2015 IEEE International Conference on Big Data (Big Data)

Discovering Time-evolving Influence from Dynamic Heterogeneous Graphs

Chuan Hu
Department of Computer Science
New Mexico State University
New Mexico, 88003
Email: chu@cs.nmsu.edu

Huiping Cao
Department of Computer Science
New Mexico State University
New Mexico, 88003
Email: hcao@cs.nmsu.edu

***Abstract*—Influence among objects prevalently exists in graph structured data. However, most existing research efforts detect influence among objects from snapshots of homogeneous graphs. In this paper, we study a new problem of detecting time-evolving influence among objects from dynamic heterogeneous graphs. We propose a probabilistic graphical model, Time-evolving Influence Model (TIM), to capture the temporal dynamics of graphs, in which the time-evolving influence is hidden, and to leverage the information from heterogeneous graphs, with which we can improve the learned knowledge. To learn the graphical model, we design both non-parallel and parallel Gibbs sampling algorithms. We conduct extensive experiments on both synthetic and real data sets to show the effectiveness of the proposed model and the efficiency of the learning algorithms.**

***Keywords*-Graph mining, Machine learning, Dynamic graphs**

I. Introduction

Influence is prevalent among objects in graph structured data such as social networks, citation networks, and co-authorship graphs. For example, an author of a research article influences and is influenced by other researchers. Discovering influence among objects from graphs has gained increasing interests in recent years [1]–[3].

Most research efforts in influence discovery use graph snapshots. In other words, the graphs are assumed to be static. However, this assumption is not consistent with the actual situation where graphs keep changing. For example, in social networks such as Twitter and LinkedIn, new users, new tweets, and new connections are added to the networks every day. The dynamic nature of graphs needs to be considered.

A heterogeneous graph [2], [4], [5] refers to a graph containing different types of objects (nodes) and different types of connections (edges). The influence among objects of one type (e.g., Twitter users) is not only reflected in these objects' direct connections and their content similarities; it is also hidden in their connections to other types of objects (e.g., tweets) in heterogeneous graphs.

This paper is to discover *time-evolving influence* among objects from *dynamic heterogeneous graphs* which contain different types of graph nodes. We propose to build a probabilistic model, Time-evolving Influence Model (TIM), to capture the dynamic interactions among entities such as Twitter users and tweets. Then, through the learning of this probabilistic model, we can derive the dynamic influence relationships among one type of objects, Objects of Interests (OIs).

This work is supported by NSF Grants HRD-1345232 and CNS-1337884.

We focus on addressing two major challenges in formulating TIM. The first challenge is to model the dynamic nature of heterogenous graphs. In particular, we consider the situation that specific types of nodes and edges are added to the heterogenous graphs dynamically. To our best knowledge, the dynamic nature of heterogenous graphs is not captured by existing works. The second challenge is to learn the time-evolving influence efficiently. Real world graphs, such as social networks and citation networks, are huge. Designing a learning algorithm that can discover the time-evolving influence from graphs remains a challenge.

To address the first challenge, we propose to utilize multinomial distributions to capture the discrete timestamps (dynamics) that are associated with graph nodes. We also propose to separate the OIs (e.g., users), among which we need to derive the dynamic influence, and the facilitating objects (e.g., tweets), which are utilized to endorse the influence relationship among OIs. The separation of objects of interest and facilitating objects helps address the first challenge. We design efficient machine learning algorithms which can learn the time-evolving influence from dynamic heterogenous graphs to address the second challenge.

The contributions of this paper are as what follows. **(i)** We formulate the a problem of finding time-evolving influence in a general setting where graphs contain dynamic information and heterogenous types of objects. **(ii)** We design a probabilistic model, Time-evolving Influence Model (TIM), to capture the time-evolving nature of the influence and the heterogeneity of graphs. This model is not designed for a specific application. Instead, it can be applied to general dynamic heterogeneous graphs. **(iii)** We design a blocking Gibbs sampling approach to learn TIM. We also propose a useful property and an update strategy for the Gibbs sampling algorithm. Based on the property and the update strategy, a parallel Gibbs sampling algorithm is designed to further improve efficiency. **(iv)** We demonstrate the effectiveness of TIM by conducting extensive experiments on both synthetic and real data sets to show the model validity and the evolving influence discovered from real graphs. We also show the efficiency of the non-parallel and parallel learning algorithms.

The rest of this paper is organized as follows. Section II defines the research problem and the related terminologies. Section III and Section IV explain our model and the learning algorithms respectively. Section V details our experimental findings. Section VI discusses the related work. Finally, Section VII is conclusion and future work.

II. Problem definition

We use the term *heterogenous graphs* to denote the graphs with different types of graph nodes and graph edges. We focus on discovering the *influence degrees* together with their *changes over time* among one type of objects, which is of the *users*' interest and forms a set V_{OI}, from a heterogeneous graph. The other types of objects in the graph form a set of facilitating objects V_{OF}. The objects in V_{OF} are modeled to endorse the influence among OIs. Categorizing the objects in heterogenous graphs to V_{OI} and V_{OF} is useful in many applications. For instance, people who study the relationships of Twitter users (which are treated as objects of interest, OIs) can utilize users' tweets (which are modeled as facilitating objects, OFs). Similarly, when studying the influence relationships among authors (OIs) in coauthor networks, people may utilize the papers that these authors wrote (OFs).

The edges among objects of interests are denoted as E_{OI}=$\{v_i \rightarrow v_j | v_i \in V_{OI} \wedge v_j \in V_{OI}\}$. The connections between the objects of interests and facilitating objects form the set E_{OIF}=$\{v_i \rightarrow v_j | v_i \in V_{OI} \vee v_j \in V_{OI}\} - E_{OI}$. Graph edges that connect only objects in V_{OF} are not considered in our model because their contribution to the influence relationships among objects in V_{OI} is indirect. These two types of edges cover most relationships in heterogenous networks. For instance, in the Twitter network, E_{OI} captures the *follow* relationships among users (OIs) and E_{OIF} captures the behaviors that users post tweets. In the coauthor network, E_{OI} captures the coauthor relationships among researchers (OIs) and E_{OIF} captures that authors publish papers.

A heterogeneous graph is defined as G=(V_h, E_h) where $V_h = V_{OI} \cup V_{OF}$, $E_h = E_{OI} \cup E_{OIF}$ with the constraints of $V_{OI} \cap V_{OF} = \emptyset$ and $E_{OI} \cap E_{OIF} = \emptyset$.

Graph dynamics can be shown in different aspects: node addition/removal/change and edge addition/removal and change. In this work, we consider a very common dynamics in social networks and citation networks: adding new graph facilitating objects and new relationships between OIs and OFs (e.g., a new tweet is added by a Twitter, or an author publishes a new article). Other types of dynamics are less popular in these two types of graphs. For instance, once the co-authorship relationship was established, it will not be removed.

Let u and u' be two objects in V_{OI}. The **time-evolving influence** u' has over u, $\mathcal{I}(u', u)$, is defined as the influence degrees together with the corresponding time with these degrees. I.e., $\mathcal{I}(u', u) = (I(u' \xrightarrow{t_1} u), I(u' \xrightarrow{t_2} u), \cdots)$, where $I(u' \xrightarrow{t_i} u) \in (0, 1]$ for $\forall i \geq 1$ and t_i is the ith timestamp.

Given a heterogenous graph G=(V_h, E_h), our **problem** is to learn $\mathcal{I}(u', u)$ for every edge $u' \rightarrow u$ in E_{OI}.

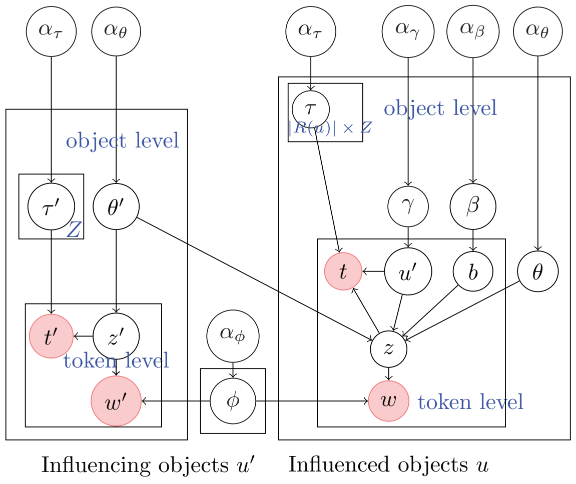


Fig. 1. Time-evolving influence model (TIM)

III. Time-evolving influence model

Sociology studies [6], [7] show that the behavior of graph users depends on network structures. Merely using the content of objects in graphs, for example TF-IDF, can not capture the overall perspectives of the time-evolving influence. We propose a graphical model, time-evolving influence model (TIM), to capture the temporal interactions among objects in a heterogeneous graph by taking both the object content and graph structures into consideration. Our TIM, shown in Fig. 1, describes the conditional dependence relationships among the observed and latent variables. The meaning of the variables is briefly described in Table I. The detailed corresponding generative process for TIM is in Fig. 2. The left (or right) hand side of the TIM represents the dependence relationships among the variables for an influencing (or influenced) object u' (or u). Since an object can be influencing and be influenced by other objects at the same time, TIM models such object as both influencing and influenced object.

TIM models the interactions among the objects of interest using two variables. First, an asymmetric boolean variable b, following a Bernoulli distribution with parameter β, is used to decide whether an object u is influenced by other objects. Second, a multinomial distribution γ is used to draw an influencing object u' that influences u. Parameters β, γ are object specific distributions. The intuition of choosing the Bernoulli distribution to model the interactions is that, when an OI is creating something (text in our problem definition), each token of the text is either influenced or not influenced by other OIs. For example, when a researcher writes an article, this article's content may come from his/her innovative idea or from other papers that he/she read. Thus, this researcher is implicitly influenced by other researchers. When a number of researchers influence an author, they may influence this author differently (i.e., with different weights). The influence weights are captured by the multinomial distribution γ.

The objects in V_{OF} are modeled through their connections E_{OIF} to the OIs. The content (tokens) of both OIs and their

TABLE I
VARIABLES FOR TIME-EVOLVING INFLUENCE MODEL

Symbol	meaning
u, u'	influenced object and influencing object in V_{OI}
w, w'	token for u and u' respectively
t, t'	observed timestamp for u and u' respectively
b	latent boolean variable
z, z'	latent topic for u and u' respectively
$R(u)$	the set of objects that influence u
T	the number of distinct discrete timestamps
Z	the number of distinct latent topics
W	the number of distinct observed tokens in objects
U	the number of objects of interests, i.e., $\|V_{OI}\|$
$\vec{\alpha}_\phi$	length-W hyperparameter $(\alpha_\phi, \cdots, \alpha_\phi)$ to generate ϕ
$\vec{\alpha}_\theta$	length-Z hyperparameter $(\alpha_\theta, \cdots, \alpha_\theta)$ to generate θ
$\vec{\alpha}_\tau$	length-T hyperparameter $(\alpha_\tau, \cdots, \alpha_\tau)$ to generate τ
$\vec{\alpha}_\gamma$	length-$L(U)$ hyperparameter $(\alpha_\gamma, \cdots, \alpha_\gamma)$ to generate γ, where $L(U) = max_u(\|R(u)\|)$
$\vec{\alpha}_\beta$	hyperparameter (α_0, α_1) to generate β
ϕ	a $Z \times W$ matrix with topic to word mixture
θ, θ'	a $U \times Z$ matrix with topic distribution
γ	a $U \times L(U)$ matrix with influencing object distribution
β	a U-vector for coin-toss distribution
τ	a $U \times (L(U)+1) \times Z \times T$ matrix with the timestamp distribution for all influenced objects on all the topics
τ'	a $U \times Z \times T$ matrix with the timestamp distribution for all the influencing objects on all the topics

```
1) for each latent topic $z$, draw topic to token mixture $\vec{\phi}_z \sim Dirichlet(\vec{\alpha}_\phi)$
2) for each influencing object $u' \in \{u'|u' \rightarrow u \in E_{OI}\}$
   a) for each topic $z$, draw $\vec{\tau}'_{u',z} \sim Dirichlet(\vec{\alpha}_\tau)$
   b) draw latent topic distribution $\vec{\theta}' \sim Dirichlet(\vec{\alpha}_\theta)$
   c) for each position $i$ of object $o' \in \{o'|u' \rightarrow o' \in E_{OIF}\}$
      i) draw latent topic $z'_i \sim Multinomial(\vec{\theta}')$
      ii) draw timestamp $t'_i \sim Multinomial(\vec{\tau}'_{u',z'_i})$
      iii) draw token $w'_i \sim Multinomial(\vec{\phi}_{z'_i})$
3) for each $u \in \{u|u' \rightarrow u \in E_{OI}\}$
   a) draw time distribution $\vec{\tau}_{u,u',z} \sim Dirichlet(\vec{\alpha}_\tau)$ for each topic
         $z$ and influencing $u' \in \{R(u) \cup u\}$,
   b) draw coin-toss distribution $\beta \sim Beta(\vec{\alpha}_\beta)$
   c) draw influencing user distribution $\vec{\gamma}_u \sim Dirichlet(\vec{\alpha}_\gamma)$
   d) draw latent topic distribution $\vec{\theta} \sim Dirichlet(\vec{\alpha}_\theta)$
   e) for each position $i$ of object $o \in \{o|u \rightarrow o \in E_{OIF}\}$
      i) draw $b_i \sim Bernoulli(\beta)$
      ii) if $b_i = 0$
         A) draw latent topic $z_i \sim Multinomial(\vec{\theta})$
         B) draw timestamp $t_i \sim Multinomial(\vec{\tau}_{u,u,z_i})$
         C) draw token $w_i \sim Multinomial(\vec{\phi}_{z_i})$
      iii) if $b_i = 1$
         A) draw influencing $u'_i \sim Multinomial(\vec{\gamma}_u)$
         B) draw latent topic $z_i \sim Multinomial(\vec{\theta}')$
         C) draw timestamp $t_i \sim Multinomial(\vec{\tau}_{u,u'_i,z_i})$
         D) draw token $w_i \sim Multinomial(\vec{\phi}_{z'_i})$
```

Fig. 2. Generative process

facilitating objects is modeled using the topic model [8]. For each token w of an influencing OI or its facilitating objects, there is a latent topic z representing its semantic meaning. The token w is generated by a multinomial distribution ϕ_z given z, which is generated by an object specific multinomial distribution θ. The influenced OIs and their facilitating objects are modeled in a similar way. The only difference is, if variable $b = 1$, an influencing OI u' is drawn using γ and the latent topic z is drawn using θ' of u'.

In *dynamic* graphs, an object of interest is associated with many timestamps t (or t') denoting the changes of this object. E.g., a user ($\in V_{OI}$) writes different tweets ($\in V_{OF}$) at different time. When an object in V_{OF} is created, every token of the object is associated with a timestamp t (or t') which is an observed variable. For example, when a user u_1 creates a tweet o_1 at time t_1, all the tokens in o_1 are associated with timestamp t_1. In this way, the TIM captures the dynamics in heterogenous networks.

TIM models timestamps as discrete variables, which was adopted in other works (e.g., [9]). The observed variable t (or t') conditionally depends on its latent topic z (or z'). For a token w (or w'), given its latent topic z (or z'), its timestamp t (or t') follows an object specific distribution τ_z (τ'_z).

IV. LEARNING TIM

A. Gibbs Sampling Algorithm

We design a Gibbs sampling approach to learn the parameters of the graphical model TIM. To draw samples from the joint distribution $p(x_1, \cdots x_K)$ of K random variables, the Gibbs sampling approach iteratively samples the value of one variable x_i by drawing from the probability distribution of this variable conditioned on the values of the remaining variables $\{x_1, \cdots, x_{i-1}, x_{i+1}, \cdots, x_K\}$, which is denoted as $x_{\neg i}$. I.e., x_i is drawn from $p(x_i|x_{\neg i})$.

$$
\begin{aligned}
& p(\vec{w}, \vec{w'}, \vec{t}, \vec{t'}, \vec{z}, \vec{z'}, \vec{u'}, \vec{b} | \vec{\alpha}_\theta, \vec{\alpha}_\beta, \vec{\alpha}_\gamma, \vec{\alpha}_\tau, \vec{\alpha}_\phi) \\
= & \int p(\vec{w}, \vec{w'} | \vec{z}, \vec{z'}; \phi) \cdot p(\phi|\vec{\alpha}_\phi)\, d\phi \\
& \cdot \int p(\vec{t}, \vec{t'} | \vec{z}, \vec{z'}, \vec{u'}; \vec{\tau}, \vec{\tau'}) \cdot p(\vec{\tau}|\vec{\alpha}_\tau) \cdot p(\vec{\tau'}|\vec{\alpha}_\tau)\, d\vec{\tau}\vec{\tau'} \\
& \cdot \int p(\vec{z}, \vec{z'} | \vec{u'}, \vec{b}; \vec{\theta}, \Theta') \cdot p(\vec{\theta}|\vec{\alpha}_\theta) \cdot p(\Theta'|\vec{\alpha}_\theta)\, d\vec{\theta}\Theta' \\
& \cdot \int p(\vec{u'}|\gamma) \cdot p(\gamma|\vec{\alpha}_\gamma)\, d\gamma \cdot \int p(\vec{b}|\beta) \cdot p(\beta|\vec{\alpha}_\beta)\, d\beta
\end{aligned}
\tag{1}
$$

A critical component in learning a model through Gibbs sampling approach is the derivation of the distributions of every variable conditioned on the other variables. Let Ω denote the set of all the latent and observed variables. I.e., $\Omega = \{\vec{w}, \vec{w'}, \vec{t}, \vec{t'}, \vec{z}, \vec{z'}, \vec{u'}, \vec{b}\}$. The joint probability of all the variables is derived as Eq. (1), given the dependence relationships of variables in Fig. 1. All the symbols used in the derivation are summarized in Table I. When deriving the equations, we use the variables' vector notations in [10] to represent a set of variables. Fig. 3 shows the conditional distributions that we derived for each variable. Due to space limitation, we do not include all the theoretical proof and derivation steps here. These details can be found in technical report [11]. In particular, the derivation steps of equations in Fig. 3 can be found in Sections 3.1 and 3.2 [11]. In these conditional distributions, the count $N_{x_1, x_2, \cdots, x_K}(val_1, val_2, \cdots, val_K)$ refers to the number of samples for x_1=val_1, x_2=val_2, $\cdots$, x_K=val_K.

The detailed meaning of each count is also shown in Table 2 of [11]. The Gibbs sampling algorithm updates the conditional probability using the equations in Fig. 3 and updates the different counts $N_{x_1,x_2,\cdots,x_K}(val_1, val_2, \cdots, val_K)$ in each iteration following the generative process. The steps of the Gibbs sampling algorithm are detailed in Section 3.3 of [11].

Gibbs sampling algorithms can be implemented differently. To improve the efficiency, we implement our algorithm using a blocking strategy [12], which samples several variables together as a block, conditioned on the remaining variables.

Complexity. For an influenced user u, we need to sample the latent topics and possible influencing users. For an influencing user u', only the latent topics need to be sampled. Thus, the total asymptotical time complexity of the sampling process for one iteration is $O(L' \cdot Z + L \cdot Z \cdot L(u))$, where L' and L are the total number of tokens (or positions) in the influencing and influenced objects respectively.

B. Parallel Gibbs Sampling Algorithm

Fast inference algorithms [13]–[15] and parallel algorithms [16]–[18] were proposed to learn topic models by taking advantage of the access patterns of topic models or the weak dependencies among variables. However, directly applying these topic-model learning algorithms to TIM is not straightforward because TIM models more variables and more complex relationships than the basic topic models.

Directly parallelizing the Gibbs sampling algorithm described above by drawing each variable concurrently is not efficient. The Gibbs sampling algorithm needs to draw samples for each variable iteratively by utilizing the updating equations shown in Fig. 3. The update of one count blocks the sampling of the other variables that also depends on this count. For example, after we draw z_i using Eq. (6) we need to update the counts $N_{z,w}, N'_{z',w'}, N_z, N'_{z'}$ used in Eq. (6). Because these counts are also used in Eq. (7), the sampling of variable z'_i will be blocked until all the counts used in Eq. (6) are updated. However, the sampling of b_i, u'_i in Eq.s (2) (3) (4) are not blocked by the count updating in Eq. (6). This is because the counts used in Eq.s (2) (3) (4) are not used in Eq. (6).

Let us use *shared counts* to denote the counts that are used in the calculations of multiple conditional probabilities. We also use *s-variable* to denote a variable whose conditional probability calculation uses shared counts. If the variables in Fig. 3 are sampled in parallel using multiple threads, once the sample for one s-variable x_i is drawn, the threads running the sampling processes of other s-variables that use the same shared counts as x_i need to wait. These threads can continue only after all the shared counts used by x_i are updated. We call the procedure of updating shared counts *synchronization* in our problem context. The synchronization of the shared counts blocks the calculations of the conditional probabilities of s-variables. Thus, directly parallelizing the sampling process will suffer from synchronization overhead.

We observe a useful property that help us to design the parallel Gibbs sampling algorithm.

Object-dependent property. A conditional probability is *object-dependent* when its calculation depends on counts of only one object, but does not depend on shared counts.

In our problem, the conditional probabilities that satisfy object-dependent property are Eq.s (2) (3) (4). Thus, they can be calculated in parallel without synchronization. However, the counts $N_{z,w}, N'_{z',w'}, N_z, N'_{z'}$ are shared counts, thus they need to be synchronized.

To reduce the synchronization overhead of shared counts, we propose a *lazy-update strategy* to update the shared counts. Typical Gibbs sampling algorithms immediately update the counts after drawing samples for *each variable*. This immediate updating creates much synchronization overhead in parallel algorithm design. Newman et al. [18] first investigate distributed algorithms for topic models, in which the shared counts are copied into different sampling threads and the shared counts are updated after all threads terminate. Inspired by their work, we propose a strategy as following.

Lazy-update strategy for shared counts. Our strategy updates shared counts in a lazy way. In particular, when drawing samples for the variables in parallel, the shared counts stay the same during one iteration of the Gibbs sampling. And they are only updated after finishing drawing samples for all the variables in one iteration. Experimentally (Section V-F Fig. 11 (a)), we show that this strategy can achieve similar effect as our serial Gibbs sampling algorithm, which verifies the correctness of this parallel algorithm.

Our parallel Gibbs sampling algorithm is designed by utilizing the object-dependent property and the lazy-update strategy for shared counts. The procedure of the parallel Gibbs sampling algorithm is illustrated in Fig. 4. For each object of interest $u \in V_{OI}$, a thread is created to sample the variables for all the tokens in the profile of u. For a heterogeneous graph containing U (U=$|V_{OI}|$ as defined in Table I) objects of interest, the parallel algorithm creates U threads. The threads for all the objects of interest are submitted to a thread pool of a fixed size (# of cores). Among these U threads, only a few number (# of cores) of threads are actually running. The other threads are either terminated or waiting. Within one thread, the non-parallel Gibbs sampling algorithm is used to draw samples for the variables. The counts in object-dependent conditional probabilities are updated after each variable is sampled because these counts do not block sampling. However, the shared counts are fixed and not updated before all threads terminate. The shared counts are updated after all the threads terminate. The updated shared counts are used in the next iteration of the parallel Gibbs sampling algorithm until convergence.

C. Derivation of influence over time

The influence that u' has over u at timestamp t, $I(u' \xrightarrow{t} u)$, is derived from the learned parameters of TIM. Let $x_{[idx_1,\cdots,idx_d]}$ represent the learned value of the d-dimensional variable x at index $idx_1, \cdots, idx_d$. E.g., $\gamma_{[u_1,u'_2]}$ (= $\gamma[u_1][u'_2]$) denotes the value in the matrix γ when the influenced user is u_1 and the influencing user is u'_2. $I(u' \xrightarrow{t} u)$ is represented as $p(u'|t, u)$. Using Bayes' rule, we can derive Eq. (8).

$$p(b_i=0|\Omega-\{b_i\}) \propto \frac{N_{u,b}(u_i,0)+\alpha_0-1}{N_u(u_i)+\alpha_0+\alpha_1-1} \cdot \frac{N_{u,z,b}(u_i,z_i,0)+\alpha_\theta-1}{N_{u,b}(u_i,0)+Z\alpha_\theta-1} \cdot \frac{N_{u,z,t,b}(u_i,z_i,t_i,0)+\alpha_\tau-1}{N_{u,z,b}(u_i,z_i,0)+T\alpha_\tau-1} \tag{2}$$

$$p(b_i=1|\Omega-\{b_i\}) \propto \frac{N_{u,b}(u_i,1)+\alpha_1-1}{N_u(u_i)+\alpha_0+\alpha_1-1} \cdot \frac{N_{u',z',b}(u'_i,z_i,1)+N_{u',z'}(u'_i,z_i)+\alpha_\theta-1}{N_{u',b}(u'_i,1)+N_{u'}(u'_i)+Z\alpha_\theta-1} \cdot \frac{N_{u,u',z,t,b}(u_i,u'_i,z_i,t_i,1)+\alpha_\tau-1}{N_{u,u',z,b}(u_i,u'_i,z_i,1)+T\alpha_\tau-1} \tag{3}$$

$$p(u'_i|\Omega-\{u'_i,b_i\},b_i=1) \propto \frac{N_{u,u',b}(u_i,u'_i,1)+\alpha_\tau-1}{N_{u,b}(u_i,1)+R(u)\alpha_\tau-1} \cdot \frac{N_{u',z',b}(u'_i,z_i,1)+N_{u',z'}(u'_i,z_i)+\alpha_\theta-1}{N_{u',b}(u'_i,1)+N_{u'}(u'_i)+Z\alpha_\theta-1} \cdot \frac{N_{u,u',z,t,b}(u_i,u'_i,z_i,t_i,1)+\alpha_\tau-1}{N_{u,u',z,b}(u_i,u'_i,z_i,1)+T\alpha_\tau-1} \tag{4}$$

$$p(z_i|\Omega-\{z_i,b_i\},b_i=0) \propto \frac{N_{z,w}(z_i,w_i)+N_{z',w'}(z_i,w_i)+\alpha_\phi-1}{N_z(z_i)+N_{z'}(z_i)+W\alpha_\phi-1} \cdot \frac{N_{u,z,b}(u_i,z_i,0)+\alpha_\theta-1}{N_{u,b}(u_i,0)+Z\alpha_\theta-1} \cdot \frac{N_{u,z,t,b}(u_i,z_i,t_i,0)+\alpha_\tau-1}{N_{u,z,b}(u_i,z_i,0)+T\alpha_\tau-1} \tag{5}$$

$$p(z_i|\Omega-\{z_i,b_i\},b_i=1) \propto \frac{N_{z,w}(z_i,w_i)+N_{z',w'}(z_i,w_i)+\alpha_\phi-1}{N_z(z_i)+N_{z'}(z_i)+W\alpha_\phi-1} \cdot \frac{N_{u',z'}(u'_i,z_i)+N_{u',z',b}(u'_i,z_i,1)+\alpha_\theta-1}{N_{u'}(u'_i)+N_{u',b}(u'_i,1)+Z\alpha_\theta-1} \cdot \frac{N_{u,u',z,t,b}(u_i,u'_iz_i,t_i,1)+\alpha_\tau-1}{N_{u,u',z,b}(u_i,u'_i,z_i,1)+T\alpha_\tau-1} \tag{6}$$

$$p(z'_i|\Omega-\{z'_i\}) \propto \frac{N_{z,w}(z'_i,w'_i)+N_{z',w'}(z'_i,w'_i)+\alpha_\phi-1}{N_z(z'_i)+N_{z'}(z'_i)+W\alpha_\phi-1} \cdot \frac{N_{u',z'}(u'_i,z'_i)+N_{u',z',b}(u'_i,z'_i,1)+\alpha_\theta-1}{N_{u'}(u'_i)+N_{u',b}(u'_i,1)+Z\alpha_\theta-1} \cdot \frac{N_{u',z',t'}(u'_iz'_i,t'_i)+\alpha_\tau-1}{N_{u',z'}(u'_i,z'_i)+T\alpha_\tau-1} \tag{7}$$

Fig. 3. Gibbs sampling updating equations (u_i in Eq.s (2)-(6) refers to the influenced user for which the variables are sampled; u'_i in Eq. (7) refers to the influencing user for which the variables are sampled.)

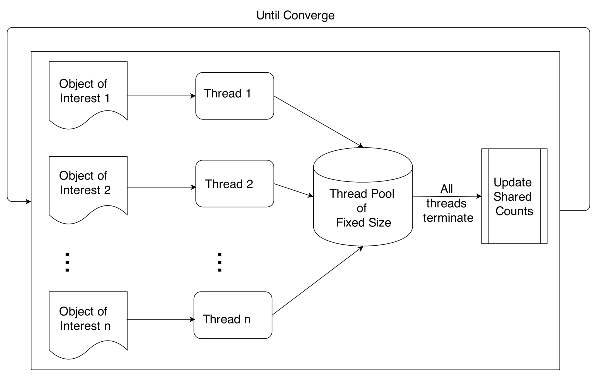


Fig. 4. Diagram of parallel Gibbs sampling algorithm

$$p(u'|t,u) \propto p(t|u',u)\cdot p(u'|u) = (\sum_z \tau_{[u,u',z,t]})\cdot \gamma_{[u,u']} \tag{8}$$

The Bayesian estimation of γ and τ is calculated according to Eq. (9).

$$\begin{aligned} &p(\vec{\gamma}|\vec{u}';\vec{\alpha}_\gamma) = \frac{p(\vec{u}'|\vec{\gamma})\cdot p(\vec{\gamma}|\vec{\alpha}_\gamma)}{p(\vec{u}';\vec{\alpha}_\gamma)} \propto \prod_u \frac{\prod_{u'\in R(u)} \gamma_{u,u'}^{N_{u,u',b}(u,u',1)+\alpha_\gamma-1}}{\Delta(\vec{N}_{u,u',b}+\vec{\alpha}_\gamma)} \\ &\vec{\gamma}_u \sim Dir(\vec{N}_{u,u',b=1}+\vec{\alpha}_\gamma) \\ &\vec{\tau}_{u,u',z} \sim Dir(\vec{\tau}_{u,u',z}|\vec{N}_{u,u',z,t,b=1}+\vec{\alpha}_\tau) \end{aligned} \tag{9}$$

The parameters γ and τ in Eq. (8) are substituted with their expectation values. Then, $I(u' \underset{\rightarrow}{t} u)$ is rewritten as Eq. (10). A detailed proof and derivation of Eq. (10) are presented in Section 4 of the technical report [11].

$$I(u' \underset{\rightarrow}{t} u) \propto \left(\sum_z \frac{N_{u,u',z,t,b=1}+\alpha_\tau}{N_{u,u',z,b=1}+T\alpha_\tau}\right)\cdot \frac{N_{u,u',b=1}+\alpha_\gamma}{N_{u,b=1}+L(u)\alpha_\gamma} \tag{10}$$

Inspired by [2], which discovers topic level influence $\Phi_u(u'|z)$ representing the scalar influence value from u' to u on topic z, we define *topic-level time-evolving influence* as the influence degrees together with the corresponding time on a specific topic. I.e., $\mathcal{I}_z(u',u) = (I(u' \underset{z}{\xrightarrow{t_1}} u), I(u' \underset{z}{\xrightarrow{t_2}} u), \cdots)$, where $I(u' \underset{z}{\xrightarrow{t_i}} u) \in (0,1]$ ($\forall i \geq 1$) represents the influence from u' to u on topic z at timestamp t_i. The topic-level influence can show the influence in different dimensions. For example, on citation networks, even if the overall influence from author u' to author u stays the same over time; the influence on different topics may change over time.

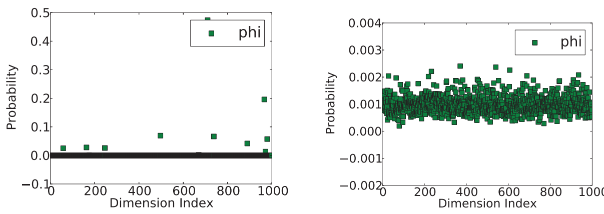
Similar to the derivation of Eq. (10), the topic level influence $I(u' \underset{z}{\xrightarrow{t}} u)$ is derived as

$$I(u' \underset{z}{\xrightarrow{t}} u) \propto \frac{N_{u,u',z,t,b=1}+\alpha_\tau}{N_{u,u',z,b=1}+T\alpha_\tau}\cdot \frac{N_{u,u',b=1}+\alpha_\gamma}{N_{u,b=1}+L(u)\alpha_\gamma} \tag{11}$$

V. Experimental results

A. Data sets

Synthetic data. We generate a synthetic data set using the RMAT algorithm in [19] since it can generate graphs similar to real world social networks. The synthetic graph has ∼ 300 objects of interest and ∼ 50K facilitating objects. The content of these objects is generated using the generative process in Fig. 2 with W=1000, Z=10, and T=100. The hyperparameters are set to be $\alpha_\theta = 0.1$, $\alpha_\tau = 0.01$, $\alpha_\phi = 0.0001$, $\alpha_\beta = 0.5$, $\alpha_\gamma = 0.05$. Thees hyperparameter values are set to be less than 1 because the Dirichlet distribution with small hyperparameter values (less than 1) generates a sparse multinomial distribution (Fig. 5(a)) while larger hyperparameter values generate dense distributions (Fig. 5(b)). A sparse multinomial distribution means that most words have very small probability (close to zero) to occur which is consistent with the observations in topic models [8].



(a) W=1000, $\alpha_\phi = 0.001$ (sparse distribution)

(b) W=1000, $\alpha_\phi = 10$ (dense distribution)

Fig. 5. ϕ distribution generated using different α_ϕ

Twitter50000 data is crawled from the Twitter website from Oct. 1 2014 to May. 18 2015. It contains 50000 Twitter users and their recent 200 tweets. The total number of tokens is ∼90M and the number of distinct tokens is ∼200K. Each

Twitter user in this data set follows ∼200 users on average. Then, we extract subgraphs with 100, 500, 1K, 2K, 5K, 10K, 20K, 30K and 40K users. These nine subgraphs are used to conduct the scalability test of both non-parallel and parallel Gibbs sampling algorithms.

DBLP data. We extract heterogeneous co-authorship and citation graphs from an augmented DBLP data set [20]. This small graph, denoted as *Cite1000*, contains ∼1.5K authors with ∼100K articles, in which there are ∼10K distinct tokens and ∼4M tokens in total.

B. Evaluation measures

Let an object u consist of a list of tokens with associated timestamps, which come from the facilitating objects connected with u in V_{OF} ($\{o|u \to o \in E_{OIF}, u \in V_{OI}\}$). Then u can be represented as $(w_1, t_1), \cdots, (w_n, t_n)$ where n is the number of tokens for u.

(1) Log-likelihood is commonly used to measure the probability that the observed data is generated from a set of parameter values. The log-likelihood (*llh*) for an influencing object u' and an influenced object u is calculated as follows.

$$\mathrm{llh}(\Omega|u') = p(u'|\Omega) = \prod_{(w',t')\in u'} \sum_{z'} \theta'_{[u',z']} \cdot \phi_{[z',w']} \cdot \tau'_{[u',z',t']}$$

$$\mathrm{llh}(\Omega|u) = p(u|\Omega) = \prod_{(w,t)\in u} ((1-\beta_{[u]})(\sum_z \theta_{[u,z]} \cdot \phi_{[z,w]} \cdot \tau_{[u,u,z,t]}) + \beta_{[u]}(\sum_{u'} \sum_z \gamma_{[u,u']} \cdot \theta'_{[u',z]} \cdot \phi_{[z,w]} \cdot \tau_{[u,u',z,t]}))$$

(2) Perplexity measures how well a probability distribution predicts a sample. The perplexity for u' and u is calculated as $\mathrm{perp}(u') = e^{-(\log p(u'|\Omega))/n'}$ and $\mathrm{perp}(u) = e^{-(\log p(u|\Omega))/n}$.

(3) Kullback-Leibler(KL) divergence is a non-symmetric measure of the difference between two probability distributions. The smaller KL divergence of two distributions is, the closer these two distribution are from each other. We use it to measure the difference of the learned parameter values (e.g., $\hat{\phi}$) from the ground truth values of a parameter (e.g., ϕ), which is used to generate the synthetic data. The KL divergence of the learned $\hat{\phi}$ from the ground truth ϕ is calculated as $D_{KL}(\phi||\hat{\phi}) = \sum_i \phi(i) \ln \frac{\phi(i)}{\hat{\phi}(i)}$.

(4) Retweet prediction. Inspired by [21]–[23], we conduct retweeting predictions to show that the temporal influence relationships we learned is effective.

We first define the retweet prediction problem. When an influencing user u' posts a new tweet o at time t', we can use the probability learned from our model to estimate whether another user (which may be influenced by u') will retweet it in a future time t ($t > t'$). The retweet prediction problem becomes a binary classification problem.

We define two probabilities for each token w for an object o that is written by a user u at time t. When w is a newly created token, the probability is calculated using Eq. (12). When w is created through retweeting, the probability is derived through Eq. (13). In these two equations, the conditional probabilities represent the posterior.

$$\begin{aligned} p_0(u,o,w,t) &= p(u \text{ creates a new token } w \text{ at } t) \\ &= \sum_z p(b{=}0|z,u,w,t) \cdot p(z|u,w,t) \\ &= \sum_z (1-\beta_{[u]}) \cdot \theta_{[u,z]} \cdot \tau_{[u,u,z,t]} \cdot \phi_{[z,w]} \end{aligned} \tag{12}$$

$$\begin{aligned} p_1(u,u',o,w,t) &= p(u \text{ retweets } w \text{ from } u' \text{ at } t) \\ &= \sum_z p(b{=}1|z,u',u,w,t) \cdot p(z|u',u,w,t) \cdot (u'|u,w,t) \\ &= \beta_{[u]} \cdot \gamma_{[u,u']} \cdot (\sum_z \theta'_{[u',z]} \cdot \tau_{[u,u',z,t]} \cdot \phi_{[z,w]}) \\ p_1(u,o,w,t) &= \sum_{u'\in R(u)} p_1(u,o,u',w,t) \end{aligned} \tag{13}$$

For each tweet o, we maintain two counts $b0$ and $b1$ for the positions that b is predicted as 0 and 1. In particular, if $\frac{p_0(u,o,w,t)}{p_1(u,o,w,t)} > \delta_0$, count $b0$ is incremented; otherwise, $b1$ in increased by 1. After scanning over all the tokens in tweet o, if $\frac{b1}{b0} \geq \delta_1$, we predict tweet o to be newly created; otherwise, o is predicted to be retweeted. In our experiments, the mean value of the learned β is used for δ_1. δ_0 value is tuned with different values to get better results. In our experiments, δ_0 is set to 0.31 after parameter tuning.

(5) Time-evolving influence is plotted for users in DBLP data.

(6) Model running time and speedup ratio is reported to show the efficiency and scalability of the learning algorithm.

C. Validity of the model

We examine the convergence and the correctness of our learning algorithm by using a simulation with a similar idea in [9] [24]. In the simulation, the Gibbs sampling algorithm is performed on the *Synthetic* data. The parameters that are used to generated the *Synthetic* data are treated as the ground truth. Comparison between the parameters learned by the proposed Gibbs sampling algorithm and the ground truth are presented to validate the proposed model.

Convergence of the learning algorithm. We check the convergence of the sampling process by examining the trend of the log-likelihood and the perplexity (Fig. 6). The process converges after about iteration 500 since both the log-likelihood and the perplexity stabilize after that.

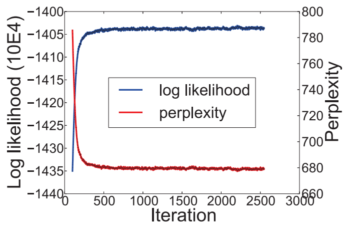
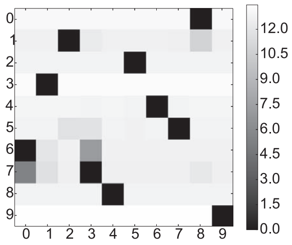


Fig. 6. Log-likelihood and perplexity (*Synthetic* data set)


(a) KL-divergence

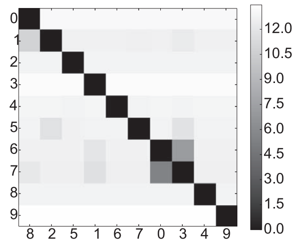
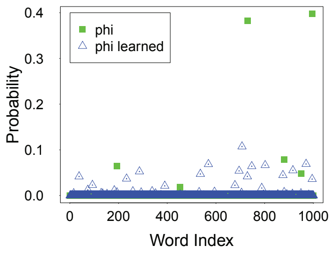

(b) Rearranged KL-divergence

Fig. 7. KL-divergence (*Synthetic* data set)

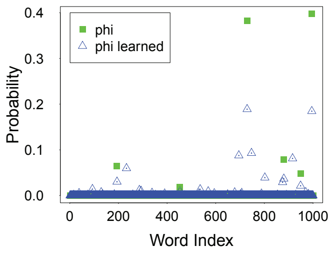
Correctness of the learning algorithm. To show the correctness of the proposed Gibbs sampling algorithm, we plot the

KL divergence of the learned parameters $\hat{\phi}$ from the ground truth values ϕ ($p(t|z)$). Fig. 7 shows the KL divergence of all the $\hat{\phi}_m$s (learned parameters) from all the ϕ_ns (ground truth parameters), where $m, n \in \{1, \dots Z\}$. The x and y axes are the indexes for ϕ and $\hat{\phi}$ respectively. The darker the pixel at (m, n) is, the smaller the KL-divergence of $\hat{\phi}_m$ from ϕ_n is which means the more similar $\hat{\phi}_m$ is to ϕ_n.

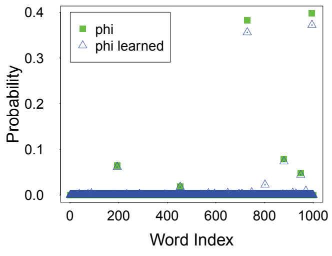
Fig. 7(a) shows the KL-divergence between each pair of $\hat{\phi}_m$ and ϕ_n. After rearranging the columns in Fig. 7(a), we get Fig. 7(b). For each ϕ_n, its corresponding learned parameter is $\hat{\phi}_m$ with the minimal KL divergence. As shown in Fig. 7, each learned parameter $\hat{\phi}_m$ has a unique matching ground truth parameter ϕ_n. These results indicate that our algorithm can correctly learn the knowledge from data.



(a) Iteration 0



(b) Iteration 40



(c) Iteration 200

Fig. 8. Learned $\hat{\phi}_3$ vs. ground truth ϕ_5 on Synthetic data

We further examine the details of the learning process by plotting $\hat{\phi}_3$ and its matching ground truth ϕ_5 in Fig. 8. At the beginning of the learning process, $\hat{\phi}_3$ is very random and does not match well with ϕ_5. As the learning process continues, $\hat{\phi}_3$ quickly stabilizes and converges to ϕ_5, which is consistent with the trend of log-likelihood and perplexity in Fig. 6. After the model converges, most values are close to zero because the hyperparameter values that are used to generate the ground truth parameter are 0.01. Other parameters in the model show similar behaviors.

D. Retweet Prediction

Naive baseline. We design a naive baseline method to predict the retweeting behavior. This baseline method utilizes the similarity in tweets by leveraging a temporal factor, which is denoted as a time-dependent text similarity. Given a user u, a timestamp t, and a time window q, $\vec{u}_t^q$ represents the tf-idf text vector of the tokens in all the objects which are created by u within the time interval $[t-q, t+q]$. Given a time window q, the time-dependent text similarity between a user u and a tweet o (which is written at time t) is defined as $tsim_q(u, o, t) = cosine(\vec{o}, \vec{u}_t^q)$. When $q = \infty$, we are comparing the text similarity between u and o. For each test tweet o_{test} posted by a specific user u, if $tsim_q(u, o_{test}, t) \geq tsim_q(u', o_{test}, t)$ for $\forall u'$, the tweet o_{test} is predicted as self posted; otherwise, o_{test} is predicted as being retweeted.

LRC-BQ [23]. LRC-BQ is the state-of-the-art retweet behavior prediction model. LRC-BQ uses logistic regression model to predict retweet behavior. The features that LRC-BQ utilizes are structure and content features from social networks. However, LRC-BQ does not take consider any temporal features that can describe the graph dynamics.

On Twitter1K data set, we reserve ~200 tweets as test objects, where half of these objects are retweeted tweets and the other half are self-posted tweets. For our model, we first learn the model parameters by running the algorithm presented in Section IV-A. Then, we make predictions by Eq.s (13) and (12). The prediction accuracy is reported in the TIM row in Table II. For the naive baseline approach, we make predictions based on the procedure described above by varying different time windows q. In Table II, the tf-idf rows report the results from the naive baseline approach and the LRC-BQ row shows the results reported by [23].

TABLE II
COMPARISON OF RETWEET PREDICTION PERFORMANCE

model	accuracy	precision	recall	F_1
tf-idf ($q = \infty$)	0.55	0.55	1	0.71
tf-idf ($q = 5$)	0.65	0.55	0.8	0.65
tf-idf ($q = 3$)	0.65	0.55	0.8	0.65
tf-idf ($q = 1$)	0.65	0.56	0.8	0.66
LRC-BQ	0.71	0.69	0.77	0.73
TIM	0.7	0.7	0.78	0.74

The results show that the proposed TIM outperforms the baseline approaches. However, we also observe that the retweeting accuracy of TIM is not high on the absolute value. As future work, we will design more accurate retweeting prediction strategy. For the naive baseline approach, a higher prediction accuracy can be achieved when using a smaller time window (i.e.., a smaller q). It is consistent with the intuition that retweets happens in the near future. Our retweet prediction strategy performs similar to (slightly better than) the LRC-BQ approach. Note that our TIM detects time-evolving influence which LRC-BQ is not able to discover.

E. Case studies on time-evolving influence

Case studies are conducted to show that TIM can capture the overall and topic-level time-evolving influence between researchers. We perform TIM on *Cite1000* and choose four

TABLE III
TOP 15 WORDS FROM 3 TOPICS z_4, z_8, AND z_{14}

topic	top 15 words
z_4	query, model, search, stream, web, show, online, one, use, find, match, information, data, structure, user
z_8	method, graph, large, data, real, mine, pattern, propose, find, use, analysis, time, cluster, set, correlate
z_{14}	query, data, database, xml, tree, optimization, efficiency, join, index, path, text, process, technique, update, evaluation

TABLE IV
OVERALL INFLUENCE BETWEEN S AND P (TRUNCATED TO 2 DIGITS)

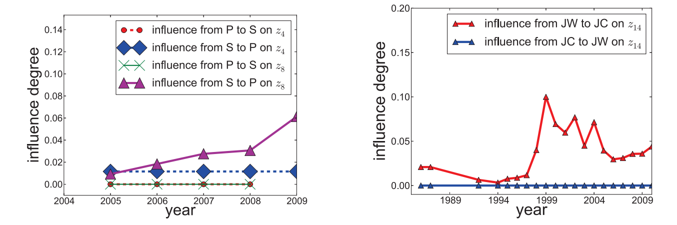
year	2005	2006	2007	2008	2009
influence from S to P	0.07	0.12	0.07	0.25	0.09
paper # that P cited S	3	4	2	11	3
paper # that P cited in total	119	55	154	220	156
influence from P to S	0.00	0.00	0.00	0.00	
paper # that S cited P	1	1	1	7	
paper # that S cited in total	98	144	118	236	

researchers (whom we can verify subjectively) out of the ∼1.5K authors to exemplify the learned knowledge. The first pair of researchers we chose are "Jimeng Sun" (S for short) and "Spiros Papadimitriou" (P for short), because they published reasonable number of articles for analysis: the author S published 47 articles with the focus on time series and graph mining, and the researcher P published 41 articles on mining streaming data and dynamic graphs.

TIM can easily capture the *overall time-evolving influence* among users. Table IV shows the overall influence (Eq. (10)) and the citation counts between S and P. The influence from S to P fluctuates over time from 2005 to 2009. This is consistent with the times that P cited S. On the other hand, the influence from P to S is low (close to zero) over time because S cited P only once in these years (except 2008). In 2008, the influence from P to S is low because S cited other researchers 236 times, out of which S cited P only 7 times. The influence degree 0 is the actual influence rounded to two digits. This zero value does not mean that the influence does not exist. It indicates there is a weak influence from P to S from 2005 to 2008.

TIM can capture the *topic-level time-evolving* influence (Eq. (11)), which cannot be extracted from citation counts. Fig. 9(a) shows how the influence between researchers S and P evolves on topic 4 (z_4 in Table III, interpreted as "query streaming data") and topic 8 (z_8, interpreted as "mining time series and graphs"). The figure shows that the influence between authors is not symmetric, which is consistent with the reality. On topic z_4, the influence from S to P (dashed blue line) and from P to S (dashed red line) is constantly low since P does not work on z_4. $I(S \xrightarrow[z_4]{t} P)$ (the influence from S to P) is slightly higher than $I(P \xrightarrow[z_4]{t} S)$ because P is influenced by more researchers than S (i.e., $|L(P)| > |L(S)|$). According to Eq. (10), the influence is bigger when $L(u)$ is smaller. On topic z_8, the influence from P to S (solid green line) is constantly low because S cited only one paper each year from P on z_8. Different from this trend, the influence from S to P on z_8 (solid purple line) keeps increasing. This trend is consistent with the publication records of S and P on topic z_8, which keep increasing from year 2006 to 2009.

The TIM model is able to discover *long-term evolving influence* which may fluctuate. We show the results for the second pair of researchers "Jennifer Widom" (JW) and "Jon Chomicki" (JC) who have long publication history (1986-2010) in the database area. Fig. 9(b) shows that the influence from JW to JC on z_{14} (Table III, interpreted as "XML query processing"), reaches peak at 1999, which is consistent with the research trend in topic z_{14}. While the influence from JC to JW keeps low over time because JW did not cite any paper from JC. Case studies on Twitter data show similar results, which are omitted due to space limitation.

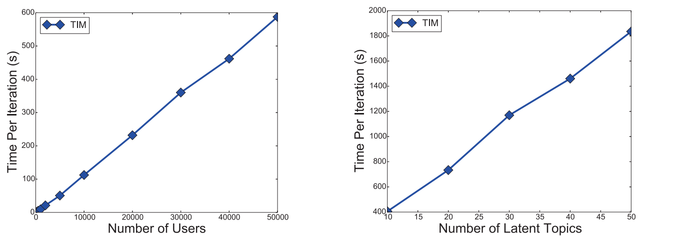


(a) Influence between S and P (b) Influence between JC and JW

Fig. 9. Topic-level time-evolving influence between researchers

F. Efficiency

This section demonstrates the efficiency of the algorithms presented in Section IV.

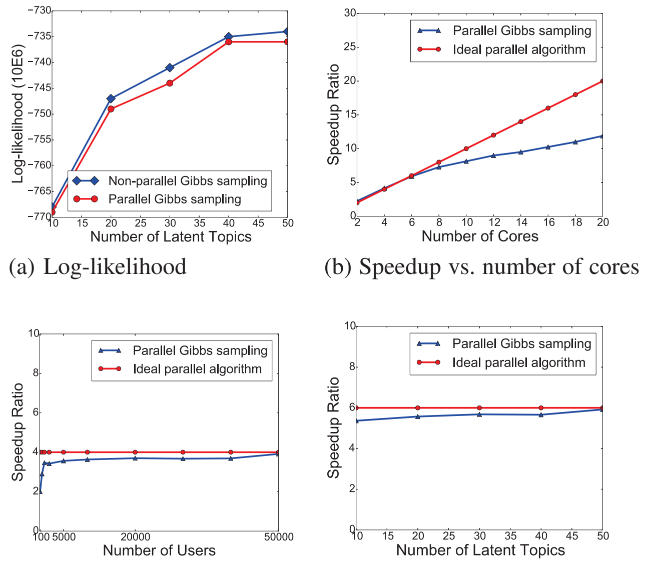


(a) Time per iteration vs. data size (b) Time per iteration vs. model complexity

Fig. 10. Efficiency of non-parallel Gibbs sampling algorithm (Twitter data)

Fig. 10 shows the running time of our sampling algorithm per iteration on *Twitter50000* data set. Fig. 10(a) and (b) show that the per-iteration running time grows linearly in the number of OIs in the graphs and the number of latent topics respectively. All these experimental results are consistent with our time complexity analysis in Section IV.

We design experiments to show the performance of the parallel Gibbs sampling algorithm (to learn TIM). Fig. 11(a) compares the log-likelihood of the models that are learned using the non-parallel and the parallel Gibbs sampling algorithms on *Twitter50000* data set. There is no essential difference between the log-likelihood values of the models. The similar log-likelihood values indicate that the parallel algorithm can correctly learn the TIM model as the non-parallel one. Figs. 11(b-d) show the efficiency of the parallel algorithm that we describe in Section IV-B. Both the non-parallel and the parallel Gibbs sampling algorithms are tested on the *Twitter50000* data set and its nine sub-graphs. We compare the speedup ratio of our parallel algorithm with the ideal parallel speedup ratio. The ideal speedup ratio of a parallel algorithm running on n cores over its corresponding serial algorithm is n, according to Amdahl's Law [25]. It can be achieved when each step of a serial algorithm can be parallelized.



(a) Log-likelihood

(b) Speedup vs. number of cores

(c) Speedup vs. data size

(d) Speedup vs. model complexity

Fig. 11. Scalability of parallel Gibbs sampling algorithm (Twitter data)

Fig. 11(b) shows the speedup ratio of the parallel Gibbs sampling algorithm with the increase of the number of cores that are used in running the parallel algorithm. When the number of cores is small (2-8), the speedup ratio is close to ideal. When the number of cores is more than eight, the speedup ratio is a little less than the ideal ratio. This decrease of performance is due to the overhead of thread scheduling and communication costs. Despite the decrease, it is observed that the speedup ratio still grows linearly in the number of cores. Overall, Fig. 11(b) shows that the speedup ratio of our parallel algorithm scales well in the number of cores.

We also test the performance of the parallel sampling algorithm on various sizes of data to demonstrate how the algorithm scales in the increase of graph sizes. For this test, we fix the number of cores to be four, and utilizing the *Twitter50000* and its nine sub-graphs. The results are shown in Fig. 11(c). The figure shows that the parallel algorithm is only about twice faster than its non-parallel counterpart on small data sets. It is because the thread scheduling overhead takes more time than the computation time on small data sets. When the data set size grows, the speedup ratio of the parallel algorithm becomes close to ideal (four). This indicates that the parallel algorithm has better speedup on larger data sets. This figure also shows that the parallel implementation scales well on real world large data sets.

We further test how the parallel algorithm scales when changing the model complexity (in particular, the number of latent topics Z). We fix the number of cores to be six and the data set to be *Twitter50000*. The results are shown in Fig. 11(d). It shows that when the model becomes more complicated (with a larger Z), the parallel Gibbs sampling can achieve better speedup ratio. We also observe that the performance of the parallel sampling under different model complexities is close to the ideal speedup ratio. This demonstrates that the parallel Gibbs sampling algorithm can achieve better speedup ratio when the model complexity increases.

VI. Related works

Several probabilistic models are proposed to capture influence among objects from static *homogeneous* graphs. Dietz et al. [1] present a graphical model to learn the citation influence that one research article has over the other research articles. Hu et al. [3] introduce aspect-level influence models to detect both influence aspects and the influence degrees on specific aspects from graphs. Kataria et al. [21] [26] present a generative process to model correlations among linked corpus. These works create probabilistic models from graphs with homogeneous graph nodes (i.e., the same types of objects in the graphs). Along the direction of influence detection, Wang et al. [27] propose a different approach to detect the top influential users in communities by aggregating the re-tweet counts among users.

Several techniques detect influence from *static heterogeneous* networks. Liu et al. [2] utilize two types of graphs (citation graphs and co-authorship graphs) to discover influence among users at topic levels. Tang et al. in [28] present approaches to infer the relationship type for each edge in a target graph given a source graph whose edges are partially annotated with relationship types (e.g., family, colleague). These works do not consider the dynamic nature of graphs.

Many approaches are presented to model temporal information on text corpus. Wang et al. [29] present the Topics Over Time (TOT) model, a variant of the topic model [8], to capture the temporal evolution of topics in document collections. Wang et al. [30] detect topic changes over time from document corpus or streaming documents by formulating an optimization problem. These two methods are built on document corpus which does not encode graph structures.

Modeling temporal information in graph structured data has also been studied. Rossi et al. [31] learn a behavior transition model to detect the role changes of graph nodes over time from graphs with dynamic edge additions and deletions. Hu et al. [9] propose a temporal topic model to simultaneously detect communities, community topics, and the temporal changes of community topics from social networks. This model does not consider the interactions among community users, which are modeled in our work. Iwata et al. [32] present a shared cascade Poisson processes to discover latent relationships among users from the series of item-adoption actions. This work has the same objective as ours to discover user relationships. However, the input data does not encode user relationships, which is different from our model. Tan et al. [22] present a graphical model (in particular, time-varying factor model) to capture user retweeting actions on a particular topic in social networks. Goyal et al. [33] propose five probabilistic models to learn the probability of users adopting an action in a social network. They work on static graphs and a series of user-adoption actions which is different from the scenario that we deal with (dynamic heterogeneous graphs). The narrowly defined user actions cannot be directly generalized to other connections among graph nodes. Wang et al. [34] present a factor graph model to learn advisor-advisee relationship and the temporal period for each relationship from coauthor networks. In [34], the application-specific features (co-authorship) and their properties can not be directly generalized to capture other types of graphs.

VII. Conclusions and future work

We study the problem of learning time-evolving influence among nodes of interests from dynamic heterogeneous graphs. We design a probabilistic time-evolving influence model (TIM) to abstract the heterogeneous types of objects and their dynamic interactions. We design a blocking Gibbs sampling approach and a parallel Gibbs sampling algorithm to learn TIM. We demonstrate the effectiveness of this model on both synthetic and real data. In particular, we show the correctness of the learning algorithms and the discovered time-evolving influence using Synthetic and DBLP data set respectively. Efficiency studies of the serial and the parallel Gibbs sampling algorithms show that the learning algorithms scale linearly in the size of the data sets, the complexity of the models, and the number of cores. As future work, we will model the temporal information as continuous variable and model the more general dynamics of graphs (e.g., node/edge removal).

References

[1] L. Dietz, S. Bickel, and T. Scheffer, "Unsupervised prediction of citation influences," in *Proc. of Intl. Conf. on Machine Learning (ICML)*, 2007, pp. 233–240.

[2] L. Liu, J. Tang, J. Han, M. Jiang, and S. Yang, "Mining topic-level influence in heterogeneous networks," in *Proc. of ACM Conf. on Information and Knowledge Management (CIKM)*, 2010, pp. 199–208.

[3] C. Hu, H. Cao, and C. Ke, "Detecting influence relationships from graphs," in *Proc. of SIAM Intel. Conf. on Data Mining (SDM)*, 2014, pp. 821–829.

[4] L. Singh, "Exploring graph mining approaches for dynamic heterogeneous networks," in *National Science Foundation Symposium on Next Generation of Data Mining and Cyber-Enabled Discovery for Innovation*, October 2007.

[5] K. Lee and L. Liu, "Efficient data partitioning model for heterogeneous graphs in the cloud," in *Intl. Conf. for High Performance Computing, Networking, Storage and Analysis (SC)*, 2013, p. 46.

[6] S. Aral, L. Muchnik, and A. Sundararajan, "Distinguishing influence-based contagion from homophily-driven diffusion in dynamic networks," *Proceedings of the National Academy of Sciences*, vol. 106, no. 51, pp. 21 544–21 549, 2009.

[7] G. V. Steeg and A. Galstyan, "Statistical tests for contagion in observational social network studies," in *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2013, Scottsdale, AZ, USA, April 29 - May 1, 2013*, 2013, pp. 563–571.

[8] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

[9] Z. Hu, C. Wang, J. Yao, E. P. Xing, H. Yin, and B. Cui, "Community specific temporal topic discovery from social media," *CoRR*, vol. abs/1312.0860, 2013.

[10] G. Heinrich, "Parameter estimation for text analysis," Technical report, Tech. Rep., 2005.

[11] H. C. Chuan Hu, "Discovering time-evolving influence from dynamic heterogeneous graphs," Tech. Rep., 2015. [Online]. Available: http://www.cs.nmsu.edu/dbdm/TR/tr.ieee.bigdata.workshop.pdf

[12] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2007.

[13] L. Yao, D. M. Mimno, and A. McCallum, "Efficient methods for topic model inference on streaming document collections," in *Proc. of the 15th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining, 2009*, 2009, pp. 937–946.

[14] I. Porteous, D. Newman, A. T. Ihler, A. U. Asuncion, P. Smyth, and M. Welling, "Fast collapsed gibbs sampling for latent dirichlet allocation," in *Proc. of the 14th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining, 2008*, 2008, pp. 569–577.

[15] A. Q. Li, A. Ahmed, S. Ravi, and A. J. Smola, "Reducing the sampling complexity of topic models," in *The 20th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining, KDD 2015*, 2014, pp. 891–900.

[16] J. Yuan, F. Gao, Q. Ho, W. Dai, J. Wei, X. Zheng, E. P. Xing, T. Liu, and W. Ma, "Lightlda: Big topic models on modest computer clusters," in *Proc. of the 24th Intl. Conf. on World Wide Web, WWW 2015*, 2015, pp. 1351–1361.

[17] Z. Liu, Y. Zhang, E. Y. Chang, and M. Sun, "PLDA+: parallel latent dirichlet allocation with data placement and pipeline processing," *ACM TIST*, vol. 2, no. 3, p. 26, 2011.

[18] D. Newman, A. U. Asuncion, P. Smyth, and M. Welling, "Distributed algorithms for topic models," *Journal of Machine Learning Research*, vol. 10, pp. 1801–1828, 2009.

[19] D. Chakrabarti, Y. Zhan, and C. Faloutsos, "R-MAT: A recursive model for graph mining," in *Proc. of SIAM Intel. Conf. on Data Mining (SDM)*, 2004, pp. 442–446.

[20] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, "Arnetminer: extraction and mining of academic social networks," in *Proc. of the ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, 2008, pp. 990–998.

[21] S. Kataria, P. Mitra, and S. Bhatia, "Utilizing context in generative bayesian models for linked corpus," in *Proc. of the AAAI Conf. on Artificial Intelligence (AAAI)*, 2010.

[22] C. Tan, J. Tang, J. Sun, Q. Lin, and F. Wang, "Social action tracking via noise tolerant time-varying factor graphs," in *SIGKDD*, 2010, pp. 1049–1058.

[23] J. Zhang, B. Liu, J. Tang, T. Chen, and J. Li, "Social influence locality for modeling retweeting behaviors," in *Intl. Joint Conf. on Artificial Intelligence (IJCAI)*, 2013.

[24] T. L. Griffiths and M. Steyvers, "Finding scientific topics," *Proceedings of the National Academy of Sciences*, vol. 101, no. suppl 1, pp. 5228–5235, 2004.

[25] D. A. Patterson and J. L. Hennessy, *Computer Organization and Design, Fourth Edition, Fourth Edition: The Hardware/Software Interface (The Morgan Kaufmann Series in Computer Architecture and Design)*, 4th ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2008.

[26] S. Kataria, P. Mitra, C. Caragea, and C. L. Giles, "Context sensitive topic models for author influence in document networks," in *Intl. Joint Conf. on Artificial Intelligence (IJCAI)*, 2011, pp. 2274–2280.

[27] L. Wang, T. Lou, J. Tang, and J. E. Hopcroft, "Detecting community kernels in large social networks," in *Intl. Conf. on Data Mining*, 2011, pp. 784–793.

[28] J. Tang, T. Lou, and J. M. Kleinberg, "Inferring social ties across heterogenous networks," in *ACM Intl. Conf. on Web Search and Data Mining*, 2012, pp. 743–752.

[29] X. Wang and A. McCallum, "Topics over time: a non-markov continuous-time model of topical trends," in *Proc. of the ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, 2006, pp. 424–433.

[30] Y. Wang, E. Agichtein, and M. Benzi, "TM-LDA: efficient online modeling of latent topic transitions in social media." in *Proc. of the ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, 2012, pp. 123–131.

[31] R. A. Rossi, B. Gallagher, J. Neville, and K. Henderson, "Modeling dynamic behavior in large evolving graphs." in *ACM Intl. Conf. on Web Search and Data Mining*, 2013, pp. 667–676.

[32] T. Iwata, A. Shah, and Z. Ghahramani, "Discovering latent influence in online social activities via shared cascade poisson processes," in *Proc. of the ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, 2013, pp. 266–274.

[33] A. Goyal, F. Bonchi, and L. V. S. Lakshmanan, "Learning influence probabilities in social networks," in *Proc. of the Int. Conf. on Web Search and Web Data Mining, WSDM 2010*, 2010, pp. 241–250.

[34] C. Wang, J. Han, Y. Jia, J. Tang, D. Zhang, Y. Yu, and J. Guo, "Mining advisor-advisee relationships from research publication networks," in *Proc. of the ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, 2010, pp. 203–212.