Efficient Online Summarization of Large-Scale Dynamic Networks

Qu, Qiang; liu, siyuan; zhu, feida; Jensen, Christian Søndergaard

Published in:
I E E E Transactions on Knowledge & Data Engineering

DOI (link to publication from Publisher):
10.1109/TKDE.2016.2601611

Publication date:
2016

Document Version
Publisher's PDF, also known as Version of record

Link to publication from Aalborg University

Citation for published version (APA):
Qu, Q., liu, S., zhu, F., & Jensen, C. S. (2016). Efficient Online Summarization of Large-Scale Dynamic Networks. *I E E E Transactions on Knowledge & Data Engineering*, *28*(12), 3231-3245. https://doi.org/10.1109/TKDE.2016.2601611

Efficient Online Summarization of Large-Scale Dynamic Networks

Qiang Qu, Siyuan Liu, Feida Zhu, and Christian S. Jensen, *Fellow, IEEE*

Abstract—Information diffusion in social networks is often characterized by huge participating communities and viral cascades of high dynamicity. To observe, summarize, and understand the evolution of dynamic diffusion processes in an informative and insightful way is a challenge of high practical value. However, few existing studies aim to summarize networks for interesting dynamic patterns. Dynamic networks raise new challenges not found in static settings, including time sensitivity, online interestingness evaluation, and summary traceability, which render existing techniques inadequate. We propose dynamic network summarization to summarize dynamic networks with millions of nodes by only capturing the few most interesting nodes or edges over time. Based on the concepts of diffusion radius and scope, we define interestingness measures for dynamic networks, and we propose OSNet, an online summarization framework for dynamic networks. Efficient algorithms are included in OSNet. We report on extensive experiments with both synthetic and real-life data. The study offers insight into the effectiveness, efficiency, and design properties of OSNet.

Index Terms—Dynamic networks, network cascades, graph summarization, diffusion process, interestingness, graph mining

1 Introduction

Driven by the ever-increasing sizes of real-world networks, the problem of network summarization has never been more important. While most existing studies consider the summarization of static networks according to criteria such as compression ratio, network representation, minimum loss, and visualization friendliness [1], [2], [3], recent developments in social network mining and analysis [4], [5], location-based services [6], [7], [8] and bioinformatics [3] have given prominence to the study of a new kind of dynamic network [9], [10], [11] that captures progressive information diffusion processes in an underlying network.

An information diffusion process in a network can be represented by a stream of interactions between node instances, namely time-stamped pairs of nodes from the underlying network, denoting the information propagation from one node to the other at the time as indicated by the associated timestamp [12], [13], [14]. An example of a diffusion process is the spread of news items among Twitter users by means of the "retweet" functionality. Such a stream describes a *dynamic network* where a diffusion process grows with each incoming node instance pair as a new interaction in the stream.

The summarization task of such a dynamic network, which is significantly different from that of a static one, poses new research challenges. The critical difference lies in that, for a dynamic diffusion process, it is valuable to capture each "interesting" development as the process evolves, in an online fashion. This problem, termed as *dynamic network summarization (*DNS*)*, has a wide range of applications, among which we highlight several as follows.

In information visualization, massive dynamic networks are hard to visualize due to their huge sizes and complicated evolution [15]. DNS makes it possible to create online, time-labeled summaries in the form of "trajectories" to enable closer examination of important changes in a diffusion process as it evolves. In social network studies, DNS offers the identification of interesting dynamics in the form of "backbones" that describe key information propagation flow and give insight to the evolving roles of different participants. This is useful for tasks such as change detection [16]. In road traffic analysis, DNS can capture major traffic flows and population movement. Summaries for given periods can be projected onto the road network to detect traffic thoroughfares, and benefit road planning, urban management and human mobility analysis [17].

Given a diffusion process, a straightforward approach to obtain a summary is to periodically compute a summary from the evolving process. Thus, the summary is represented as a sequence of summaries of static networks each aggregating edges and nodes in a time interval of size Δt (i.e., sliding window) [15]. However, this approach is costly when networks are large. Further, the parameter Δt is fundamentally hard to set—too small a value would compromise the performance and too large miss important diffusion dynamics. Even if an appropriate Δt is available, most of the previous methods, which were designed to target other criteria in the first place, fall short in producing results that would capture and reveal interesting dynamics.

Two useful phenomena observed from social network studies have motivated us to summarize dynamics based on interestingness. First, user interactions on social networks may show the interestingness of time-stamped posts.

• *Q. Qu is with the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China. E-mail: qiang@siat.ac.cn.*
• *S. Liu is with the Smeal College of Business, Pennsylvania State University, State College, PA 16801. E-mail: siyuan@psu.edu.*
• *F. Zhu is with the School of Information Systems, Singapore Management University, Singapore 188065. E-mail: fdzhu@smu.edu.sg.*
• *C.S. Jensen is with the Department of Computer Science, Aalborg University, Aalborg 9100, Denmark. E-mail: csj@cs.aau.dk.*

Manuscript received 21 Nov. 2014; revised 15 May 2016; accepted 2 Aug. 2016. Date of publication 24 Aug. 2016; date of current version 2 Nov. 2016.
Recommended for acceptance by K. Chang.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TKDE.2016.2601611

For example, the release of a new Nintendo DS game as a time-stamped post on Twitter can induce many users' tweets [14]. We consider such a time-stamped post as a node, and the measure of the number of user interactions as node degree has been used for finding hot topics, detecting burst events, and mining influential users. Second, social networks are born for social conversations. Interesting topics often provoke long and active conversation chains among a group of users [13], [18], [19]. For instance, such a long conversation chain is quite common for interesting questions on Stackoverflow,[1] where users have multiple QA discussions on a specific problem to achieve a solution. Multiple involvement of one user in a conversation chain (i.e., node instances) often encourage the participation of other users [20]. Ideally, a conversation chain is traceable such that we can follow conversations from the start to a particular node instance. The traceability is straightforward to demonstrate how users interact, e.g., on Stackoverflow, to show a problem and its responses, traceable chains enable us to follow user discussions. The measure of user conversations may have various applications for customer satisfaction survey, user engagement study, and fraud user detection. This study considers the two measures, which intuitively could help us find the summaries where users are influential and actively engaged through particular instances in social activities. Note that in information diffusion processes, interestingness is associated with user instances. For example, in Twitter, users are interested in particular tweets posted at specific time [21]. We regard a user's particular tweet as one of the user instances.

1.1 Research Challenges

We identify the following research challenges in the task of DNS. *(1) Time Sensitivity.* Diffusion processes often represent vast, viral, and unpredictable processes, e.g., breaking news and bursty events [22]. As a result, the rate of diffusion can vary drastically over a short period of time [23]. It is a big challenge to respond adaptively to dynamics and to achieve timely summarizations. *(2) Online Interestingness Evaluation.* A key challenge here is to capture the most interesting nodes and edges in a summarization. Compared with traditional network summarization, interestingness evaluation in DNS assumes an extra degree of difficulty because of the partial view of the network at any given time of evaluation. *(3) Summary Traceability.* An important goal is to enable a better understanding of the evolution of a diffusion process throughout its life cycle. A good summary should reveal the flow of the dynamics such that interesting developments can be traced.

1.2 Our Approach and Contribution

To tackle the DNS problem, we propose OSNet, a framework for online summarization of dynamic networks that aims to produce concise, interestingness-driven summaries that capture the evolution of diffusion processes. Our contribution is summarized as follows. 1) Unlike previous proposals that apply optimization criteria in offline settings, we consider a setting where network summarization occurs online, as the diffusion process evolves. 2) Based on the concepts of propagating radius *proRadius* and propagating scope *proScope*, we formalize the problem of characterizing the interesting dynamics of an evolving diffusion process in a traceable manner. 3) We propose OSNet that encompasses online and incremental dynamic network summarization algorithms on a spreading-tree model. In terms of entropy, OSNet archives the best summaries with respect to informativeness. 4) We propose two efficient algorithms, a tree-based and a Bloom filter based algorithm. The false positive probability of the proposed Bloom filter based algorithm is bounded. 5) Extensive experiments are conducted with both synthetic and real-life datasets including case studies for different applications as well as a user study to demonstrate the effectiveness and efficiency of the proposed scheme.

1. http://www.stackoverflow.com

TABLE 1
Key Symbol Summary

Symbol	Description
$\mathcal{D}(G)$	A diffusion process on a network G
L	A labeling function.
$x(\delta, u, v, t)$	Interaction x: δ is diffused from u to v at time t
$\mathcal{I}(G)$	A seed node set $\{v_r\}$ of a diffusion process $\mathcal{D}(G)$
$l(v)$	The diffusion path from the root to node instance v
d	The max propagation radius or the depth of a tree
$y(v)$	Propagation radius of a node v in a diffusion process
$w(v)$	Propagation scope of a node v in a diffusion process
$deg^+()$	The number of direct infectees of a node
α	Control parameter for $w(v)$
$\xi_t(v)$	Interestingness of node instance v at time t
τ	Interestingness threshold
$\mathbf{S}(C)$	Traceable interesting summary
ϱ	Acceleration Intensity
H	Entropy
T	A spreading tree
$\mathbf{S}(G)$	A set of traceable interesting summaries
η	The number of chosen labeled nodes for a dataset

2 Problem Statement

Table 1 summaries the key symbols used in the paper.

The input to the problem is a stream of time ordered interactions representing diffusion processes on a network G. A diffusion process on a network G, denoted by $\mathcal{D}(G)$, is a stream of time-ordered interactions. An interaction $x = (\delta, u, v, t) \in \mathcal{D}(G)$ indicates that a specific story is diffused from node u to node v at time $t \in \mathcal{T}$. Note that u and v in x are node instances associated with the specific timestamp t. A story is defined by a textual keyword list used to describe an event, such as breaking news in Twitter. The diffusion from u to v captures that node v receives the story from u. We also say that u is an infector of v while v is an infectee of u. We call time t the infection time of node v. Note that a diffusion process of a story can be initiated by different nodes that are regarded as seeds or roots. For each interaction x, we further define δ to be a three-tuple as a canonical identifier, i.e., $\delta = (storyID, v_r, t')$, where $storyID$ is the identity of the diffusing story, v_r represents the seed node starting the diffusion, and t' is the infection time of the infector u. The diffusion process from a seed over a time period forms a time-stamped graph, known as a network cascade C [10], [16] where each interaction is a directed edge from the infector to the infectee. The output of a dynamic process as a summary is a subset of interactions with connected node instances of the process.

Definition 1 [Cascade C]. *A cascade C is a directed graph $C = (V_C, E_C, L_{V_C}, L_{E_C})$, representing a diffusion process $\mathcal{D}(G) = \{x = (\delta, u_i, v_i, t_i)\}$ of a story propagated from a seed v_r during a time period $\mathcal{T}$. The node set is $V_C = \cup u_i + \cup v_i$, and the edge set is $E_C = \cup_x (u_i, v_i)$. A node pair (u_i, v_i) for each x is considered as a directed edge from u_i to v_i. $L_{V_C} : V_C \mapsto \Sigma$ is a node labeling function to label nodes, and $L_{E_C} : E_C \mapsto \mathcal{T}$ is an edge labeling function associated with time-stamps.*

A network G with diffusion processes is termed a diffusion network or a dynamic network, which, for simplicity, we also denote by G. Given a diffusion network G, a set $\mathcal{I}(G) \subseteq V(G)$ is given that contains the seed nodes from which a diffusion starts. The infection time of a seed v_r is given as t_{v_r}. We use $\circ +_{(v_r)} (u)$ to denote the number of direct infectees of a node u for a diffusion process from a seed v_r in a cascade C.

Before we present the definition of interestingness, two measures are introduced to evaluate nodes in a dynamical process by i) how far the information can travel (Measure 1: depth) and ii) how many infectees a node can have (Measure 2: breadth). These two measures can be used for capturing the *interestingness* of a diffusion process for three reasons : 1) The two measures agree with intuition. 2) The two measures capture the cascade, enabling reconstruction with little more information. 3) The two measures offer a foundation for computing different properties of a cascade. In addition, we observe that other studies also suggest that the two measures can characterize diffusion processes [18], [24].

Measure 1 [Propagation Radius (proRadius)]. The propagation radius of a node v in a diffusion process from a root (represented by a cascade C), denoted by $y(v)$, is the length of the path $l(v)$ from the root of C to v, $|l(v)|$. The maximum propagation radius of a node in C is the diameter of C: $d(C) = \max(y(v))$. Note that the propagation radius of the root is 0.

Measure 2 [Propagating Scope (proScope)]. The *proScope*, $w(v) = deg^+(v)$, of a node v in a diffusion process from a root (represented by a cascade C), is the number of infectees of v in C.

Definition 2 [Interestingness]. *We represent a node v by a vector $(y(v), w(v))$ and use Equation (1) to quantify the total interestingness of the node. As the degree distribution of many networks follows a power-law, we use a log value of the* proScope

$$\xi(v) = \alpha \log w(v) + (1 - \alpha) y(v), \tag{1}$$

where $\alpha \in [0, 1]$ balances the two measures. We set $\log w(v) = 0$ if $w(v) = 0$. Note that cascades evolve over time as interactions arrive. We thus use $\xi_t(v)$ to denote the interestingness of a node v at time t, which is calculated using the values of proScope *and* proRadius *of v at t.*

Definition 3 [Interesting Summary $S(C)$]. *Given a cascade C and a threshold τ at time t, an interesting summary $S(C)$ is a subgraph of C satisfying that for any node $v_i \in S(C)$, $\xi_t(v_i) > \tau$ holds; for two nodes u and v in $V(S(C))$, the edge $e' = (u, v)$ exists in $S(C)$ if and only if $e = (u, v)$ exists in C. Labels of the edges and nodes in $S(C)$ retain the labels they have in C.*

Definition 4 [Traceable Interesting Summary $\mathbf{S}(C)$]. *Given an interesting summary $S(C) \subset C$ at time t, a traceable interesting summary $\mathbf{S}(C)$ is a super-graph of $S(C)$, denoted $S(C) \subset \mathbf{S}(C)$. A node v_i in C is in $\mathbf{S}(C)$ if: v_i is the seed, or $\xi_t(v_i) > \tau \vee (\exists v_j \in C, (v_i \in l(v_j) \wedge \xi_t(v_j) > \tau))$.*

As some nodes are removed from an interesting summary (Definition 3), remaining interesting nodes may become disconnected. Definition 4 includes the missed nodes on the paths from the seed to the remaining interesting nodes. A traceable interesting summary thus is possible to reveal the flow of dynamics and interesting developments can be traced throughout their life cycle. To explain the evolution in a traceable interesting summary, we next introduce the concepts *diffusion rise* and *diffusion decay*, defined by the notion of acceleration intensity. In the rest of the paper, we use a summary (summaries) to indicate a traceable interestingness summary (summaries) for simplicity.

Definition 5 [Acceleration Intensity ϱ]. *Given a node v_i as an infector of a node v_j in a cascade C, the acceleration intensity is defined based on the diffusion from v_i to v_j in C as*

$$\varrho(v_i, v_j) = \frac{\xi_{t_j}(v_j) - \xi_{t_i}(v_i)}{|t_j - t_i|}, \tag{2}$$

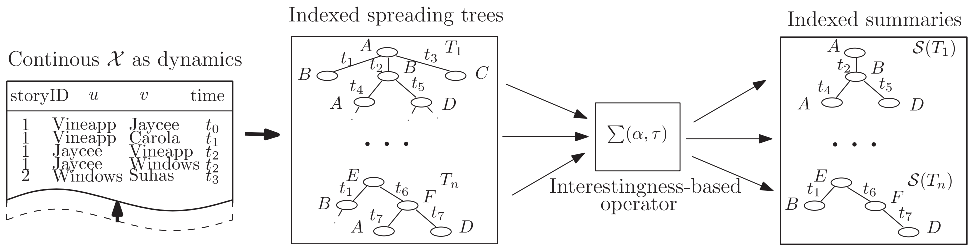
where t_i and t_j are the infection times of v_i and v_j, respectively.

We can now define the rise and decay of a diffusion process: When $\varrho > 0$, the propagation process from v_i to v_j is a diffusion rise process; otherwise, it is a diffusion decay process.

The goal of the DNS problem is to better understand network dynamics. A summary thus needs to be informative with respect to the original data. There are several methods to evaluate informativeness. Among these, we propose to use *Entropy*. A review of *Shannon Entropy* and details are presented in Section 3.3.1. Here we denote the entropy of a traceable interesting summary $\mathbf{S}(C)$ by $H(\mathbf{S}(C))$. Recall that the entropy gains when its value decreases. We thus aim to find a summary with minimal entropy to achieve the best informativeness. The problem is stated as follows:

Problem Statement (Interestingness-driven Diffusion Process Compression). Given a diffusion network G with seed sets $\cup\mathcal{I}(G)$, stories diffuse from each seed over time. The dynamic process is represented by a stream of interactions, which forms a set of cascades $\{\ldots, C_i, \ldots\}$. The output of the problem at time t is a set of traceable interesting summaries $\mathbf{S}(G) = \{\ldots, \mathbf{S}_i(C_i), \ldots\}$ ($|\mathbf{S}_i(C_i)| > 0$). The entropy ($H(\mathbf{S}(C))$) of each summary $\mathbf{S}_i(C_i)$, which reveals diffusion rise and decay, is minimized subject to the balancing parameter $0 \leq \alpha \leq 1$ of the aggregate score and the interestingness threshold $\tau \geq 0$.

To solve the problem, two sub-problems have to be solved: i) How to model the dynamics on the top of graphs? Is the cascade model suitable? The diffusion processes we discuss are evolving over time. And all the cascades on a node are merged. This may cause problems for the summarization because the interestingness of a node is associated with time-stamps and stories as node instances. This requires


Continous $\mathcal{X}$ as dynamics
Indexed spreading trees
A B C D t_1 t_2 t_3 t_4 t_5 T_1
E B F A D t_1 t_6 t_7 T_n
$\sum(\alpha, \tau)$
Interestingness-based operator
Indexed summaries
$\mathcal{S}(T_1)$
$\mathcal{S}(T_n)$

storyID	u	v	time
1	Vineapp	Jaycee	t_0
1	Vineapp	Carola	t_1
1	Jaycee	Vineapp	t_2
1	Jaycee	Windows	t_2
2	Windows	Suhas	t_3

Fig. 1. Overview of the O$SNet$ framework.

to design a labeling function to distinguish the node instances, which is ineffective. ii) How to set proper values for α and τ for different diffusion processes? Given an α in the range $[0, 1]$, each connected subgraph of a cascade C over time can be a summary, which yields a hard graph decomposition problem. On the other hand, the scale of a summary mostly depends on the threshold τ. A proper value is necessary because we intend to find all interesting developments. We proceed to develop the OSNet framework that encompasses new and incremental techniques capable of continuously summarizing dynamics based on a spreading tree model in step with the evolution of diffusion processes.

3 Our Method

3.1 Framework Overview

An overview of OSNet is shown in Fig. 1. The input is a diffusion process $\mathcal{D}(G)$. Instead of using cascades, we model the interactions by a set of indexed spreading trees. A collection of spreading trees is equivalent to a cascade network. There are indexes on storyID and seeds, such that we can insert an interaction into a spreading tree T_i efficiently. By Equation (1), the interestingness-based operator is to evaluate the interestingness of nodes in spreading trees with two parameters, α and τ. We evaluate the interestingness of a node v when it infects new nodes (i.e., $w(v)$ increases). If v has $\xi_t(v) > \tau$, it is inserted into a summary $\mathbf{S}(T_i)$. The summaries are also indexed in the same way as T. We thus insert v into $\mathbf{S}(T_i)$ by searching storyID and seed. Once a node v is inserted into tree T, it is tagged with its branch such that a node cannot be reinserted into the summary $\mathbf{S}(T_i)$. We only insert new nodes and edges into a tree over time, and it is not necessary to rebuild any part of T or $\mathbf{S}(T)$.

When a node v of T_i is to be inserted into $\mathbf{S}(T_i)$ at time t, there exist three cases: 1) $\mathbf{S}(T_i)$ does not exist and v is not a seed ($v \notin \mathcal{I}(G)$); 2) $\mathbf{S}(T_i)$ exists and the infector of v in T_i is already in $\mathbf{S}(T_i)$; 3) $\mathbf{S}(T_i)$ exists and the infector of v in T_i is not in $\mathbf{S}(T_i)$. Cases 1) and 2) are straightforward. We can create a new tree for case 1); and for case 2), we insert v as a child of its infector in $\mathbf{S}(T)$. In case 3), the insertion of v renders $\mathbf{S}(T_i)$ disconnected, and the process thus cannot be traced from the seed to v. A solution is to recover all the nodes in the path from the root to v. We call this problem the *Recovery Problem*.

We proceed to present the spreading-tree model in Section 3.2. Parameters used for interestingness evaluation are discussed in Section 3.3. Two algorithms for the recovery problem are proposed in Sections 3.4 and 3.5.

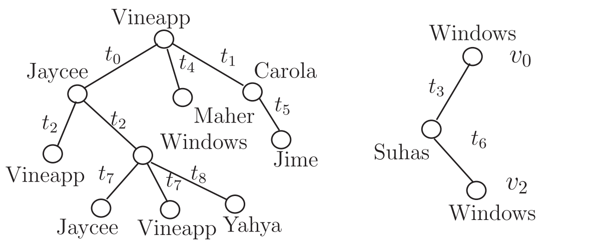
3.2 Spreading-Tree Model

Although network cascades can model diffusion processes, several issues of dynamics challenge the effectiveness of network cascade model. First, the time-stamped instances of a node are merged in cascades because a node in a cascade model can only appears once [10]. However, in dynamic networks, the interestingness is defined on node instances. Namely, a node becoming interesting is associated with a specific time in an interaction. To distinguish node instances in different interactions and cascades, a cascade model requires a labeling scheme as extra effort. Furthermore, as cascades are directed graphs, there exist backward and forward edges or even cycles. This makes a cascade hard to interpret and navigate between node instances. Second, summary search on cascades can be regarded as subgraph search. However, graph search is usually time-consuming since it involves isomorphism checks. Third, since cascades are merged into one directed graph, the graph search space grows exponentially, which makes dynamic summarization even harder. We propose to instead use a *Spreading-Tree* model. First, spreading trees are constructed directly by interactions without any other efforts. A spreading tree models an individual diffusion process. Information is diffused from the root to the leaves. The model distinguishes interactions and cascades by itself. Next, tree search is relatively efficient. Numerous proposals of efficient tree operations (e.g., update) exist. Third, there are no backward and forward edges in spreading trees. The tree structure is not as complex as a cascade. The search space is proportional to the scale of the interactions.

Definition 6 [Spreading Tree T]. *A spreading tree $T = (v_r, V', E', L_{V'}, L_{E'})$, is a rooted and labeled n-ary tree, where $v_r \in V'$ is the root, V' is a multiset of node instances, $E' \subseteq V' \times V'$ is a set of edges, $L_{V'} : V' \mapsto \Sigma$ is node labeling function, and $L_{E'} : E' \mapsto \mathcal{T}$ is an edge labeling function.*

Intuitively, a node represents a specific user instance in a diffusion process, and the node's label is the identity of the user; an edge in a spreading tree connects an infector node with an infectee node, and the edge's label is the infection time of the infectee node. A non-root node has one infector. A non-leaf node has one or more infectees, and a leaf node has no infectees.

Given a diffusion network G, each seed $v_r \in \mathcal{I}(G)$ forms the root of a spreading tree. When an interaction $x = (\delta, u, v, t) \in \mathcal{D}(G)$ arrives, the spreading tree for δ is updated by inserting a *new* node labeled v and an edge labeled t from an *existing* node labeled u to v. Note that both u and v are labels of the nodes. To find the existing node u, we search the tree in breadth-first order starting from the root until a node with label u and infection time t is found. Therefore, although multiple nodes have the same label, the three-tuple δ can determine from which node to insert the edge to the new infectee.



(a) T_1 by t_8 for storyID=1 (b) T_2 by t_8 for storyID=2

Fig. 2. Spreading tree.

To illustrate the construction procedure, assume that a new interaction $x = (\delta, Jaycee, John, t)$ is to be inserted into the tree in Fig. 2a. Note that we need to be able to determine whether to insert a node John as an infectee of the leftmost node labeled Jaycee or the leaf one.

In Twitter, we can use "@mention" tags to form δ and to identify the correct existing node where to insert an edge for an interaction. For example, as shown in Fig. 3, when Jaycee tweets with "@John" as a reply to the tweet by Vineapp at t_0, John is then the infectee of the leftmost Jaycee under Vineapp instead of Windows in Fig. 2a. Fig. 2 shows two spreading trees extracted from Twitter. Each edge represents an interaction x, and each tree captures the spread of a story. It follows from the tree construction procedure that multiple nodes may have the same label (e.g., Windows in tree T_2). It also follows that there are no cycles [10], [25].

In summary, the spreading-tree model achieves the following properties: 1) cascades can be modeled as spreading trees such that the summarization on cascades equals the task on spreading trees; 2) the trees are separated by seeds; 3) a node can be duplicated in a spreading tree, which shows that the model distinguishes node instances; 4) the size of the trees is proportional to the scale of the interactions; 5) infection occurs top-down, and diffusion occurs from a parent node to a child node.

3.3 Self-Adjusted Parameters

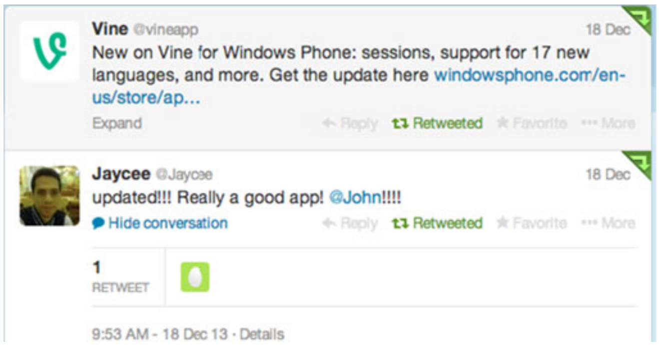
Although using fixed values for parameters is simple to implement, there are two main issues that demand better approaches. First, for a single diffusion process, prediction of the network statistics (change rate, number of infectees, propagating range, etc.) is usually difficult. It is hard to find parameter settings that can best capture the dynamics. Second, different diffusion processes vary substantially in range and scope. Thus, the same settings are not likely to work across different processes. Our study aims to provide a self-tuning mechanism that adapts to differences in the summarization of dynamics.

3.3.1 Alpha Estimation

Recall that the entropy H of a random variable E with possible values $\{e_1, \ldots, e_n\}$ is defined as

$$H(E) = -\sum_{i}^{n} p(e_i) log_2 p(e_i), \tag{3}$$

where $p(e_i)$ is the probability mass function of outcome e_i. $H(E)$ is close to 0 if the distribution is highly skewed and informative.



Vine @vineapp 18 Dec
New on Vine for Windows Phone: sessions, support for 17 new languages, and more. Get the update here windowsphone.com/en-us/store/ap...
Expand Reply Retweeted Favorite More

Jaycee @Jaycee 18 Dec
updated!!! Really a good app! @John!!!!
Hide conversation Reply Retweeted Favorite More
1 RETWEET
9:53 AM - 18 Dec 13 · Details

Fig. 3. A user interaction example from Twitter.

We measure the entropy of a summary $\mathbf{S}(C)$ and aim to maximize the informativeness of $\mathbf{S}(C)$ to have the maximum possible information out of T. Given a set of continuous interactions $\mathcal{D}(G)$ by time t (denoted by $\mathcal{D}(G)_t$), the probability of diffusing story κ from v_i to v_j is regarded as a conditional probability based on the percentage of v_j was an infectee out of all the interactions in set $D(G)_t$

$$p_{(\kappa,t)}(v_j|v_i) = p_{(\kappa,t)}(v_i) \times \frac{\sum^{\mathcal{D}(G)_t} f_{(\kappa,t)}(v_j, x)}{|\mathcal{D}(G)_t|}, \tag{4}$$

where $p_{(\kappa,t)}(v_i)$ is the probability of the node that directly infects v_j obtained by the occurrences of v_i in spreading κ by t . Note that for a seed node, the probability of its infector is 1 in order to guarantee that a root is infected. The function $f_{(\kappa,t)}(v_j, x)$ is an indicator that is 1 if v_j is an infectee in interaction $x \in \mathcal{D}(G)$ when $storyID = \kappa$ by time t, otherwise 0. Then we use the entropy $H_{p(\kappa,t)}(\mathbf{S}(C))$ as an informativeness measure of a summary $\mathbf{S}(C)$ with respect to T

$$H_{p(\kappa,t)}(\mathbf{S}(C)) = -\sum_{j=1}^{|\mathbf{S}(C)|} p_{(\kappa,t)}(v_j|v_i) \log p_{(\kappa,t)}(v_j|v_i). \tag{5}$$

Thus, $\mathbf{S}(C)$ is the most informative by time t with respect to T if the value of its entropy $H_{p(\kappa,t)}$ is minimized. Before we present the details of the estimation, Lemma 1 is introduced as a property of a summary's entropy.

Lemma 1. *If two summaries $\mathbf{S}(T)$ and $\mathbf{S}'(T)$ satisfy $d(\mathbf{S}(T)) > d(\mathbf{S}'(T))$, $V'(\mathbf{S}'(T) \backslash \mathbf{S}(T)) = \emptyset$, and $|l(v)| > d(\mathbf{S}'(T))$ where $v \in V'(\mathbf{S}(T) \backslash \mathcal{S}'(T))$, then we have $H_{p(\kappa,t)}(\mathbf{S}(T)) \leq H_{p(\kappa,t)}(\mathbf{S}'(T))$ holds.*

The proof is omitted due to the space limitation. It shows that the entropy is smaller for those summaries with greater depth. By Equation (1), to achieve the smallest entropy, we need to minimize α because a smaller α yields a higher weight for depth such that deep summaries are preferred. In the remainder of the section, we present the bounds on α followed by our estimation based on entropy.

Theorem 1. *Let n as the maximal number of nodes in a summary $\mathbf{S}(T)$ with a threshold τ. Given $\log \sqrt[d]{n-1} - d > 0$, the lower bound of α is*

$$\alpha \geq \frac{\tau - d}{\log \sqrt[d]{n-1} - d}, otherwise\ 0, \tag{6}$$

where d is the depth of the summarized tree.

Proof. Given a node v in a summarized tree $\mathbf{S}(T)$, we have $\alpha \log w(v) + (1-\alpha)y(v) > \tau$. The maximal value of $y(v)$ is the depth of the summarized tree d such that the fanout is lower-bounded by $e^{(\tau-(1-\alpha)d)/\alpha}$, which is positive. Since the size of $\mathbf{S}(T)$ is bounded by n, $(e^{(\tau-(1-\alpha)d)/\alpha})^d + \cdots + e^{(\tau-(1-\alpha)d)/\alpha} + 1 = \sum_{i=0}^{d} (e^{(\tau-(1-\alpha)d)/\alpha})^i \leq n$ holds. As $e^{(\tau-(1-\alpha)d)/\alpha}$ is positive, $(e^{(\tau-(1-\alpha)d)/\alpha})^d + 1 \leq \sum_{i=0}^{d} (e^{(\tau-(1-\alpha)d)/\alpha})^i \leq n,$. Transforming the inequality $(e^{(\tau-(1-\alpha)d)/\alpha})^d + 1 \leq n$ completes the proof assuming $\log \sqrt[d]{n-1} - d > 0$, otherwise the lower bound of α is 0. □

As we know, the minimum α turns out to produce the most informative summaries. Thus given $\log \sqrt[d]{n-1} - d > 0$ by Equation (6) we have the estimation for α as

- when $\tau > d$, $\alpha = (\tau - d)/(\log \sqrt[d]{n-1} - d)$,
- when $\tau \leq d$, $\alpha = 0$,

to obtain the minimum entropy. The summarization can therefore adapt dynamically.

3.3.2 Threshold Selection

The goal of the DNS problem is to find the most interesting developments of dynamics over time as summaries. This naturally requires OSNet to only focus on the small set of the interesting nodes and edges in a spreading tree T. Our goal is to find a proper threshold that can make the summarization converge fast and produce a small sized summary over time. However, the changes and differences of dynamics challenge the setting of such a threshold. Therefore, a selection mechanism adapting to the trends of dynamics (i.e., rise and fall) is necessary.

The idea of the proposed solution is to maintain a variable τ' for each spreading tree T, which is the maximum value (MAX) of $\xi_{t'}(v_i)$, $v_i \in T$ by time t'. During the summarization, we compare a new interestingness score $\xi_t(v_j)$ with τ': if $\xi_t(v_j) > \tau'$, then $\tau' = \xi_t(v_j)$, and v_j is inserted into the corresponding $\mathbf{S}(T)$. If we have a value of τ' that is large enough, OSNet converges to a relatively steady state until there is a more interesting node, e.g., far away from the seed and with many infectees, to exhibit another rise of the diffusion. Thus, in a summary $\mathbf{S}(T)$ based on MAX, the interesting nodes (by the first condition in Definition 3) in deeper levels always show diffusion rises from those in lower levels. From an interesting node to a node recovered for the next interesting node, the flow is always a diffusion decay.

Other methods than MAX would be possible, e.g., average value (AVG) of $\xi_{t'}(v_i)$ as $\sum_{v_i \in V} \xi_{t'}(v_i)/|V|$. We compare these alternatives experimentally in Section 4.

3.4 Tree-Based Approach

An efficient way in a tree-based data model to solve the *Recovery Problem* is to construct $\mathcal{S}(T)$ as a search tree. The basic idea is that all the siblings in each level of $\mathcal{S}(T)$, namely the nodes getting infected from the same infector, are ordered. We thus can perform a binary search. The canonical ordering is based on time-stamps labeled on tree branches (edges) and node labels. If a node v_j gets infected from v_i at time t_i, v_j is inserted as: the time-stamps as edge labels of all the siblings on the left are not later than t_i, and the node labels on the left are not lexicographic larger than v_j. Lemma 2 presents the worst case search cost of the search tree.

Lemma 2. *Let a tree T have n nodes and the fanout of T is d. We have the worst case search cost when $d(T)$ is minimum as*

$$O(\log_d^{(n(d-1)+1)}(\log_d^{(n(d-1)+1)} - 1)\log_2 d).$$

Algorithm 1 captures two essential aspects of OSNet: 1) Constructing spreading trees (lines 7 to 10); 2) summarizing the most interesting dynamics into $\mathbf{S}(T)$ (lines 13 to 16). We proceed to explain the details. We allow users to terminate a summarization process through the variable breakFlag in line 6. Depending on the applications, one can also use a bound on the size of $\mathbf{S}(T)$ to abort the algorithm. Note that we have no limitation on n. Once a new interaction $x(\delta, v_i, v_j, t)$ arrives (line 7), we call mapT in line 8 to retrieve the T of story δ. Next, branchOut in line 10 inserts an infectee v_j from v_i with edge label t into T. We implement each $\mathbf{S}(T)$ as a search tree. From line 11, we summarize the updated node according to Equation (1). If the node's interestingness exceeds the threshold, it shows a diffusion rise, and the node is inserted into $\mathbf{S}(T)$. Parameters are adjusted in line 12 as discussed in Section 3.3.

Algorithm 1. Algorithmic Description of the OSNet

```
Input :  Network $G$, seed set $\mathcal{I}(G)$.
Output: A set of summarized spreading trees, $\mathbf{S}(G)$.
 1: begin
 2:   Threshold $\tau \leftarrow 0$; $\alpha \leftarrow 0$; $n \leftarrow |V|$
 3:   Boolean breakFlag $\leftarrow false$;
 4:   List path $\leftarrow null$;
 5:   Spreading tree set Set($T$) rooted by seeds in $\mathcal{I}(G)$;
 6:   if breakFlag $== false$ then
 7:     if $x(\delta, v_i, v_j, t) \leftarrow \mathcal{D}(G)[t_{ij}]$ exists then
 8:       $T \leftarrow$ mapT(Set($T$), $\delta$);
          /* add $x$ onto $T$. */
 9:       $v_i \leftarrow$ Search($T, v_i$);
10:       branchOut($v_i, v_j, t$);
11:       if $\xi(v_i) > \tau$ then
12:         $\tau \leftarrow \xi(v_i)$, set $\alpha$;
            /* retrieve path from $T$. */
13:       while $v_i.getInfector(T) \notin \mathcal{I}(G)$ do
14:         path.Push($v_i.getInfector(T)$);
15:       $\mathbf{S}(T) \leftarrow$ getST(Set($T$), $T$);
16:       insertPath($\mathbf{S}(T)$, path);
17:   return Set($T$);
```

Before inserting a node into $\mathbf{S}(T)$, we retrieve the path from the root in line 13 by iteratively pushing an infector (function Push) into list path. We then insert the missed nodes and edges into $\mathbf{S}(T)$ in line 16. These nodes show diffusion decays from the last interesting node, but rises to the next. Summaries are returned as necessary in line 17.

3.5 Path Hierarchical Locating Bloom Filter

The recovery procedure performs a search on $\mathcal{S}(T)$ and recovers the missed nodes/edges on the path $l(v_i)$ from the root v_0 in T. The time complexity to retrieve l from T is linear in the length, $O(|l(v_i)|)$. Lemma 2 shows that the search

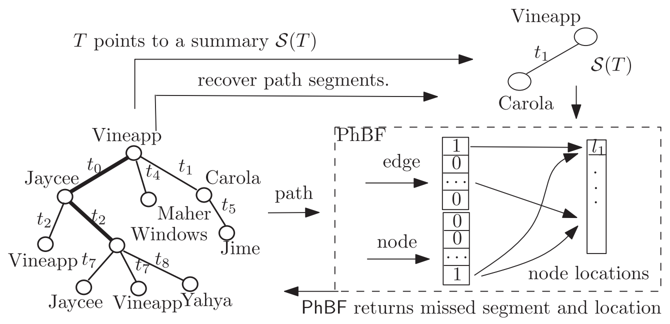


Fig. 4. Indexing for fast recovery.

complexity on $\mathcal{S}(T)$ depends on the fanout and the depth that is bounded by the scale of the tree. Although the search performs well on trees in general, large-scale and continuous dynamics are challenging for the recovery procedure, especially when the fanouts, depths, and lengths of the paths are large.

We thus propose a fast recovery method called Path Hierarchical Locating Bloom Filter (PhBF), which is an extended version of a Bloom filter. Similar to a Bloom filter, the time complexity of PhBF to search a node or an edge is constant, such that PhBF is able to efficiently determine which segment of a path needs to be recovered in $\mathcal{S}(T)$ and where it should be inserted [26]. In PhBF, two Bloom filters encode an edge-node pair. Each bit that equals one points to a location in the hierarchical tree $\mathcal{S}(T)$, which is the physical address of the infector in a pair. Fig. 4 gives an example of the proposed approach. Compared with a Bloom filter, PhBF has the following benefits: 1) it is capable to void encoding conflicts between nodes and edges; 2) it has well-proved false positive bounds for checking ordered pairs in a path; and 3) it has an indexing structure to locate physical addresses of the indexed nodes and edges.

We map each T to its summary $\mathcal{S}(T)$. When an interaction x is inserted into T, we retrieve its path to the root in the corresponding T. For instance in Fig. 4, for the interaction where a node Jaycee branches to Windows with edge t_2, a path (Vineapp, t_0, Jaycee, t_2, and Windows) is retrieved (the bold path in T). In our method, we check the path from the root to the leaf node to determine which segments have to be recovered. Each check concerns a node-edge pair of the path. We thus start the check with the pair consisting of the root Vineapp and the edge t_0. Because PhBF hashes the nodes and edges in $\mathcal{S}(T)$, the edge t_o is not indexed in PhBF. The check of the pair thus fails. In the figure, we can see that the Vineapp is in $\mathcal{S}(T)$, but that edge t_0 is not. PhBF then returns segment t_0 that is to be inserted into the root node location.

The design of PhBF is detailed below. First, each edge-node pair from top to down in $\mathcal{S}(T)$ is hashed into the edge Bloom filter and node Bloom filter. In total, $|\mathcal{D}(G)|$ (i.e., $\sum|\mathcal{S}(T)|$) pairs are hashed into the PhBF without consideration of the leaves. A PhBF begins with arrays of all 0s for the node and edge filters. Each node or edge of a pair in a Bloom filter is hashed k times by different hash functions, and each hash yields a bit location that is set to 1. If one bit is set to 1 from 0, this bit is inserted into a queue that contains the location of the infector.

For example, in Fig. 4, when we hash the edge labeled t_0 or the node Jaycee, the location of the root node is pushed into the queue. If this bit is already set to one, the location is pushed to the back of the queue.

Next, when we check a path l in T, we start from the edge-node pair (such a pair is also termed a value) next to the root in l. We maintain a pointer $\mathcal{P}$ to the root in $\mathcal{S}(T)$. If both elements of a pair are hashed to bit locations with 1s by the k hash functions, we update $\mathcal{P}$ to be the common location of the pair in the queue; otherwise, $\mathcal{P}$ and the checking pair are returned. With the returned pointer, we insert the path l from the checking pair into the location indicated by $\mathcal{P}$.

We proceed to analyze the properties of PhBF. As we know, the complexity of a Bloom filter is constant to check whether a value is hashed. Hence, the checking for each pair is constant. Since there can be multiple locations for a bit with one, we compute the intersection of the queues of the node and edge filters, which contain exactly one common location for all k hash functions because such a pair exists in the process if and only if such an interaction x is modeled in T. The complexity for this is $O(m)$, where m is the size of the queue.

A Bloom filter never produces a false negative, but may produce false positives. PhBF has an upperbound on the false positive probability. We hash a path into PhBF by using k hash functions. The number of pairs is $\hat{n}$, and the length of a Bloom filter is m. After all the pairs are hashed into PhBF, the probability that a specific bit is still 0 is $p = (1 - \frac{1}{m})^{k\hat{n}}$. In PhBF, the probability that the bits are 1s is $(1-p)^k$. We can now bound the false positive probability. Let a be the number of paths, and let b be the number of checked pairs in the path. To store the weight we can intuitively use 16 bits (float value), or we can encode the weight by $log(a)$. For the number of hash functions and the length of PhBF are usually set empirically [26].

Lemma 3. *The probability of a false positive of* PhBF *that considers a non-existing path as existing in a summary $\mathcal{S}(T)$ is* $\frac{aC(bk,k)}{C(\hat{n}k,k)}(1-p)^k$.

By construction, PhBF has much fewer false positives than the Bloom filter because the false positive probability is $\frac{aC(bk,k)}{C(\hat{n}k,k)}(1-p)^k$, while in the Bloom filter, the probability is $(1-p)^k$, and $\frac{aC(bk,k)}{C(\hat{n}k,k)}$ is obviously below 1.

Theorem 2. *Let q be the probability that a value is legal. The probability of a false positive during the recovery of a path in* PhBF *is* $\frac{ab^b}{\hat{n}^b}q^b$.

Proof. Each path is a set of b pairs. We hash these b values into PhBF. If all the values are legal (a value is legal if this value is in PhBF) and have equal weight, this path is a positive path. The probability that all b values are legal is q^b. For the b values, there are $\hat{n}^b$ possible combinations. For each path, the probability of all the values having the same location can be represented as b^b, and there are a paths. The probability that a path is positive if all the pairs are legal is $\frac{ab^b}{\hat{n}^b}$. □

Lemma 4. *The false positive probability of an edge-node pair (i.e., a value) in* PhBF *is upper-bounded by* $a(\frac{b}{\hat{n}-1})^k$.

Proof. For all the $C(\hat{n}k,k)$ possible situations where all the corresponding bit locations are set to one, we have

$$\begin{aligned} C(\hat{n}k, k) &= \frac{(\hat{n}k)!}{k!(\hat{n}k-k)!} = \frac{\overbrace{(\hat{n}k)\cdots(\hat{n}k-k+1)}^{\text{k components}}}{k!} \\ &\geq \frac{(\hat{n}k-k+1)^k}{k!} \\ &\geq \frac{(\hat{n}k-k)^k}{k!} = \frac{k^k(\hat{n}-1)^k}{k!}. \end{aligned} \tag{7}$$

Similarly, we have

$$C(\hat{n}k, k) \leq \frac{(\hat{n}k)^k}{k!}.$$

Hence, we obtain

$$\begin{aligned} q &= \frac{aC(bk,k)}{C(\hat{n}k,k)}(1-p)^k \leq \frac{ak!(bk)^k}{k^k(\hat{n}-1)^k k!}(1-p)^k \\ &= a(\frac{b}{\hat{n}-1})^k(1-p)^k \leq a(\frac{b}{\hat{n}-1})^k. \end{aligned} \tag{8}$$

□

Theorem 3. *The false positive probability of a path in* PhBF *is upper bounded by* $\frac{1}{a^{b-1}}(\frac{b}{\hat{n}-1})^{bk}$.

Proof. The probability of a path in PhBF is $\frac{q^b}{a^{b-1}}$, and thus we have $\frac{ab^b}{(ab)^b}q^b \leq \frac{q^b}{a^{b-1}} \leq \frac{1}{a^{b-1}}(\frac{b}{\hat{n}-1})^{bk}$. □

We show the psudocode of PhBF in Algorithm 2. Lines 4 to 15 check the parts of a path that are missing in $\mathcal{S}(T)$, within which we get the locations by using the edge and node filters (lines 7 to 10). Edge and node hash functions are denoted by fe_m and fn_m ($m \in [0-k)$), respectively. The common locations for each index are kept in Δ_m, and we check whether there is such an edge-node pair in line 11, where the same location for both filters should appear for k functions. If such a location exists in $\mathcal{S}(T)$, we break the check of the path and insert the checked parts into $\mathcal{S}(T)$ from the last infector v_i (line 13); otherwise, we continue the check from line 15. The insertion is done in lines 16 to 17. Note that in the implementation, we check the path in a bottom-to-top fashion because we can then directly recover the missed parts and skip the repeated checking of the top parts of T for paths. This is efficient, especially when the missed parts are relatively short.

Complexity Analysis. The space used by Algorithm 1 includes the space used for spreading trees and the space used for summaries. The space for spreading trees equals the space of the $|\mathcal{D}(G)|$ received interactions. The space can be compressed by using statistics to reduce some parts of a tree. For example, we can associate a label with a node as the number of infectees instead of inserting all infectees. The space for summaries is in the worst case the same as that of the original data. This occurs when all the leaf nodes of a T have the maximum interestingness score (Equation (1)). In practice, the space is much lower.

We quantify the I/O cost in experiments. If $|\mathcal{D}(G)| = n$, the time complexity to build spreading trees is $O(n\log n)$ for performing a binary search on n interactions. The time complexity to retrieve a path from T is $O(d(T))$, which is at most $O(\log n)$. For PhBF, we need to hash $n = ab$ pairs into the Bloom filter by k hash functions, so the time complexity is $O(nk)$. To recover a path, we need to hash b pairs into PhBF by k hash functions, so the time complexity is $O(bk)$. The comparison costs at most kb. In total, the time complexity is $O(bk)$, where b is at most equal to $O(\log n)$, which is the maximum depth. The time complexity is thus $O(n \log n)$ in total.

4 Experimental Study

This section presents an extensive series of experiments to evaluate OSNet. We compare OSNet with existing algorithms as well as implementations of OSNet. We also study the efficiency and design properties of OSNet. Case study and applications, including a user study, are further discussed on two large scale real-life dynamic networks.

4.1 Experimental Methodology and Settings

The experiments consider four questions: 1) *Sense-Making Evaluation*: Compared with the state-of-the-art, do the summaries generated by OSNet make sense and achieve the goal of capturing interesting dynamics? Do summaries of OSNet meet user expectations? 2) *Parameter Study*: Can we use fixed parameters? What are the effects of the parameters? Does OSNet converge fast, using MAX or AVG? 3)*Algorithm Efficiency*: Does PhBF perform better than the tree-based approach for the recovery problem? 4) *Real-life Data*: How does OSNet work on real-life data? Does OSNet adapt to different dynamics? What can we derive from the summaries?

Algorithm 2. Fast Recovery PhBF

```
Input : Summarized spreading tree $\mathcal{S}(T)$, a path Path.
Output: Updated summarized spreading tree $\mathcal{S}(T)$.
1:  begin
2:    Buffer edge ← $null$, node ← $null$;
3:    Integer flag ← 0;
4:    for $i$ ← 0 to |Path| do
5:        node ← Path[$i$];
6:        edge ← $getEdge(node, v_i)$;
7:        for $m$ ← 0 to $k$ do
8:            addrA ← hashEdge($fe_m, edge$);
9:            addrB ← hashNode($fn_m, node$);
10:           $\Delta_m$ ← addrA ∩ addrB;
11:       if $\Delta_0 \cap \ldots \cap \Delta_k \neq \emptyset$ then
12:           flag = $i + 1$;
13:           $v_j$ ← Path[flag];
14:           break;
15:       $v_i$ ← node;
      /* recovery path */
16:   for $i$ ← 0 to flag − 1 do
17:       branchOut($v_i$, Path[$i$], $get\,Edge(node, v_i)$);
18:   return $\mathcal{S}(T)$;
```

Experiments on synthetic data are used to test whether our methods produce expected results in a controlled environment. As OSNet makes no assumption on the influence of underlying networks, the intentional test of graphs with various distributions are not considered for this study. Instead, we provide sets of interactions generated by different distributions, which are on a set of nodes G_0 containing 10,000 labeled nodes. With a random seed set $\mathcal{I}(G_0)$, we then start the propagation for each seed. The number of infectees of a node v obeys the following models to simulate different dynamics as interactions: I) Gaussian distribution (G); II) Poisson distribution (P); III) Zipf distribution (Z), which is an

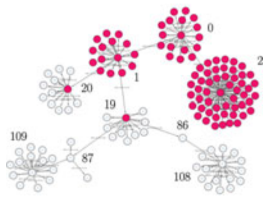


Fig. 5. $\mathcal{D}(G)$ at t_1.

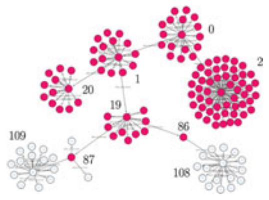


Fig. 6. $\mathcal{D}(G)$ at t_2.

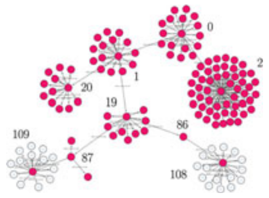


Fig. 7. $\mathcal{D}(G)$ at t_3.

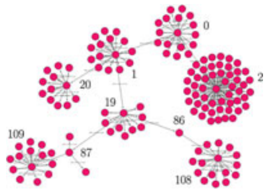


Fig. 8. $\mathcal{D}(G)$ at t_4.

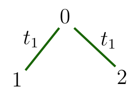


Fig. 9. The summary at t_1.

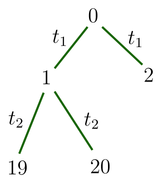


Fig. 10. The summary at t_2.

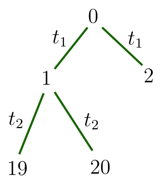


Fig. 11. The summary at t_3.

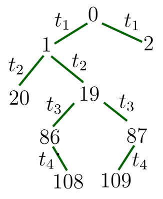


Fig. 12. The summary at t_4.

approximate power law probability distribution. We define the *modeled number of nodes* (η) to be the number of labeled nodes we choose for a dataset, and we require that their numbers of infectees obey one of the three distributions.

All experiments were conducted on a 3.2 GHz Intel Core i5 with 16GB 1,600 MHZ DDR3 main memory and running OSX. All algorithms were implemented in JDK 1.6.

4.2 Sense-Making Evaluation on Synthetic Data

The sense-making experiments consist of two parts. In the first part, we compare OSNet with several existing algorithms using synthetic data. In the second part, we use a real-life dataset and conduct a user study. We study whether summaries meet user expectations. We also compare with summaries generated by other algorithms.

To enable existing methods to support diffusion processes, we generate a graph sequence for each dataset, in which each graph aggregates all edges and nodes in a time interval Δt. Due to the space limitation, we only report results for several time intervals. Similar findings apply to other intervals. We compare our techniques against the following state-of-the-art algorithms:

- *DisSim-Alg*: This is a graph compression algorithm that abstracts a large graph into a smaller graph that contains approximately the same information. It is developed based on the notion of dissimilarity between the decompression graph and the original graph. We use an existing implementation [3] and set the weight of an edge to 1 if the adjacent nodes diffuse infection by time t: otherwise, edge weights are set to 0.
- *MDL-Alg*: MDL is a successful and popular technique for graph compression. We compare against a recent study by Navlakha et al. [2] where a graph is compressed and represented as a graph summary and a set of corrections. We use the original GREEDY algorithm that offers the best compression and lowest cost [2]. To enable cliques to be merged into a single supernode, we add self-edges to each node before applying the algorithm.

(OSNet). Figs. 5, 6, 7, and 8 show a diffusion process $\mathcal{D}(G)$ from t_1 to t_4 using data generated by applying Zipf distribution. The infector as the central node of each group is labeled with a canonical identifier for ease of explanation. The node with identifier 0 is the seed of the propagation. In the four figures, the red and darker nodes are the nodes that are already infected; the grey and lighter nodes are other nodes in the synthetic networks. To facilitate visualization, we remove the background nodes and edges in the underlying networks that are not involved in the diffusion process. Figs. 9, 10, 11, and 12 present the summaries by OSNet from t_1 to t_4. The results show the approach is incremental and a summary in each figure grows based on the previous results. The intuitively interesting nodes are captured, and the summaries are traceable and connected paths, such that

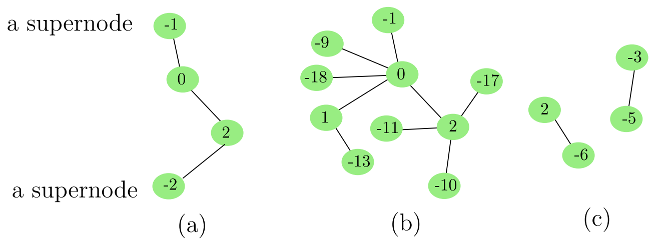


Fig. 13. Summaries by *DisSim-Alg* at t_1.

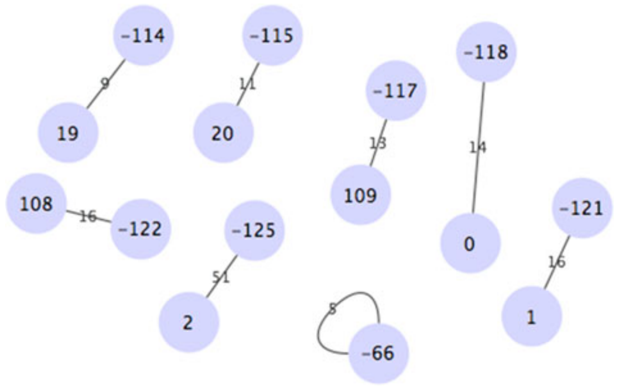


Fig. 14. Summaries by *MDL-Alg*.

we can spot the dynamics from the start to the nodes i) that can infect many others; ii) that are far from the seed. We observe that the summaries in Figs. 10 and 11 are the same. Back to the original diffusion process $\mathcal{D}(G)$ from t_2 to t_3, the diffusion reached nodes 86 and 87 at t_3. However the number of infectees is quite few. Compared with the other nodes in $\mathcal{S}(T)$, 86 and 87 are thus not that interesting to be summarized. This actually shows from t_2 to t_3 the diffusion process falls down in diffusion from nodes 19 and 20, and OSNet adapts to the changes in diffusion. In contrast, at time t_4, both 108 and 109 have many infectees and they are far away from the seed 0. They again expedite the diffusion process and are captured as interesting nodes by OSNet. If we only summarize the two without including nodes 86 and 87, we lose the connections that allow us to interpret how information propagates. Thus, 86 and 87 are recovered and included. The findings show that OSNet is capable of finding a small set of connected interesting nodes that meaningfully capture the diffusion process.

(*DisSim-Alg*). We aggregate interactions by varying sliding window Δt to generate graph sequences and try various values for the internal compression ratio parameter. We report three representatives at t_1 in Fig. 13 where a node with negative number indicate supernode (i.e., a cluster of graph nodes).

The findings show that the summaries vary a lot w.r.t. compression ratio. Comparing (a) and (c) where (c) is with a higher compression ratio, the graph size of (c) is much smaller but it is with less information of the propagation because *DisSim-Alg* aims to minimize the dissimilarity according to edge weights. To maintain a smaller dissimilarity, some edges or superedges are removed (e.g., (c)). Fig. 13b shows a summary produced by using a smaller Δt that is larger than that of (a). This occurs because when the compression ratio is achieved, although new edges and nodes arrive, the algorithm only considers the dissimilarity and does not attempt further

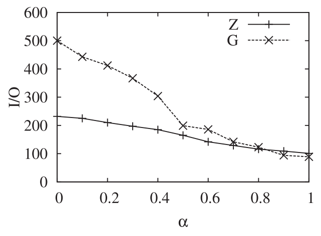


Fig. 15. I/O cost by varying α (a weight parameter).

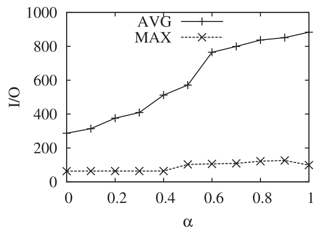


Fig. 16. I/O cost by various strategies on τ selection.

compression. As a result, the algorithm does not adapt to dynamics and capture traceable flows well.

(*MDL-Alg*). The summary by *MDL-Alg* on the same diffusion process is shown in Fig. 14. *MDL-Alg* is parametricless, which computes the best cliques to merge in order to maintain a low cost. Particularly, in Fig. 14, the rightmost clique means node '1' connects with supernode '-121' with edge weight 16. However, separate cliques cannot support the traceability of a diffusion process.

4.3 Parameter Study

We evaluate the effect of three parameters that are used in OSNet. The first is α in Equation (1) used to balance the weights. The second is the threshold τ. The last is the maximal number of nodes n that can be in $\mathcal{S}(T)$ we used to estimate α in Equation 3.3.1.

(*Weight α on proScope*). We increase α from 0 to 1 in steps of 0.1. For synthetic data, we generate the maximum spreading trees with depth 100, and η is 1,000. Note that in the experiments, we set $d = 2$ for finding summarized trees as the minimum depth of interesting summaries. For Gaussian (G) datasets, we set the mean to 100 and the standard deviation to 20. The expect value for Poisson distribution (P) is 50. The maximum $deg^+(v)$ of the *Zipf* distribution (Z) is 200. Consequently, we have three datasets with 89,037 (G), 45,306 (P), and 36,892 (Z) interactions, respectively. We set τ to 100. Fig. 15 shows the I/O efficiency with respect to α. We count the I/O cost as the size of summaries, namely the number of interactions in $\mathsf{S}(T)$. The I/O cost for the dataset P is 0 that means that no node in the propagation process gains a score that reaches 100. The findings show that the same fixed threshold does not work well across different datasets. For both G and Z in Fig. 15, the I/O cost decreases as α increases. As we know, α controls the weight of *proScope*. Thus, when α is small, the *proRadius* becomes more important in Equation (1). As a result, nodes that are far away from a seed are more likely to be captured, which yields a higher I/O cost.

(*Threshold τ*). We compare our proposal that uses the current maximum score against using the average historical score AVG. We use the same datasets as above. Fig. 16 shows the

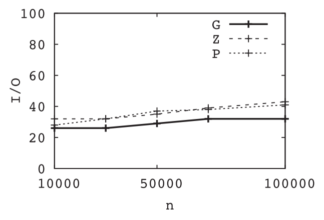


Fig. 17. I/O cost by varying n (the maximal number of nodes).

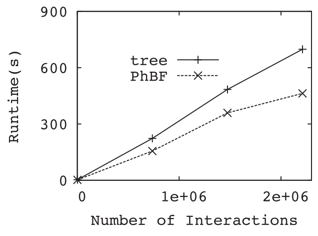


Fig. 18. Runtime (including data arrival time).

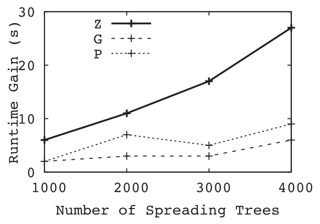


Fig. 19. Varying the number of processes.

findings on dataset G, which indicate that the I/O cost of MAX is much lower than that of AVG. And with a given α, the summarization with MAX converges faster to a relatively steady state than with AVG. MAX requires less updates on the summarized spreading trees than does AVG. Compared with the findings in Fig. 15, the I/O cost increases as α increases when using AVG, because a larger value of α yields a larger score. This allows more nodes of a $\mathcal{D}(G)$ to be summarized, which increases the I/O cost. However for MAX, the cost remains almost the same when $\alpha < 0.6$ and it increases only slightly afterwards. We obtain similar results on the other two datasets.

By Equation 3.3.1, α never decreases because τ is based on the MAX strategy. This is beneficial for summarization for two reasons: i) With MAX, a larger α allows a bit more nodes to be summarized if diffusion rises; ii) a larger α decreases the influence of *proRadius* such that the summarization converges faster. This keeps OSNet from capturing too many nodes even when many are far away from seeds.

(*Maximum Possible Summary Size*). We evaluate the effect of maximum possible summary size n in Equation 3.3.1 by varying the parameter from 10,000 to 100,000. The findings in Fig. 17 for all the three datasets show that the I/O cost increases as the parameter increases. A larger maximum possible summary size n yields a smaller α. Fig. 17 thus shows the same I/O cost trend as does Fig. 15. However, the variation in Fig. 17 is slight. For simplicity, we suggest to set the parameter as maximum possible summary size to be the number of nodes, which is also the maximum number of nodes that can be summarized in an $\mathbf{S}(T)$.

4.4 Algorithm Efficiency

We evaluate the efficiency of the PhBF-based the tree-based OSNet in two situations: 1) the efficiency of summarizing a large number of diffusion processes; 2) the efficiency as varying the length of recovery paths when the number of diffusion processes is fixed. In both experiments, d in Equation 3.3.1 is set to two as the lower bound of the expected depth of the interesting summaries in search. In a similar way, we generate three datasets with $\eta = 100$ following the three distributions P (expect value is 20), G (mean is 100, deviation is 20), and Z ($deg^{+}(v) = 100$). For each distribution, we vary the number of diffusion processes (i.e., spreading trees) from 1,000 to 4,000 with step 1,000 and generate five sets for each number. The average runtime gains of PhBF outperforming tree-based method are shown in Fig. 19. Fixing the number of trees as 1,000 for each distribution dataset, we vary the length of recovery paths and randomly pick paths from each tree and recovery locations, the results of runtime gain are reported in Fig. 20. The findings reveal that the PhBF-based approach outperforms the tree-based approach especially for power-law graphs.

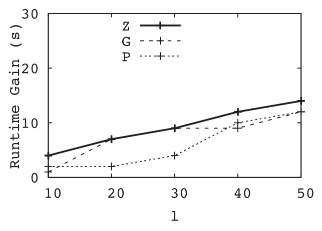


Fig. 20. Vary the length of recovery paths l.

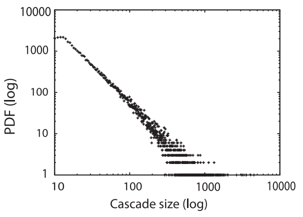


Fig. 21. Cascade size (Weibo).

4.5 Case Study and Applications

4.5.1 Social Networks

We use data from Sina Weibo, a Chinese Twitter-like microblogging service platform (http://www.weibo.com) that has two important features that are not yet offered by Twitter: 1) A user can comment on any other user's tweets, which yields more user interactions; 2) The retweeting/forwarding chain is visible to the public, which is important for studying diffusion processes. Our dataset covers more than 1.8 million users, and we reconstruct the diffusion processes from their replies. There are 41,561 cascades (diffusion processes) with 2,211,221 interactions. We show that the probability density distribution (PDF) of the cascade size (Log-Log) as a property of the original data in Fig. 21. OSNet outputs 8,647 summaries in which a seed has at least one infectee. Among the results, the summary with the most edges has 62 edges. The PDF of the summary size (Log-Log) is shown in Fig. 22, which shows that most of the summaries are small. The lower bound of depth in Equation 3.3.1 is set to 4 since the majority of the diffusion processes (> 78 percent) are with depths smaller than four. Interactions are read from external files by random access.

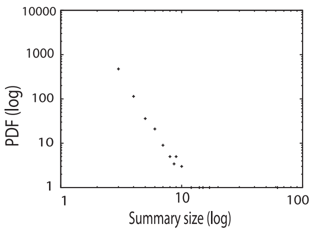


Fig. 22. PDF of summary size.

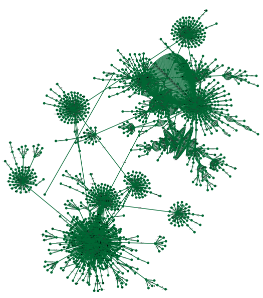


Fig. 23. A $\mathcal{D}(G)$ sample.

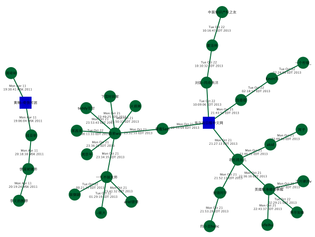


Fig. 24. Summaries of $\mathcal{D}(G)$.

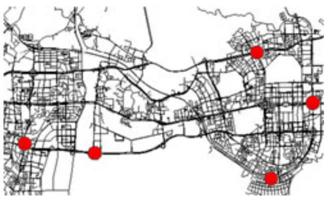


Fig. 25. Summarization of traffic. A red dot indicates major traffic thoroughfare, and dark lines indicate roads.

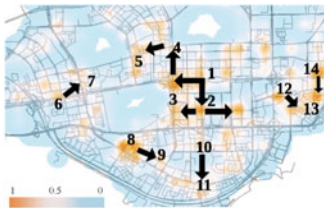


Fig. 26. Hot spots detection in traffic networks. In the heat map, color density indicates the normalized traffic density.

Fig. 18 shows the runtimes of the tree-based and the PhBF-based OSNet. Fig. 23 shows a sample of diffusion processes represented by cascades. Fig. 24 gives the corresponding OSNet summaries where a rectangle node is a root. Although the cascades in Fig. 23 may merge on some nodes, the summaries are separated from each other w.r.t. stories. In the study we find that some of the summarized nodes are actively engaged in other diffusion processes, which meets the intuition of the interestingness.

(User Study for Sense-Making Evaluation). A user study is conducted as a field study on Sina Weibo data. The study is conducted using a questionnaire in which we show an example of dynamics with 100 interactions and give a list of questions, each with several options. We obtain 27 participants with either a computer science or a sociology background and cover participants from five different countries (China, Denmark, India, Korea, and Singapore). We visualize the example as network cascades [10], [25]. The questions can be summarized into four categories: 1) (Seed) Do the participants think the seed is interesting and necessary to know for understanding the dynamics? 2) (Interestingness) With only few nodes to represent the dynamics, do the participants agree with the intuitions that the nodes with more infectees (high degree) and far away from the seed are more interesting in the dynamic process? 3) (Traceability) Is traceability necessary for understanding a dynamical process? Do the participants think a set of connected nodes is suitable for understanding dynamics? 4) (Algorithms) Among the summaries generated by *DisSim-Alg*, *MDL-Alg*, and OSNet, which one do the participants think best describes the dynamics? The statistical results for each category are: 1) 85.2 percent. 2) 96.3 percent. 3) 92.6 percent. 4) All participants consider OSNet as the best for understanding the dynamics. The results show that when the participants try to understand dynamics, i) the nodes that have many infectees and are far from the seeds are more interesting (from 2)); and ii) traceable summaries are more comprehensible and intuitive (from 1) and 3)).

4.5.2 Traffic Networks

Different sources of traffic information were merged to produce this work. A shapefile (a popular geospatial vector data format that describes the geometry of a network) of the city region of Shenzhen[2] was provided along with sequences of road segments tagged with taxi identifiers and timestamps representing taxi movements. The shapefile contained 264,425 road segments. We obtained one year of GPS data from approximately 15,000 taxis in Shenzhen to capture the traffic dynamics [8], [17].

For traffic dynamics, first, we define a road network based on road segments (edges) and road junctions (nodes). Second, the traffic dynamics is defined by the speed on each edge [17]. Third, we apply OSNet, and then determine the major traffic thoroughfares.

Fig. 25 is based on the result of summarizing one year of data capturing traffic dynamics and shows a sample of major traffic thoroughfares in the results. The major traffic thoroughfares are all located at the intersections of highways and ordinary city roads. Traffic diffusion thus rises when traffic moves towards the thoroughfares in the city. With the thoroughfares shown in dynamic traffic data by OSNet, we can provide routing services according to traffic conditions. In Fig. 26, we visualize one week of traffic dynamics by means of a heat map, in which the regions with continuous heavy traffics have values close to one, termed hot spots. The results of applying OSNet are shown in the figure, where the numbers are ID indicators of such regions and the arrows indicate sequences of traffic flows

2. http://en.wikipedia.org/wiki/Shenzhen

between the indicators. We can observe the start, rise, and decay of traffics from the evolution of the hot spots. For instance, the summary from 8 to 9 shows that the region 8 is the start of heavy traffic and that the traffic moves towards region 9 in a decaying manner because the propagation stops at 9 and the size of region 9 is smaller than that of 8. We compare this result with a baseline: average usage of the road, that is, the average speed of the traffic. The baseline describes the traffic density as the same purpose of our approach. Out of the 13 hot spots, the baseline gives the same set of hot spots as our approach. But the difference between the baseline and our approach is that our approach is also able to describe the sequence of the traffic flow. Different from statistical methods which provide complex models and detailed figures for traffic optimization and planning, such online summarization serves to offer important visual cue for easy overview and intuitive understanding of the dynamic traffic flow, which is top priority for most real-life traffic monitoring and management systems to achieve quick response and comprehensive evaluation for both global and local situations, especially for operation personnel at the front line.

5 Related Work

Graph Mining. Statistical methods [27], [28] are widely used to characterize properties of large graphs. However, most methods do not produce topological summaries thus hard to interpret. *Graph pattern mining* [29] can be used for summarizing graphs, but usually yields overwhelmingly large numbers of patterns. Although constraint-based graph mining approaches [19], [20] are introduced to reduce the number, they only work for specific constraints. Further, summaries of dynamics are not inherently frequent. *Graph partitioning algorithms*, such as Compact Matrix Decomposition (CMD) [30], are useful in detecting dense subgraphs but node attributes are largely ignored. Next, graph OLAP has been introduced to summarize large graphs [31]. However, most studies are designed for static networks and are limited to user-specified aggregation operations. *Graph clustering* methods, such as meta attributes and statistical models [32], for measuring distance of link structures, have attracted much attention for summarization.

Graph Compression. Graph compression and simplification mainly focus on generating compact graph representations to simplify storage and manipulation. Much of the work has focused on lossless web graph compression [1], [33], [34]. Web pages with similar adjacency lists are encoded using reference encoding. Most of these studies, however, only focus on reducing the number of bits needed to encode a link, and few compute topological summaries since the compressed representation is not really a graph. An exception is a study [34] that computes graph summaries by grouping web pages based on a combination of their URL patterns and k-means clustering. Lossy topological summarization is another graph compression study. Based on the MDL principle, Navlakha et al. [2] propose an error bounded representation that recreates the original graph within a bounded error. Toivonen et al. [3] merge nodes of a graph that share similar properties to achieve a summary. Fan et al. [35] propose query-preserving graph compression that retains equivalent query results on two particular query classes. Compared with these studies, our approach is developed to summarize diffusion processes. Diffusion processes as dynamic graphs do not belong to those special families of graphs (e.g., unlabeled and static trees or planar graphs that have repeated patterns and infrequent change nodes/edges) for which efficient storage compression has been proposed in graph compression literature [36]. As a result, direct adaption of these methods is not possible for online summarization of dynamic networks.

Compared with the preliminary work [37], the techniques are significantly extended. We extend OSNet to embed different methods to support path recovery problem, and we propose a more efficient method based on Bloom filters compared with a tree-based method. New experiments are conducted to explore the new techniques. Moreover, we use OSNet for city transportation analysis.

Graph Dynamics Analysis. As one of the attempts to consider time-evolving networks, Liu et al. [15] compress weighted time-evolving graphs, which is equivalent to compressing a sequence of static graphs. Ferlež et al. [38] propose TimeFall to monitor network evolution that clusters texts in scientific networks and uses MDL to connect clusters. This class of studies are inherently distinct from ours in four aspects: 1) we use general networks and do not have assumptions on text processing; 2) OSNet takes as argument an interaction stream rather than a time-stamped offline network; 3) a sequence of time-sliced graphs are not assumed; 4) we aim to summarize diffusion processes. There are also studies on temporal dynamics of social networks, including inferring cascades [39], finding common progression stages in event sequences [5], predicting cascades [4]. They focus on tasks different from ours.

Diffusion Modeling. Many diffusion models are proposed to model information diffusion and adoption, which can be distinguished as explanatory models (e.g., NETINF [39], NETRATE [40], INFOPATH [41]) based on complete diffusion data to retrace implicit path from generative probabilistic models and predicting models of cascade unfolding based on historical data [42]. Independent cascade and linear threshold models are two extensively studied graph-based influence diffusion models originally summarized by Kempe et al. [25]. The two models are based on the intuition that often decision is correlated with the number of friends. This work does not consider the influence of nodes and makes no assumption on underlying networks. The study takes complete and timely interactions in cascades as information diffusion for the purpose of summarization.

6 Conclusion and Future Work

Motivated by information diffusion studies, we proposed the problem of dynamic network summarization and provided an online, incremental summarization framework, OSNet, capable of simultaneously capturing the most intuitively interesting summaries.

There exist limitation and open questions for this study, which however points several promising directions for future work. For instance, 1) dynamic addition of seed nodes and unobservable diffusion processes [39], [43] are beyond the consideration of this study; 2) this work has no assumption on underlying networks that may have influence on the effectiveness of summarization; 3) this study assumes simple cascades as information diffusion, which could be extended to support other models considering user influences and explicit time granularity of interactions; 4) definitions of interestingness may be application-oriented, which may require a generalized version of OSNet.

Acknowledgments

The work was partially supported by Basic Research Program of Shenzhen (JCYJ20140610152828686), and Natural Science Foundation of China (61572488).

References

[1] Y. Lim, U. Kang, and C. Faloutsos, "SlashBurn: Graph compression and mining beyond caveman communities," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 12, pp. 3077–3089, Dec. 2014.

[2] S. Navlakha, R. Rastogi, and N. Shrivastava, "Graph summarization with bounded error," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2008, pp. 419–432.

[3] H. Toivonen, F. Zhou, A. Hartikainen, and A. Hinkka, "Compression of weighted graphs," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2011, pp. 965–973.

[4] J. Cheng, L. A. Adamic, P. A. Dow, J. M. Kleinberg, and J. Leskovec, "Can cascades be predicted?" in *Proc. 23rd Int. Conf. World Wide Web*, 2014, pp. 925–936.

[5] J. Yang, J. McAuley, J. Leskovec, P. LePendu, and N. Shah, "Finding progression stages in time-evolving event sequences," in *Proc. 23rd Int. Conf. World Wide Web*, 2014, pp. 783–794.

[6] C. Hage, C. S. Jensen, T. B. Pedersen, L. Speicys, and I. Timko, "Integrated data management for mobile services in the real world," in *Proc. 29th Int. Conf. Very Large Data Bases*, 2003, pp. 1019–1030.

[7] C. S. Jensen, A. Friis-Christensen, T. B. Pedersen, D. Pfoser, S. Saltenis, and N. Tryfona, "Location-based services: A database perspective," in *Proc. Scandinavian Res. Conf. Geographical Inf. Sci.*, 2001, pp. 59–68.

[8] S. Liu, Y. Yue, and R. Krishnan, "Adaptive collective routing using gaussian process dynamic congestion models," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2013, pp. 704–712.

[9] Y.-R. Lin, H. Sundaram, and A. Kelliher, "Summarization of social activity over time: People, actions and concepts in dynamic networks," in *Proc. Int. Conf. Inf. Knowl. Manage.*, 2008, pp. 1379–1380.

[10] J. Leskovec, M. McGlohon, C. Faloutsos, N. S. Glance, and M. Hurst, "Patterns of cascading behavior in large blog graphs," in *Proc. SIAM Int. Conf. Data Mining*, 2007, pp. 551–556.

[11] M. Gomez-Rodriguez, J. Leskovec, D. Balduzzi, and B. Schölkopf, "Uncovering the structure and temporal dynamics of information propagation," *Netw. Sci.*, vol. 2, pp. 26–65, 2014.

[12] J. Leskovec, L. A. Adamic, and B. A. Huberman, "The dynamics of viral marketing," *ACM Trans. Web*, vol. 1, no. 1, 2007, Art. No. 5.

[13] M. Richardson and P. Domingos, "Mining knowledge-sharing sites for viral marketing," in *Proc. 8th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2002, pp. 61–70.

[14] T. Sakaki, M. Okazaki, and Y. Matsuo, "Earthquake shakes Twitter users: Real-time event detection by social sensors," in *Proc. 19th Int. Conf. World Wide Web*, 2010, pp. 851–860.

[15] W. Liu, et al., "On compressing weighted time-evolving graphs," in *Proc. 21st ACM Int. Conf. Inf. Knowl. Manage.*, 2012, pp. 2319–2322.

[16] J. Sun, C. Faloutsos, S. Papadimitriou, and P. S. Yu, "GraphScope: Parameter-free mining of large time-evolving graphs," in *Proc. 13th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2007, pp. 687–696.

[17] S. Liu, Y. Liu, L. M. Ni, J. Fan, and M. Li, "Towards mobility-based clustering," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery*, 2010, pp. 919–928.

[18] J. Yang and S. Counts, "Predicting the speed, scale, and range of information diffusion in Twitter," in *Proc. 4th Int. AAAI Conf. Weblogs Social Media*, 2010, pp. 355–358.

[19] F. Zhu, Z. Zhang, and Q. Qu, "A direct mining approach to efficient constrained graph pattern discovery," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2013, pp. 821–832.

[20] F. Zhu, Q. Qu, D. Lo, X. Yan, J. Han, and P. S. Yu, "Mining top-k large structural patterns in a massive network," in *Proc. VLDB Endowment*, vol. 4, no. 11, pp. 807–818, 2011.

[21] Q. Qu, C. Chen, C. S. Jensen, and A. Skovsgaard, "Space-time aware behavioral topic modeling for microblog posts," *IEEE Data Eng. Bulletin*, vol. 38, no. 2, pp. 58–67, Jun. 2015.

[22] R. Xie, F. Zhu, H. Ma, W. Xie, and C. Lin, "CLEar: A real-time online observatory for bursty and viral events," in *Proc. VLDB Endowment*, 2014, pp. 1637–1640.

[23] M. Gladwell, *The Tipping Point: How Little Things can Make a Big Difference*. Boston, MA, USA: Little Brown, 2000.

[24] M. Cha, A. Mislove, and K. P. Gummadi, "A measurement-driven analysis of information propagation in the Flickr social network," in *Proc. 18th Int. Conf. World Wide Web*, 2009, pp. 721–730.

[25] D. Kempe, J. Kleinberg, and E. Tardos, "Maximizing the spread of influence through a social network," in *Proc. 9th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2003, pp. 137–146.

[26] S. Liu, L. Kang, L. Chen, and L. M. Ni, "Distributed incomplete pattern matching via a novel weighted bloom filter," in *Proc. IEEE 32nd Int. Conf. Distrib. Comput. Syst.*, 2012, pp. 122–131.

[27] D. Chakrabarti and C. Faloutsos, *Graph Mining: Laws, Tools, and Case Studies*. San Rafae, CA, USA: Morgan & Claypool Publishers, 2012.

[28] G. Cormode, M. Datar, P. Indyk, and S. Muthukrishnan, "Comparing data streams using hamming norms (how to zero in)," *IEEE Trans. Knowl. Data Eng.*, vol. 15, no. 3, pp. 529–540, May/Jun. 2003.

[29] A. Inokuchi, T. Washio, and H. Motoda, "An apriori-based algorithm for mining frequent substructures from graph data," in *Proc. 4th Eur. Conf. Principles Data Mining Knowl. Discovery*, 2000, pp. 13–23.

[30] J. Sun, Y. Xie, H. Zhang, and C. Faloutsos, "Less is more: Sparse graph mining with compact matrix decomposition," *Statistical Analysis and Data Mining*, vol. 1, no. 1, pp. 6–22, Feb. 2008.

[31] Q. Qu, F. Zhu, X. Yan, J. Han, P. S. Yu, and H. Li, "Efficient topological Olap on information networks," in *Proc. 16th Int. Conf. Database Syst. Adv. Appl.*, 2011, pp. 389–403.

[32] T. Xu, Z. Zhang, P. S. Yu, and B. Long, "Generative models for evolutionary clustering," *ACM Trans. Knowl. Discovery Data*, vol. 6, no. 2, pp. 7:1–7:27, 2012.

[33] P. Boldi and S. Vigna, "The webgraph framework I: Compression techniques," in *Proc. 13th Int. Conf. World Wide Web*, 2004, pp. 595–602.

[34] S. Raghavan and H. Garcia-molina, "Representing web graphs," in *Proc. 19th Int. Conf. Data Eng.*, 2003, pp. 405–416.

[35] W. Fan, J. Li, X. Wang, and Y. Wu, "Query preserving graph compression," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2012, pp. 157–168.

[36] S. Chen and J. H. Reif, "Efficient lossless compression of trees and graphs," in *Proc. 6th Data Compression Conf.*, 1996, Art. no. 428.

[37] Q. Qu, S. Liu, C. S. Jensen, F. Zhu, and C. Faloutsos, "Interestingness-driven diffusion process summarization in dynamic networks," in *Proc. Mach. Learn. Knowl. Discovery Databases*, 2014, pp. 597–613.

[38] J. Ferlež, C. Faloutsos, J. Leskovec, D. Mladenic, and M. Grobelnik, "Monitoring network evolution using MDL," in *Proc. IEEE 24th Int. Conf. Data Engineering*, 2008, pp. 1328–1330.

[39] M. Gomez-Rodriguez, J. Leskovec, and A. Krause, "Inferring networks of diffusion and influence," *ACM Trans. Knowl. Discovery Data*, vol. 5, no. 4, Art. no. 21, 2012.

[40] M. Gomez-Rodriguez, D. Balduzzi, and B. Schölkopf, "Uncovering the temporal dynamics of diffusion networks," in *Proc. 28th Int. Conf. Mach. Learn.*, 2011, pp. 561–568.

[41] M. Gomez-Rodriguez, J. Leskovec, and B. Schölkopf, "Structure and dynamics of information pathways in online media," in *Proc. 6th ACM Int. Conf. Web Search Data mining*, 2013, pp. 23–32.

[42] A. Guille, H. Hacid, C. Favre, and D. A. Zighed, "Information diffusion in online social networks: A survey," *ACM SIGMOD Rec.*, vol. 42, no. 2, pp. 17–28, 2013.

[43] M. Farajtabar, M. Gomez-Rodriguez, N. Du, M. Zamani, H. Zha, and L. Song, "Back to the past: Source identification in diffusion networks from partially observed cascades," in *Proc. 18th Int. Conf. Artificial Intell. Statistics*, 2015, pp. 232–240.

Qiang Qu received the MSc degree in computer science from Peking University and the PhD degree from Aarhus University, supported by the GEO-Crowd project under Marie Skłodowska-Curie Actions. He is an associate professor with the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences. His current research interests include large-scale data management and mining.

Siyuan Liu received the first PhD degree from the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, and the second PhD degree from the University of Chinese Academy of Sciences. He is an assistant professor in the Smeal College of Business, Pennsylvania State University. His research interests include spatial and temporal data mining, social networks analytics, and mobile marketing.

Feida Zhu received the BSc degree in computer science from Fudan University, China, and the PhD degree in computer science from the University of Illinois, Urbana-Champaign, in 2001 and 2009, respectively. He is an assistant professor in the School of Information Systems, Singapore Management University. His current research interests include large scale data mining, graph/network mining, and social network analysis.

Christian S. Jensen is an Obel professor of computer science with Aalborg University, Denmark. He was recently with Aarhus University for three years and with Google Inc. for one year. His research concerns data management and data intensive systems, and its focus is on temporal and spatiotemporal data management. He received several national and international awards for his research. He is an editor-in-chief of the *ACM Transactions on Database Systems* and was an editor-in-chief of the *VLDB Endowment Journal*, from 2008 to 2014. He is a fellow of the ACM and the IEEE, and also a member of the Academia Europaea, the Royal Danish Academy of Sciences and Letters, and the Danish Academy of Technical Sciences.