [13], [19], [30], [37], [38]. However, several significant challenges have not been

well addressed by the state-of-the-art solutions. First, existing solutions did not fully solve the dynamic truth discovery problem where the ground truth of claims changes over time. There are two critical tasks in addressing the dynamic truth challenge. The first is to capture the transition of truth in a timely manner and the second one is to be robust against noisy data that may lead to the incorrect detection of the truth transition. For example, in the Boston Bombing event in 2013, CNN claimed that a bomber was arrested two days after the event. This original message was retweeted more than 3,000 times until half an hour later it was debunked by Boston police department claiming no arrest has been made. Few hours later the real arrest was finally made. There was a transition from "arrest (false rumor)" to "no arrest" to "arrest". Only a small number of schemes been proposed to solve the dynamic truth discovery problem. For example, Pal et. al considered the evolving information of objects and estimated the truth of variables in current time interval based on sources' historical claims [18]. Li et al. proposed a Maximum A Posterior based real-time algorithm to solve the dynamic truth discovery problem [9]. However, these approaches could be unresponsive when the amount of social sensing data is large or the amount of resources on the deployed system are limited. Moreover, their solutions are not robust in noisy and sparse social sensing scenarios since their truth discovery accuracy is sensitive to the quality and quantity of the sensing data.

Second, existing truth discovery solutions did not fully explore the scalability aspect of the truth discovery problem. Social sensing applications often generate large amounts of data during some important events (e.g., disasters, sports, unrests) [21]. For example, 3.8 million people have generated a total of 16.9 million tweets with tweet per minute peaked at a rate of over 152,000 in Super Bowl 2016 [16]. However, current centralized truth discovery solutions are incapable of handling such large volume of social sensing data due to the resource limitation on a single computing device. A limited number of distributed solutions have been developed to address the scalability issue of the truth discovery problem. Both Ouyang et al. [17] and Meng et al. [15] proposed a distributed solution based on Hadoop system, but there are several non-trivial drawbacks. First, Hadoop is a heavy-weight solution in the sense that it requires a long start up time. Second, Hadoop is designed as a batch processing system that is most suitable for data of very large volume (e.g., Petabytes of data) and may not be the best solution for the size of datasets collected in many social sensing events (e.g., GB to TB). Third, Hadoop assumes homogeneity of the underlying computing nodes [23], which ignores the heterogeneity of the computational resources we have in real distributed systems.

The third challenge lies in the heterogeneity and unpredictability of the streaming data traffic. First, different topics or events are likely to generate different amounts of social sensing data (e.g., Hurricane Matthew generates way more tweets than a local traffic accident) [25]. Second, the traffic volume of the same event is not constant over time (e.g., there is often a spike in the number of tweets when there's a touchdown in a football game). Such heterogeneity needs to be appropriately handled by the social sensing system to provide reliable and responsive truth discovery results. Previous truth discovery solutions assumed the datasets can be evenly partitioned into chunks of similar sizes and all partitions can be processed in a synchronized manner [15], [17]. However, such strong homogeneity assumption on the data streams barely holds in real world social sensing applications.

In this paper, we develop a Scalable Streaming Truth Discovery (SSTD) scheme to address the above challenges. To address the dynamic truth discovery challenge, we develop a Hidden Markov Model based solution to dynamically estimate the true value of claims based on the observations reported by social sensors. To address the scalability challenge, we developed a light-weight distributed framework that is both *scalable* and *efficient* to solve the truth discovery problem using Work Queue and HTCondor system. To address the data heterogeneity challenge, we integrated the SSTD scheme with an optimal workload allocation mechanism using feedback control (i.e., Proportional Integral Derivative (PID) controller) to dynamically allocate the resources (e.g., cores, memories) to the truth discovery tasks. We evaluated the SSTD scheme in comparison with the state-of-the-art truth discovery baselines using three real-world social sensing data traces (i.e., Boston Bombing, Paris Shooting and College Football) collected from Twitter. The evaluation results show that our SSTD scheme significantly outperforms the compared baselines in terms of truth discovery accuracy and computational efficiency.

In summary, the contributions of this paper are as follows:

- This paper addresses three fundamental challenges in truth discovery problem in social sensing: *dynamic truth*, *scalability* and *heterogeneity of streaming data*.
- We develop the SSTD scheme that incorporates the Hidden Markov Model (HMM) to effectively address the dynamic truth discovery challenge.
- We develop a light-weight distributed framework based on Work Queue and HTCondor system to address the scalability challenge.
- We integrate the SSTD scheme with an optimal workload allocation mechanism to address the heterogeneity of the streaming social sensing data.
- We evaluate the performance of the SSTD scheme and compare it with the state-of-the-art truth discovery solutions through real-world case studies. The evaluation results demonstrate the effectiveness and significant performance gains achieved by our scheme.

II. Problem Formulation

Consider a social sensing application where a group of M sources $S = (S_1, S_2, ..., S_M)$ report a set of N claims $C = (C_1, C_2, ..., C_N)$. Let S_i denote the i-th source and C_u denote the u-th claim. We define $R_{i,u}^t$ to be the report made by source S_i on claim C_u at time t. In this paper, we focus on binary claims and sources can report a claim to be either true or false. Twitter, for example, can be considered as a social sensing application where observations from average people are used

to obtain the real-time situation awareness of disaster events (e.g., earthquake, hurricane). A source represents a Twitter user and a claim is a statement of an event, topic or object that is derived from his/her tweet. For example, a tweet "The USC-Notre Dame game is close and the Irish have the lead." can be considered as a report to a claim "Notre Dame is leading the football game". To model the dynamics of claim truthfulness, we assume the truthfulness of the claim changes over time. Therefore, we use $C_{u,t} = T$ and $C_{u,t} = F$ to represent a claim C_u is true or false at time t respectively. We assume a claim cannot be both true and false at the same time.

One key challenge in social sensing applications lies in the fact that sources are often unvetted and they may not always report truthful claims. Therefore, we need to explicitly model the reliability of data sources. However, data sparsity is a common problem in social sensing applications where most sources only contribute a small number of claims [36]. It is challenging to accurately estimate the reliability of sources due to the lack of sufficient evidence. Fortunately, the reports themselves often contain extra evidence and information to infer the truthfulness of a claim. In the Twitter example, the text, pictures, URL links and geotags contained in the tweet can all be considered as extra evidence of the report. To leverage such evidence in our model, we define a *contribution score* for each report to represent how much the report contributes to the belief in the truthfulness of a claim. We first define the following terms related to contribution score.

DEFINITION 1: Attitude Score ($\rho^t_{i,u}$) : a score represents whether a source believes the claim is true, false or does not provide any report. We use 1, -1 and 0 to represent these attitudes respectively.

DEFINITION 2: Uncertainty Score ($\kappa^t_{i,u}$): a score in the range of (0,1) that measures the uncertainty of a report. A higher score is assigned to a report that expresses more uncertainty.

DEFINITION 3: Independent Score: ($\eta^t_{i,u}$): a score in the range of (0,1) that measures whether the report $R_{i,u}$ is made independently or copied from other sources. A higher score is assigned to a report that is more likely to be made independently.

Combining the above terms, we formally define *Contribution Score* as:

$$CS^t_{i,u} = \rho^t_{i,u} \times (1 - \kappa^t_{i,u}) \times \eta^t_{i,u} \tag{1}$$

Furthermore, we define the estimated truth of the u-th claim at time t as $\hat{x}_{u,t}$ and the ground truth label of claim C_u at time t as $x_{u,t}$. Given the sources, claims and reports in social sensing applications, the objective of dynamic truth discovery is to correctly estimate the truthfulness of claims at each time instant. In particular, for each claim C_u at time instant t, our goal is to derive the estimate $\hat{x}_{u,t}$ that is as close as possible to the ground truth $x_{u,t}$, which is given by:

$$\arg\max_{\hat{x}_{u,t}} P(\hat{x}_{u,t} = x_{u,t}|S, C, R) \tag{2}$$

The above problem is more challenging when the computational resources needed to solve the problem are taken into consideration. On one hand, many current truth discovery solutions developed batch algorithms that are shown to achieve reasonable estimation accuracy at the cost of high computational complexity and incapability to scale to large-scale social sensing events [11], [19], [34]. On the other hand, simple heuristic algorithms such as Majority Voting and Median are very fast but the truth discovery accuracy is quite low [9]. Additionally, the available computational resources (e.g., disk, memory, cores) on the deployed systems are often limited, which pose extra constraints to the dynamic truth discovery problem. Therefore, the goal of this paper is to optimize both the *estimation accuracy* and *computational efficiency* of the SSTD scheme while meeting all resource constraints of the deployed systems.

To model the system aspect of the problem, we define a few additional terms to be used in our problem formulation. First, we use $D^{\Delta t}$ to represent the total amount of data that is generated by the social sensing application for a given time interval Δt. We have a set of jobs $TD = \{TD_1, TD_2, ..., TD_J\}$ that run on a distributed system to solve the dynamic truth discovery problem. We refer to these jobs as Truth Discovery (TD) jobs. For each job, we define the Worse Case Execution Time (WCET) as the maximum running time for that job to finish the truth discovery task. In particular, we denote $w_j^{\Delta t}$ as the WCET for processing $D^{\Delta t}$ by job TD_j. For the deployed system, we consider a computer cluster of K nodes $Nodes = \{N_1, N_2, ..., N_K\}$. For each node N_k, it has a set of Z resource constraints $RC^k = RC^k_1, RC^k_2, ..., RC^k_Z$ where each constraint defines the maximum availability of a specific system resource (e.g., memory, disk space, CPU). Furthermore, we define a set of soft deadlines $dl = dl_1, dl_2, ..., dl_J$ to specify the expected finishing time of the TD jobs in order to optimize the responsiveness of the system.

With the above definitions, we can formulate the dynamic and distributed truth discovery problem with resource constraints as a constrained optimization problem. Formally, we want to achieve:

$$\begin{aligned} &\text{maximize} \quad P(\hat{x}_{u,t} = x_{u,t}|S, C, R), \forall t > 0 \\ &\text{and} \quad P(w_j^{\Delta t} \leq dl_j), \forall 1 \leq j \leq J \\ &\text{s.t. } RC^k \quad \text{is satisfied} \quad \forall 1 \leq k \leq K \end{aligned} \tag{3}$$

III. Scalable and Streaming Truth Discovery

In this section, we develop the Scalable and Streaming Truth Discovery (SSTD) scheme to solve the problem formulated in the previous section. In particular, we propose a Hidden Markov Model (HMM) based solution to decode the streaming social sensing data and output the corresponding truth values of claims in real time. This solution can be implemented in a distributed system where multiple truth discovery jobs can run in parallel to address the scalability challenge in social sensing applications.

A. Overview of Hidden Markov Model

A Hidden Markov Model (HMM) is a stochastic state transition model that is commonly used to model systems with unobserved (hidden) states and it has been commonly used in applications such as speech recognition, activity recognition and intrusion detection [22]. The HMM contains a set of hidden states $States = \{s_1, s_2, ..., s_X\}$ and a set of observation symbols $Obs = \{o_1, o_2, ..., o_Y\}$. At each time instant t, the model is in a hidden state $s_t \in States$ and we observe an observation symbol $o_t \in Obs$. The HMM is particularly suitable to handle time series data and capture the dynamics of state change in real time. In particular, given a series of observations, it can decode the hidden states that generated those observations at each time instant. In this paper, we develop a HMM based streaming truth discovery model to address the dynamic truth challenge.

B. Deriving Hidden States and Observation Sequence

Our goal is to estimate the true values of the claims and such true values are often not directly observable and constantly changing (e.g., the location of a terrorist suspect, the number of casualties during a natural disaster). We define the evolving truth of a claim as the hidden states for HMM. We formally define *hidden states of truth* as:

DEFINITION 4: Hidden States of Truth: the true value for the claim at a given time instant that is not directly observable.

We assume that the sequence of truth values can be modeled as a Markov chain in which each single state of truth is assumed to be related to its previous state. The inference of such hidden states requires a visible observation sequence that can be directly observed from data sources. In our HMM model, we define our observation sequence as a vector of Aggregated Contribution Scores (ACS).

DEFINITION 5: Aggregated Contribution Score (ACS): the sum of contribution scores on the true values of the claim during a time interval.

The ACS for a claim C_u at time instant t is calculated as:

$$ACS_u^t = \sum_{t-sw}^{t} CS_{i,u}^t \tag{4}$$

where $CS_{i,u}^t$ is the contribution score of report $R_{i,u}^t$ (made by S_i on C_u at time instant t). We use a sliding window (i.e., sw) to control the time period of historical contribution scores we want to consider for calculating ACS. The size of the sliding window is decided based on the expected change frequency of the truth from the observed event.

For claim C_u, we use a sequence of ACS (i.e., $F(u) = (ACS_u^1, ACS_u^2, ..., ACS_u^T)$) as the input for the HMM model. The output of the model is the corresponding sequence of estimated truth $(\hat{x}_{1,u}, \hat{x}_{2,u}, ..., \hat{x}_{T,u})$ for C_u. The HMM based truth discovery model is shown in Figure 1.

C. Estimating Parameters of the HMM Model

We now define the estimation parameters in our model.

- Truth Value Vector V: a set of possible true values for claims. Here we only consider two values: true and false.

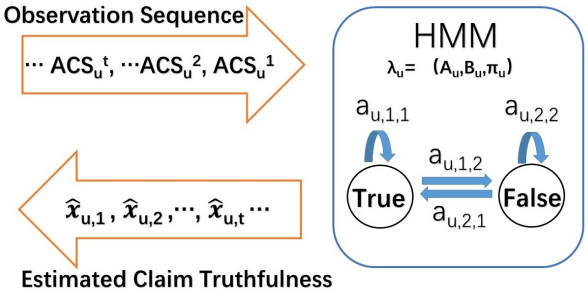


Fig. 1. HMM Based Truth Discovery Model

- Truth Transition Probability A_u: a 2 by 2 matrix where each element $a_{u,i,j}$ is the probability of the true value of C_u transits from value V_i to value V_j.
- Emission Probability B_u: a sequnce of generation liklihood with each element $b_{u,i,t}$ denoting the probability of observing ACS_u^t while the true value of C_u is V_i.
- Initial State Distribution π_u: each element $\pi_{u,i}$ represents the probability of the initial true value of C_u being V_i.
- $\lambda_u = (A_u, B_u, \pi_u)$: the set of parameters that define the HMM model of claim C_u.

We train our HMM model to find the set of parameters that maximize the probability of the observation sequence $F(u) = \{ACS_u^1, ACS_u^2, ..., ACS_u^T\}$ for each claim C_u. Formally, our goal is to find:

$$\begin{aligned} \lambda_u^* &= \arg\max_{\lambda_u} P(F(u)|\lambda_u) \text{ for } u \in N \\ &= \arg\max_{\lambda_u} P(\{ACS_u^1, ACS_u^2, ..., ACS_u^T\}|\lambda_u) \end{aligned} \tag{5}$$

We solve this problem by using an expectation maximization (EM) based algorithm. The derivations of the E and M steps of the EM algorithm are presented in Section IX-A

D. Decoding State Sequence

Finally, we estimate the true value of each claim at every time instant. This is done by the decoding step where the goal is to find the sequence of true values that is most likely to generate the observed sequence. Formally, it is given by:

$$(\hat{x}_{1,u}, \hat{x}_{2,u}, ..., \hat{x}_{T,u}) = \arg\max_{VT_u} P((ACS_u^1, ..., ACS_u^T)|VT_u) \tag{6}$$

where $VT_u = (V_u^1, V_u^2, ..., V_u^T)$ is the hidden sequence of the true values. We apply the Viterbi Algorithm [27] to solve the above truth decoding problem. Given the corresponding observation sequence of $F(u) = (ACS_u^1, ACS_u^2, ..., ACS_u^T)$ of claim C_u and the estimated parameters λ_u of our HMM model, we can infer the corresponding hidden true value sequence VT_u that is most likely to generate the observations. It is solved recursively as follows:

$$\begin{aligned} \delta_t(u,i) = \max_{V_u^1, V_u^2, ..., V_u^{t-1}} & P(V_u^1, V_u^2, ..., V_u^{t-1}, ACS_u^1, ACS_u^2, \\ & ..., ACS_u^t, V_u^t = V_i|\lambda_u) \\ \delta_{t+1}(u,i) = b_{u,i,t} \max_{1 \le i \le 2} & \delta_t(u,i) a_{u,i,j} \end{aligned} \tag{7}$$

$\delta_t(u,i)$ represents the probability that the HMM's current true value is V_i after seeing the first t observations and passing through the most probable true value sequence $V_u^1, ..., V_u^{t-1}$, given the estimated parameter set λ_u. The initialization is: $\delta_{t+1}(u,i) = \pi_{u,i} * b_{u,i,1}$. The estimated index of the true value at time t can be calculated as:

$$l = \arg\max_{1 \le i \le 2} \delta_t(u,i) \tag{8}$$

The estimated true value of C_u at time t is: $\hat{x}_{t,u} = V_l$.

E. Scalability of HMM based Truth Discovery Scheme

The HMM based model allows us to divide the social sensing data stream into multiple sub-streams based on claims and develop a distributed solution to solve the truth discovery problem in a scalable way. This scalable feature of our solution comes from the fact that the HMM based model relies on the ACS rather than individual source reliability to infer the truthfulness of claims. The ACS reflects the aggregated evidence from the collective observations of social sensors. In the SSTD scheme, for a claim C_u, we assign it to a TD job TD_u. The job implements a data preprocessing step (discussed in details in V-A) and the truth discovery step (i.e., HMM based solution) to decode truth from the data stream for C_u. All TD jobs are running in parallel and new TD jobs will be dynamically spawned when new claims are generated. The implementation of SSTD in a distributed framework is discussed in details in the next section.

IV. Implementation on A Distributed Computing Framework

In this section, we present a distributed implementation of the SSTD system using HTCondor and Work Queue. We also develop an optimal task allocation mechanism to optimize the system performance using the control theory.

A. Background

1) HTCondor: We use the HTCondor from University of Notre Dame as the underlying distribute system for the implementation of SSTD scheme. The system consists of over 1,900 machines and over 14,700 cores at the time of writing. HTCondor has been used by hundreds of organizations in industry, government, and academia to manage computing clusters ranging from a handful to many thousands of workstations cores [14]. The HTCondor system at University of Notre Dame is deployed to all available machines, including desktop workstations, classroom machines, and server clusters, all of which are typically idle 90% of the day. Users send their computation jobs to run in the HTCondor system, and the system allocates the jobs to run on machines that would otherwise go unused.

2) Work Queue: Work Queue is a lightweight user-level execution engine for constructing large scale distributed implementations [4]. The system consists of a master process and a large number of worker processes that can be deployed across heterogeneous cluster, cloud, and grid infrastructures. Work Queue allows the master process to define a set of tasks (i.e., Task Pool), submit them to the queue, and wait for completion. It also defines a Worker Pool, which is a set of workers that can run on any machines. A Worker is defined as a process that performs specific computational functions described by the tasks. Once running, each worker calls back to the master process, arranges for data transfer, and executes the tasks. Work Queue maintains an elastic worker pool that allows users to scale the number of workers up or down as required by their applications. We use Work Queue on top of HTCondor system to take advantage of its dynamic resource allocation mechanism for TD job allocations.

B. Overview of SSTD Architecture

The architecture of the SSTD system implementation is shown in Figure 2. A key component is the Dynamic Task Manager (DTM) which is implemented as a master process for Work Queue that initializes a Worker Pool and dynamically spawns new tasks into the Task Pool. The DTM divides the TD jobs into multiple tasks that run in parallel on the HTCondor system. A feedback control system is integrated with SSTD scheme to monitor the current execution speed of each TD job and estimate its expected finish time. The control system emits the control signals and feeds them back to the DTM by comparing the expected finish time of TD jobs with corresponding deadlines specified by the applications. DTM then dynamically adjusts the priorities of TD jobs based on the control signals.

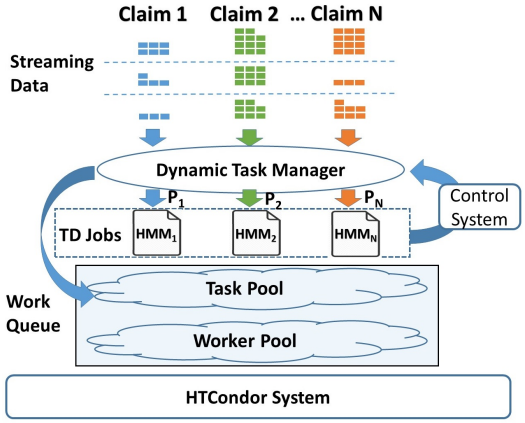


Fig. 2. SSTD Implementation Architecture

C. Dynamic Task Management and Feedback Control System

In this subsection, we discuss the implementation details of Dynamic Task Manager (DTM). The DTM is designed to dynamically manage the tasks and resources in order to optimize the system performance.

1) Deadline Driven System Optimization: In DTM, we designed a deadline driven optimization scheme to optimize the system performance and address data heterogeneity problem of the streaming social sensing data. The goal of deadline driven optimization is mainly twofold. First, the SSTD scheme should be able to meet the real-time Quality of Service (QoS) requirements from various social sensing applications. Second,

the SSTD scheme should accommodate the heterogeneity of data streams from different TD jobs. In particular, the SSTD scheme will maximize the probability for all TD jobs to finish before their deadlines. In this paper, we develop a dynamic feedback control system to achieve the above goal.

2) Dynamic Feedback Control System: The architecture of the dynamic feedback control system is shown in Figure 3. It consists of three key components: Proportional Integral Derivative (PID) feedback controller, Local Control Knob (LCK) and a Global Control Knob (GCK). The LCK refers to a local control variable that is used to tune the performance of a particular TD job. In SSTD scheme, the LCK is the priority assignment of each TD job. The GCK refers to a global control variable that is used to tune the performance of all TD jobs in the system. In SSTD scheme, we use the total number of workers in the Worker Pool as the GCK.

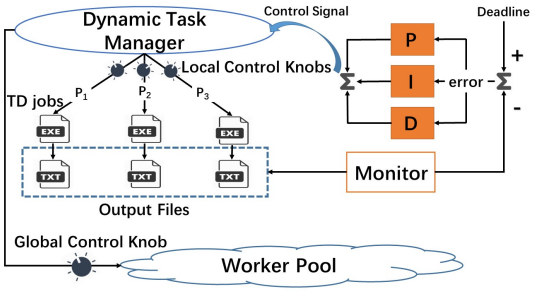


Fig. 3. Dynamic Feedback Control System

3) Feedback Control Mechanism: PID is a feedback control mechanism that is commonly used in industrial control systems. A PID controller continuously calculates an error value as the difference between the desired value (setpoint) and a measured process variable and applies a correction based on proportional, integral, and derivative terms of the error [24]. The advantage of using PID is that it provides a simple and flexible controller without requiring precise modeling of the system. In our SSTD scheme, we use the deadline of each TD job as the setpoint and compare it with the actual execution time of the job. The execution time is obtained by continuously monitoring the timestamps of the output files of the TD job. The error value $e(k,u)$ denotes the difference between the current execution time and the deadline for TD_u. Here k denotes the k-th sample. We use a sampling rate of 1 second and the control signal $y(k,u)$ for TD_u is calculated as:

$$y(k,u) = K_p e(k,u) + K_i \sum_{0}^{k} e(k,u)\Delta t + K_d \frac{\Delta e(k,u)}{\Delta t} \quad (9)$$

where K_p, K_i, K_d are the coefficients for the proportional, integral, and derivative terms in the PID controller respectively. We heuristically tune these parameters by picking the set of values when the tasks in the system the meet the most deadlines. Details of parameter tuning can be found in V-A.

4) WCET Calculation and Control Knob Tuning: The DTM dynamically adjusts control knobs to optimize system performance based on the control signals from the PID controller. In particular, the DTM adjusts both the job priority assignments (LCK) and the number of workers in the Worker Pool (GCK).

The job priority reflects the probability that a TD job acquires system resource and gets processed by workers in the system. A higher priority job is more likely to be processed earlier than a low priority job. Each TD jobs can be split into multiple TD tasks in Work Queue system where each task has the same probability of being processed by the worker. The priority of job TD_u is defined as $P_u = \frac{T_u}{\sum_{u=1}^{N} T_u}$ where T_u is the total number of tasks for TD_u. We divide the data of each TD job equally between its tasks. The execution time ET_u for each task in TD_u is calculated as:

$$ET_u = TI + D_u^{\Delta t} * \theta_1 \quad (10)$$

where TI is the initialization time for each task, θ_1 is a constant and $D_u^{\Delta t}$ is the size of data for job TD_u in time interval Δt. We assume the WECT of a TD job is inversely proportional to its priority and the size of the worker pool. We can derive the WECT for job TD_u as:

$$\begin{aligned} WCET_u^T &= TI * T_u + \frac{D_u^{\Delta t} * \theta_2}{WK * P_u} \\ &= TI * T_u + \frac{D_u^{\Delta t} * \theta_2 * \sum_{u=1}^{N} T_u}{WK * T_u} \end{aligned} \quad (11)$$

where WK is the number of workers and θ_2 is a constant. We observe the data initialization cost increases quickly when the number of tasks is large. To minimize the initialization overhead, we keep the number of tasks in each TD job small and WECT can be simplified as:

$$WCET_u^T \approx \frac{D_u^{\Delta t} * \theta_2}{WK * P_u} \quad (12)$$

Based on Equation (12), we may improve each TD job's WECT by increasing either its job priority or the number of workers. To optimize system performance, The execution time of TD jobs is used to generate the PID *control signals* defined in Equation (9) in the system. The system then tunes both LCK and GCK based on the control signals and the rubrics discussed in details in Section IX-B.

V. Evaluation

In this section, we evaluate the performance of the SSTD scheme and compare it with the state-of-the-art truth discovery baselines on three real-world data traces collected from social sensing applications. The results show that the SSTD scheme outperforms the compared baselines in terms of both truth discovery accuracy and computational efficiency.

A. Experimental Setups

1) Baseline Methods: We chose the following six representative truth discovery solutions as the baselines.

- **TruthFinder:** It uses a pseudo-probabilistic function to estimate source reliability and claim truthfulness using an iterative algorithm [34].

- **RTD:** A truth discovery scheme that is robust against misinformation in social media applications [36]. It leverages the historical claims of each source to detect widely spread misinformation.
- **CATD:** It provides confidence interval estimators for source reliability using sparse data. The proposed method can effectively estimate the reliability of sources with various levels of data contribution from sources [7].
- **Invest:** Invest algorithm "invests" source reliability among their claims and the truthfulness for each claim can be obtained by using a non-linear function [19].
- **3-Estimate:** It identifies the truth of a claim by estimating the three defined parameters in their models related with source reliability and claim truthfulness [11].
- **DynaTD:** A dynamic truth discovery algorithm using Maximum A Posteriori Estimation to dynamically estimate source reliability and truth of claims in real time [9].

2) Data Collection and Pre-Processing: We collected three real-world data traces from Twitter in the aftermath of emergency events and major sporting events for the purpose of evaluation. Twitter has emerged as a new social sensing experiment platform where massive observations are uploaded voluntarily from human sensors to report the events happened in the physical world. We noted conflicting information is common to observe on Twitter due to the open data collection environment and unvetted nature of data source. We found the truth of the claims on Twitter change dynamically. In the emergency events, for example, the location of the affected area, the number of death are changing dynamically. During a sporting event, the scores and leading status of the game also change over time. This noisy and dynamic nature of Twitter provides us a good opportunity to investigate the performance of the SSTD scheme in a real world social sensing scenario. The collected traces are summarized in Table II.

Boston Bombing Trace & Paris Shooting Trace : We collected Twitter data related to the 2013 Boston Bombing event and 2015 Paris Shooting event through Twitter open search API (using the query terms such as "Boston", "Marathon", "Bombing", "Paris", "Shooting") and specified geographic regions related to the event (using a circular region centered at event location within a radius of 100 miles).

Data Pre-processing: To derive claims of the dataset, we implemented a simple claim generator based on keywords filtering and clustering techniques. In particular, we first used a set of pre-specified keywords to filter out tweets that are irrelevant to the event of interests for the truth discovery task. We then use a variant of K-means clustering algorithm and a commonly used distance metric for micro-blog data clustering (i.e.,Jaccard distance) [26] to cluster tweets of similar content into the same cluster. The clustering algorithm is an online algorithm for the streaming social sensing data. In particular, a newly arrived tweet will be clustered into one of the existing clusters based the computed Jaccard distance and a cluster will be broken into two clusters if the diameter of the cluster is larger than some pre-specified threshold learned from previous case studies [30]. We treat a topic directly related to the terrorist attack events in each cluster as a claim (e.g., the location of the suspect, bomb threat in JFK library, whether an arrest has been made, etc.).

In order to compute the *Contribution Score*, we first calculate the *Attitude Score* using a heuristic method based mainly on the content of the tweet to classify it as "agree" or "disagree" (e.g., whether a tweet contains certain negative words such as "false", "fake", "rumor", "debunked", "not true"). We assign a score of "1" and "-1" respectively. We then calculate the *Uncertainty Score* by implementing a simple text classifier using skit-learn and trained it with the training data provided by CoNLL-2010 Shared Task [10]. To compute the *Independent Score*, we classified the retweets or tweets that are significantly similar to the previous tweets within a time interval as repeated claims and assign them relatively low independent scores.

Labeling Ground Truth: We manually verified the ground truth of the claims using the historical facts about the Boston Bombing and Paris Shooting events.

College Football Trace: We collected Twitter data from five US college football games in the weekend of Sept 29, 2016. These data traces were collected through Twitter Streaming API using query terms related to the names of the teams and schools (e.g., "Fighting Irish", "Buckeyes", "Notre Dame") and specified geographic regions of the game (e.g., within 30 miles radius from the stadium).

Data Pre-processing: For each game, the change of score is treated as the claim, therefore, we consider the binary value of either score changes or no score change. We calculate the *Attitude Score* based on keyword matching. For example, tweets containing "taking the lead", "score", "tied" are considered supportive attitude of score change and it is assigned a score of 1 while the rest of the tweets are assigned a score of "-1". For the *Uncertainty Score*, *Independent Score* and *Source Attitude*, we use the same approach as we discuss in the Boston data trace.

Labeling Ground Truth: We manually label the ground truth of the game status based on credible post-game statistics from ESPN.com[1] and FCS[2].

3) System Setup: We implemented the master program of SSTD on a single HTCondor node of 4 processors and 8G of RAM. On this node, the DTM is written as a Work Queue master script and it is connected to the data crawler which continuously fetches the social sensing data. Both programs are written in Python. The node is connected to the Notre Dame Condor Cluster which executes all TD jobs. For all compared baselines, we run them on the same HTCondor node separately to ensure fairness in the comparison. However, they are not executed in the HTCondor cluster since they are not designed as distributed schemes. To tune the PID control system, we increase each coefficient from 0.0 to 3.0 by 0.1 . We pick the set of coefficients that maximize the number of jobs that can meet their deadlines. The coefficients we use

[1]http://www.espn.com/college-football

[2]http://www.fcs.football/cfb

TABLE II
DATA TRACE STATISTICS

Data Trace	Paris (Charlie Hebdo) Shooting	Boston Bombing	College Football
Start Date	Jan. 1 2015	Apr. 15 2013	Sep. 30 2016
Time Duration	3 days	4 days	3 days
Search Keywords	Paris, Shooting, Charlie Hebdo	Bombing, Marathon, Attack	Team/College names
# of Reports	253,798	553,609	429,019
# of Sources	217,718	493,855	413,782

for our SSTD system are $1.2, 0.3$ and 0.2 for K_p, K_i and K_d respectively. We also set θ_3 and θ_4 as 2 and 1.5 for tuning control knobs using similar heuristics.

B. Evaluation on Real World Data Traces

1) Evaluation Metrics: We evaluate the performance of SSTD scheme in comparison with state-of-the-art solutions from the following perspectives: (i) *effectiveness*: the accuracy of the truth discovery solutions (i.e., *Accuracy*, *Precision*, *Recall* and *F1-Score*); (ii) *efficiency*: the time to finish execution of the truth discovery tasks; (iii) *controllability*: the percentage of truth discovery tasks that meet the specified deadlines.

2) Evaluation Results: We first evaluate the effectiveness of all compared truth discovery schemes. The evaluation results on the three data traces are shown in Table III, IV and V respectively. We observe that SSTD scheme outperforms compared baselines on all four metrics of the effectiveness. In particular, the performance gain achieved by SSTD scheme compared to the best performed baseline on accuracy, precision, recall and F1-Score is 6.5%, 2.3%, 0.7% and 4.9% on the Boston Bombing data trace; 4.9%, 1.1%, 8.2% and 6.6% on the Paris Shooting data trace and 3.6%, 10.6%, 2.8% and 8.9% on the College Football data trace. The performance gain of the SSTD scheme is achieved by explicitly modeling the dynamic truth of the claims and incorporating contribution scores of reports to compensate the sparsity of the social sensing data.

TABLE III
TRUTH DISCOVERY RESULTS - BOSTON BOMBING

Method	Accuracy	Precision	Recall	F1-Score
SSTD	**0.828**	**0.834**	**0.831**	**0.833**
DynaTD	0.722	0.811	0.756	0.783
TruthFinder	0.653	0.689	0.787	0.734
RTD	0.763	0.748	0.824	0.784
CATD	0.667	0.764	0.748	0.751
Invest	0.609	0.639	0.626	0.632
3-Estimates	0.616	0.626	0.807	0.705

TABLE IV
TRUTH DISCOVERY RESULTS - PARIS SHOOTING

Method	Accuracy	Precision	Recall	F1-Score
SSTD	**0.802**	**0.834**	**0.905**	**0.872**
DynaTD	0.731	0.822	0.788	0.805
TruthFinder	0.616	0.653	0.806	0.721
RTD	0.753	0.791	0.823	0.807
CATD	0.669	0.689	0.760	0.723
Invest	0.661	0.722	0.780	0.750
3-Estimates	0.647	0.704	0.765	0.733

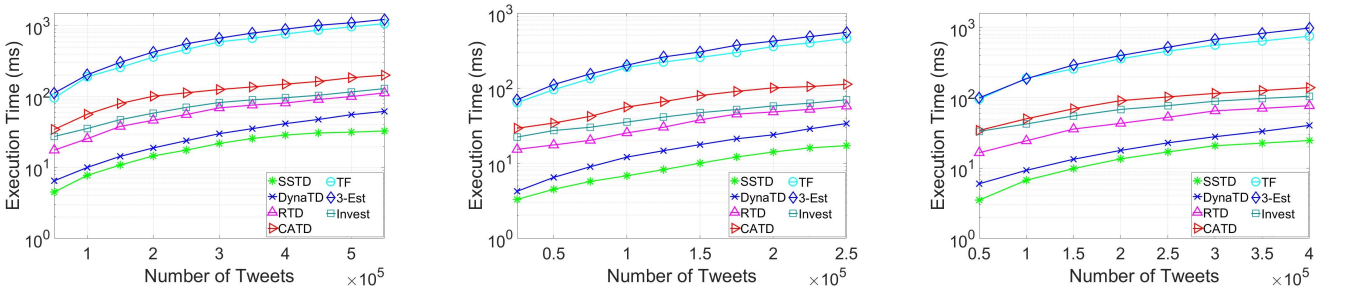
TABLE V
TRUTH DISCOVERY RESULTS - COLLEGE FOOTBALL

Method	Accuracy	Precision	Recall	F1-Score
SSTD	**0.801**	**0.661**	**0.792**	**0.723**
DynaTD	0.765	0.471	0.570	0.515
TruthFinder	0.612	0.542	0.455	0.495
RTD	0.752	0.555	0.649	0.598
CATD	0.736	0.542	0.764	0.634
Invest	0.722	0.478	0.716	0.574
3-Estimates	0.674	0.396	0.677	0.501

To evaluate the efficiency, we compare the execution time of all compared schemes. We run SSTD on the Notre Dame HTCondor cluster with a maximum number of workers as 4. We use such a small number of workers in favor to other centralized baselines that are not designed to run in a cluster. We run all other baselines on a single node of 4 processors and 8G of RAM. The results are reported in Figure 4. The results show that our scheme outperforms all other baselines by having a shorter execution time to finish the truth discovery task. We also observe that the performance gain achieved by SSTD becomes more significant when datasize becomes larger, which demonstrates the scalability of our scheme on large data traces in social sensing applications. We also envision the performance gain of SSTD over other baselines would be much larger if we run it in a larger cluster with more nodes.

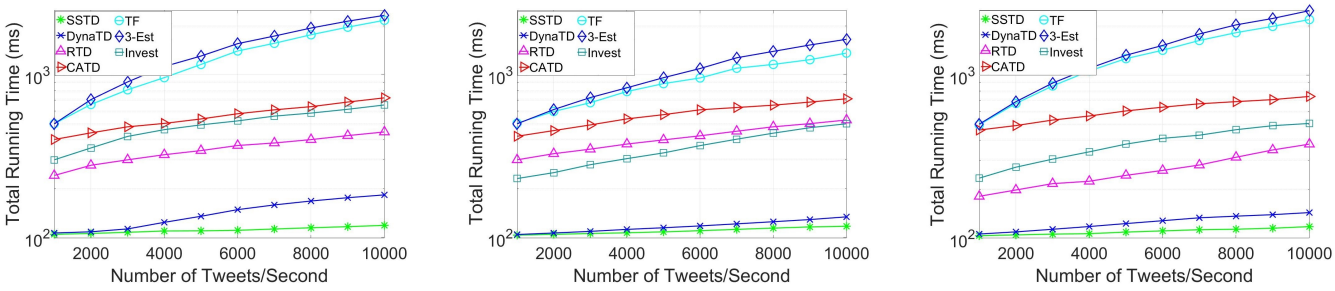
We also study the effect of the streaming speed (measured by the number of tweets per second) on the execution time of all schemes. In particular, we stream the data into compared schemes at different speed for a duration of 100 seconds. The batch (static) truth discovery schemes (e.g., CATD, TruthFinder, RTD) retrieve and process 5 seconds of data each time periodically. The streaming schemes (e.g., SSTD, DynaTD), on the other hand, keep reading new data and process them as they arrive. The results are shown in Figure 5. We observe that batch schemes can hardly catch up with the streaming data updates, especially when the streaming speed is high. The results suggest the batched schemes are not suitable to solve the dynamic truth discovery problems with streaming data. In contrast, the execution time of the streaming schemes are close to the streaming duration, which verifies the real-time feature of these streaming truth discovery algorithms. The SSTD is the fastest between all streaming schemes and is also least sensitive to the changes of the streaming speed.

We then evaluate the contrallability of SSTD scheme. We divide each data trace into 100 equal time intervals based on the timestamps of tweets. For each time interval, we record



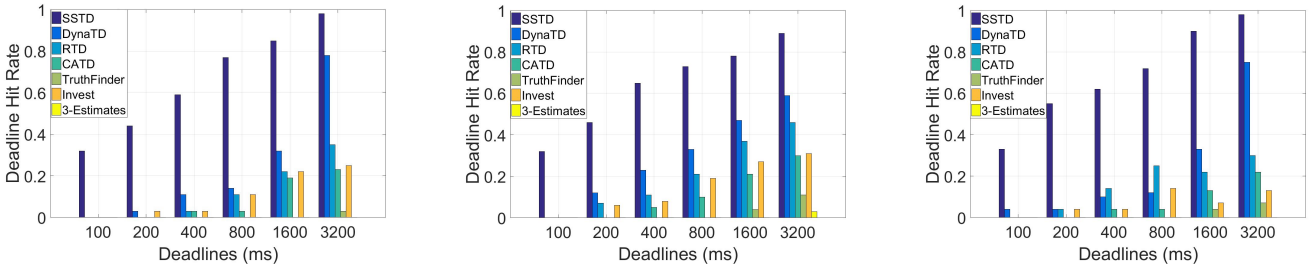
(a) Boston Bombing Trace (b) Paris Shooting Trace (c) College Football Trace

Fig. 4. Execution Time of All Compared Schemes



(a) Boston Bombing Trace (b) Paris Shooting Trace (c) College Football Trace

Fig. 5. Total Running Time vs. # of tweets per Sec of All Compared Schemes



(a) Boston Bombing Trace (b) Paris Shooting Trace (c) College Football Trace

Fig. 6. Deadline Hitting Rates of All Compared Schemes

the total execution time to process all the tweets in that time interval. We compare the execution with the deadline and we record the percentage of intervals where the execution time is less than the deadline (i.e., hit rate). The result are reported in Figure 6. We observe that SSTD consistently outperforms other baselines by meeting more deadlines. Specifically, the performance gains are very significant when the deadline is tight. This is because SSTD dynamically adjusts its resource and task assignment by monitoring the execution status of jobs in the system. The deadline driven control loop makes real-time adjustments to maximize the opportunity for TD jobs to meet their deadlines, which is one of the optimization goals we discussed in Section II.

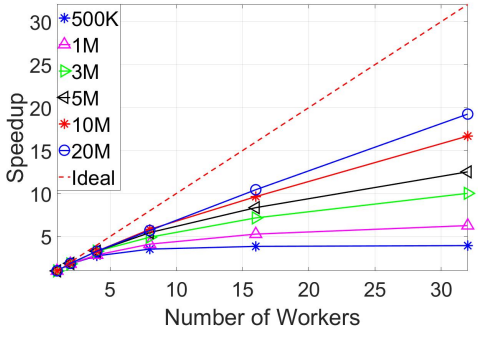


Fig. 7. Scalability of SSTD Scheme

Finally, we evaluate the scalability of SSTD scheme. We generate synthetic data traces of different sizes as the input to the SSTD scheme. We use the metric *speedup* to evaluate the performance gain achieved by SSTD scheme with different number of workers in the system. The Speedup (N) is defined as ratio of serial execution time to execution time on N workers. The ideal Speedup (N) is simply N which is not possible to achieve in practice due to the overhead cost in distributed systems (e.g., communication and I/O overhead). The results are reported in Figure 7. The size of the data trace

is measured in terms of number of tweets. We observe that the speedup ratio improves as the size of the data trace increases. In our evaluation, we push the limit on the size of data traces to be larger than the actual data volume of some extremely large-scale social sensing event in real world applications (e.g., 16.9 million tweets from Super Bowl 2016). The above results demonstrate the scalability of the SSTD scheme.

VI. Related Work

Truth discovery is a critical challenge in social sensing and previous studies have made a significant progress to address this problem [6], [19], [30], [34], [35], [37]. The truth discovery problem is first formally formulated by Yin et al. [34], in which a Bayesian based heuristic algorithm, *Truth Finder*, is proposed. Pasternack et al. proposed extended models (e.g., *AVGLog, Invest* and *PooledInvest*) to incorporate prior knowledge such as constraints on truth and background information into truth discovery solutions [19]. Dong et al. proposed algorithms to handle the source dependency in truth discovery problem [6]. A semi-supervised graph learning scheme is proposed to model the propagation of information truthfulness from the known ground truths [35]. Zhao et al. adopt probabilistic graphical models to solve the truth discovery problem [37]. Wang et al. proposed a maximum-likelihood estimation approach that offers a joint estimation on source reliability and claim correctness [30]. In contrast, we focus on the three emerging challenges (i.e., *dynamic truth*, *scalability* and *heterogeneity of streaming data*) that have not been well addressed by current truth discovery solutions.

There exist some similarities between our work and some previous studies on the topic of dynamic truth. For example, the method proposed in [18] takes into account the evolving information of objects and estimates the truths of variables in current time interval based on the historical claims of sources. Li et al. proposed a Maximum A Posterior based real-time algorithm to explicitly address the evolving truth problem [9]. Wang et al. proposed a method to model time-varying truth using a variant of a Maximum Likelihood Estimation approach [31]. A single-pass truth discovery method was also proposed to handle streaming data [5]. However, none of the above studies consider the scalability and data heterogeneity issues in social sensing, which may cause these schemes to be unscalable and inefficient to large scale social sensing events.

Our work also bears some resemblance to a few distributed system implementations for social sensing applications. For example, Ouyang et al. developed a parallel algorithm for quantitative truth discovery applications data by exploring MapReduce framework in Hadoop [17]. Yerva et al. developed a cloud-serving system for fusing the social and sensor data to deal with massive data streams [33]. Meng et al. proposed a parallel truth discovery solution which is implemented on Hadoop platform to process the large-scale data [15]. Xue et al. introduced a cloud based system for large scale social network analysis using the Hadoop framework [32] as well. However, Hadoop based approaches assume data can be split into chunks of similar sizes based on topics (or claims), which is barely true due to the heterogeneity feature of social sensing data. Second, Hadoop is too heavy weight for time-critical applications that deal with real-time streaming data [17]. In this work, we developed a light-weight distributed framework using Work Queue and HTCondor to improve the efficiency of our truth discovery scheme.

VII. Limitations and Future Work

First, we assume no dependency between claims. There may be cases, however, where claims are not completely independent. For example, weather conditions at city A may be related to weather condition at city B when A and B are close in distance. Incorporating such dependency into our model can be an interesting topic for future research. In particular, we need to explicitly model the correlation between different claims and incorporate such correlation into the HMM based model. The key challenge is to maintain the correlation between claims when the truth discovery task is implemented on a distributed framework.

Second, our scheme requires the labeling of semantic features of the claims, namely the coherence score, independent score and attitude score. In our evaluation, we use heuristic based methods to perform a rough labeling of these scores. We plan to develop accurate classifiers to scale the labeling process by leveraging more refined techniques from Natural Language Processing (NLP) and text mining [2], [3]. For example, the polarity analysis is often used to automatically decide whether a tweet is expressing negative or positive feelings towards a claim. We should note that the SSTD is designed as a general framework where one can easily update or replace components like uncertainty classifier as a plugin of the system.

Third, we use some heuristic rules to control the GCK and LCK in the PID controller. One reason for doing that is finding optimized control solution can be time-consuming and inappropriate for streaming social sensing applications [20]. We plan to explore real-time optimization (RTO) techniques to optimize resource allocation based on control signals. Specifically, we are planning to formulate the system optimization as an integer linear programming (ILP) problem that targets at finding the optimal integer values for the number of workers and the number of tasks for each job in real time [12].

VIII. Conclusion

This paper presents an effective and efficient scheme (i.e., SSTD) to solve the truth discovery problem in social sensing applications. The SSTD scheme addresses the dynamic truth challenge by explicitly modeling the truth transition using a HMM based model. It provides a scalable implementation framework using a distributed system based on HTCondor and Work Queue. SSTD also effectively addresses the heterogeneity of the streaming data by integrating a feedback controller for dynamic task allocation and resource management. We evaluate the SSTD scheme using three real world data traces. The results demonstrate that our solution achieved significant performance gains compared to the state-of-the-art baselines.

REFERENCES

[1] C. C. Aggarwal and T. Abdelzaher. In managing and mining sensor data. *Springer Science & Business Media*, pages 237–297, 2013.

[2] A. M. A. Al-Aziz, M. Gheith, and A. S. Eldin. Lexicon based and multi-criteria decision making (mcdm) approach for detecting emotions from arabic microblog text. In *Proc. First Int. Conf. Arabic Computational Linguistics (ACLing)*, pages 100–105, Apr. 2015.

[3] P. Barnaghi, P. Ghaffari, and J. G. Breslin. Opinion mining and sentiment polarity on twitter and correlation between events and sentiment. In *Proc. IEEE Second Int. Conf. Big Data Computing Service and Applications (BigDataService)*, pages 52–57, Mar. 2016.

[4] P. Bui, D. Rajan, B. Abdul-Wahid, J. Izaguirre, and D. Thain. Work queue+ python: A framework for scalable scientific ensemble applications. In *Workshop on python for high performance and scientific computing at sc11*, 2011.

[5] Z. Z. J. Cheng and W. Ng. Truth discovery in data streams: A single-pass probabilistic approach. In *In Proc. of CIKM*, pages 1589–1598, 2014.

[6] X. L. Dong, L. Berti-Equille, and D. Srivastava. Integrating conflicting data: the role of source dependence. In *Proceedings of the VLDB Endowment*, pages 550–561, 2009.

[7] Q. L. et al. A confidence-aware approach for truth discovery on long-tail data. In *Proceedings of the VLDB Endowment*, volume 8, pages 425–436, Dec. 2014.

[8] X. X. et al. Towards confidence in the truth: A bootstrapping based truth discovery approach. In *Proceedings of the 22th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, 2016.

[9] Y. L. et al. On the discovery of evolving truth. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '15*, 2015.

[10] R. Farkas, V. Vincze, G.Mora, J. Csirik, and G.Szarvas. The conll-2010 shared task: Learning to detect hedges and their scope in natural language text. In *In Proceedings of the Fourteenth Conference on Computational Natural Language Learning.*, 2010.

[11] A. Galland, S. Abiteboul, A. Marian, and P. Senellart. Corroborating information from disagreeing views. In *In Proc. of the ACM International Conference on Web Search and Data Mining (WSDM'10)*, pages 131–140, 2010.

[12] M. Ham, Y. H. Lee, and J. W. Fowler. Integer programming-based real-time scheduler in semiconductor manufacturing. In *Winter Simulation Conference*, pages 1657–1666. Winter Simulation Conference, 2009.

[13] Y. Li, Q. Li, J. Gao, L. Su, B. Zhao, W. Fan, and J. Han. Conflicts to harmony: A framework for resolving conflicts in heterogeneous data by truth discovery. *IEEE Transactions on Knowledge and Data Engineering*, 28(8):1986–1999, Aug. 2016.

[14] M. J. Litzkow, M. Livny, and M. W. Mutka. Condor-a hunter of idle workstations. In *Distributed Computing Systems, 1988., 8th International Conference on*, pages 104–111. IEEE, 1988.

[15] C. Meng, W. Jiang, Y. Li, J. Gao, L. Su, H. Ding, and Y. Cheng. Truth discovery on crowd sensing of correlated entities. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, pages 169–182. ACM, 2015.

[16] nielson. Super bowl 50: Nielsen twitter tv ratings post-game report.

[17] R. W. Ouyang, L. M. Kaplan, A. Toniolo, M. Srivastava, and T. Norman. Parallel and streaming truth discovery in large-scale quantitative crowdsourcing.

[18] A. Pal, V. Rastogi, A. Machanavajjhala, and P. Bohannon. Information integration over time in unreliable and uncertain environments. In *Proceedings of the 21st international conference on World Wide Web*, pages 789–798. ACM, 2012.

[19] J. Pasternack and D. Roth. Knowing what to believe (when you already know something). *Association for Computational Linguistics*, pages 877–885, 2010.

[20] P. Pellegrini, G. Douchet, G. Marlière, and J. Rodriguez. Real-time train routing and scheduling through mixed integer linear programming: Heuristic approach. In *Industrial Engineering and Systems Management (IESM), Proceedings of 2013 International Conference on*, pages 1–5. IEEE, 2013.

[21] J. Qadir, A. Ali, A. Zwitter, A. Sathiaseelan, J. Crowcroft, et al. Crisis analytics: Big data driven crisis response. *arXiv preprint arXiv:1602.07813*, 2016.

[22] L. Rabiner and B. Juang. An introduction to hidden markov models. *ieee assp magazine*, 3(1):4–16, 1986.

[23] B. T. Rao, N. Sridevi, V. K. Reddy, and L. Reddy. Performance issues of heterogeneous hadoop clusters in cloud computing. *arXiv preprint arXiv:1207.0894*, 2012.

[24] D. E. Rivera, M. Morari, and S. Skogestad. Internal model control: Pid controller design. *Industrial & engineering chemistry process design and development*, 25(1):252–265, 1986.

[25] T. Shelton, A. Poorthuis, M. Graham, and M. Zook. Mapping the data shadows of hurricane sandy: Uncovering the sociospatial dimensions of 'big data'. *Geoforum*, 52:167–179, 2014.

[26] M. Y. S. Uddin, M. T. A. Amin, H. Le, T. Abdelzaher, B. Szymanski, and T. Nguyen. On diversifying source selection in social sensing. In *Proc. Ninth Int Networked Sensing Systems (INSS) Conf*, pages 1–8, June 2012.

[27] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, Apr. 1967.

[28] D. Wang, M. T. Amin, S. Li, T. Abdelzaher, L. Kaplan, S. Gu, C. Pan, H. Liu, C. C. Aggarwal, R. Ganti, X. Wang, P. Mohapatra, B. Szymanski, and H. Le. Using humans as sensors: An estimation-theoretic perspective. In *Proc. 13th Int Information Processing in Sensor Networks Symp. IPSN-14*, pages 35–46, Apr. 2014.

[29] D. Wang, L. Kaplan, and T. F. Abdelzaher. Maximum likelihood analysis of conflicting observations in social sensing. *ACM Transactions on Sensor Networks*, 10(2):1–27, Jan. 2014.

[30] D. Wang, L. Kaplan, H. Le, and T. Abdelzaher. On truth discovery in social sensing: A maximum likelihood estimation approach. In *Proc. ACM/IEEE 11th Int Information Processing in Sensor Networks (IPSN) Conf*, pages 233–244, Apr. 2012.

[31] S. Wang, D. Wang, L. Su, L. Kaplan, and T. F. Abdelzaher. Towards cyber-physical systems in social spaces: The data reliability challenge. In *Proc. IEEE Real-Time Systems Symp. (RTSS)*, pages 74–85, Dec. 2014.

[32] W. Xue, J. Shi, and B. Yang. X-rime: cloud-based large scale social network analysis. In *Services Computing (SCC), 2010 IEEE International Conference on*, pages 506–513. IEEE, 2010.

[33] S. R. Yerva, H. Jeung, and K. Aberer. Cloud based social and sensor data fusion. In *Information Fusion (FUSION), 2012 15th International Conference on*, pages 2494–2501. IEEE, 2012.

[34] X. Yin, J. Han, and P. S. Yu. Truth discovery with multiple conflicting information providers on the web. *IEEE Transactions on Knowledge and Data Engineering*, 20(6):796–808, June 2008.

[35] X. Yin and W. Tan. Semi-supervised truth discovery. In *Proceedings of the 20th international conference on World wide web*, number 217-226. ACM, 2011.

[36] D. Zhang, H. Rungang, and D. Wang. On robust truth discovery in sparse social media sensing. In *Big Data (Big Data), 2016 IEEE International Conference on*. IEEE, 2016.

[37] B. Zhao and J. Han. A probabilistic model for estimating real-valued truth from conflicting sources. In *Proc. of QDB*, 2012.

[38] B. Zhao, B. I. P. Rubinstein, J. Gemmell, and J. Han. A bayesian approach to discovering truth from conflicting sources for data integration. In *Proceedings of the VLDB Endowment*, volume 5, pages 550–561, 2012.

IX. APPENDIX

A. E and M Steps of EM Algorithm for HMM Parameter Estimations

The E and M steps of the EM algorithm in HMM based solution in Section III are derived as follows.

E-Step: In the E-step, we first calculate the probability that the hidden true value is V_i when data ACS_u^t was observed using:

$$\gamma_{u,t,i} = \frac{\alpha_{u,t,i}\beta_{u,i,t}}{\sum_{i=1}^{2} \alpha_{u,t,i}\beta_{u,i,t}} \tag{13}$$

where $\alpha_{u,t,i}$ and $\beta_{u,i,t}$ are defined as follows:

$$\alpha_{u,t,i} = b_{u,i,t} \sum_{j=1}^{2} \alpha_{u,t-1,j} a_{u,j,i} \qquad \alpha_{u,1,i} = b_{u,i,1} a_{u,j,i}$$
$$\beta_{u,t,i} = \sum_{j=1}^{2} \beta_{u,t+1,j} a_{u,i,j} b_{u,j,t+1} \qquad \beta_{u,T,i} = 1 \tag{14}$$

Then we compute the probability that a transition from true value V_i to V_j occurred between time instants t and $t+1$ based on the following equations:

$$\begin{aligned} \zeta_t(u,i,j) &= \frac{\alpha_{u,t,i} \beta_{u,t+1,j} a_{u,i,j} b_{u,j,t+1}}{P(F(u)|\lambda_u)} \\ &= \frac{\alpha_{u,t,i} \beta_{u,t+1,j} a_{u,i,j} b_{u,j,t+1}}{\sum_{i=1}^{2} \alpha_{u,t,i} \beta_{u,i,t}} \end{aligned} \tag{15}$$

M-Step: In the M-Step, we use γ and ζ to recompute the model parameters $\overline{\lambda_u} = \{\overline{a_{u,i,j}}, \overline{b_{u,i,t}}, \overline{\pi_{u,i}}\}$ as follows:

$$\begin{aligned} \overline{a_{u,i,j}} &= \frac{\sum_{t=1}^{T-1} \zeta_t(u,i,j)}{\sum_{t=1}^{T} \gamma_{u,t,i}} \\ \overline{b_{u,i,t}} &= \frac{\sum t \in \{t|O_t = ACS_u^t\} \gamma_{u,t,i}}{\sum_{t=1}^{T} \gamma_{u,t,i}} \\ \overline{\pi_{u,i}} &= \gamma_{u,1,i} \end{aligned} \tag{16}$$

B. LCK and GCK Tuning in PID Controller

We present the details of the LCK and GCK tuning process. We first define the key states of the system based such control signals.

- Bad State: a state that is indicated by all positive PID signals (i.e., execution time is larger than deadlines). In this state, all TD jobs cannot meet their deadlines.
- Good State: a state that is indicated by all negative PID signals i.e., execution time is smaller than deadlines). In this state, all TD jobs can meet their deadlines.
- Unsychronized Good State: a vast majority of PID signals are negative, with few being positive. In this state, we only have few slow jobs.
- Unsychronized Bad State: a vast majority of PID signals are positive, with few being negative. In this state, we have many slow jobs.

Based on the current state of the system, DTM dynamically adjusts control knobs (i.e., LCK and GCK) based using the following rubrics:

- If the system is in a Bad State, DTM tunes the GCK by increasing the total number of workers by a factor of θ_3 as long as it does not exceed the number of available workers in the system.
- If the system is in Good State, DTM does nothing.
- If the system is in Unsychronized Good State, DTM tunes LCK by increasing the number of tasks for jobs with positive control signals (i.e., slow jobs) and decreasing the number of tasks jobs with negative control signals (i.e., fast jobs).
- If the system is in Unsychronized Bad State, DTM increases the number of worker by a factor of θ_4 as long as it does not exceed the number of available workers.

We also note that the adjustment of the Worker Pool (i.e., GLK) requires stopping all the current tasks which introduces extra overhead to the system. Therefore, we only increase the number of workers after the system is constantly in a Bad state or Unsychronized Bad State. We only decrease the number of workers when the system is constantly in Good State and the PID signals are above a threshold (which indicates the system is way faster than the deadline requirement.).