Community Detection in Weighted Temporal Text Networks

Yuting Jia, Ying Huang, Yaowei Huang, Bo Wang, Xinbing Wang
Shanghai Jiao Tong university
{hnxxjyt,hy941001,14330222150355,wb1996,xwang8}@sjtu.edu.cn

ABSTRACT

The community detection in real-life text networks, the majority of which are weighted temporal text networks, is confronted with two main problems – how to model the weight of edges and how to exploit the temporal information.

In this paper, we offer a new perspective to encode the edge weight and temporal information. A probabilistic generative model, named *Custom Temporal Community Detection* (CTCD) is introduced, which views the link between two nodes as a weighted edge with several time stamps. Our model utilizes network, semantic and temporal information simultaneously to extract temporal community affiliations for individual user, influence strength across communities and temporal interested topic in each community. An efficient inference method and corresponding parallel implementation are proposed to scale to large datasets.

Through the knowledge extracted by CTCD, we are able to spot the community shift of the individual user and track the development of the communities over time. Moreover, experiments on two large-scale temporal text networks show that CTCD achieves significant improvement over state-of-the-art methods on the task of user behavior prediction.

CCS CONCEPTS

•Information systems →Clustering; Social networks; •Computing methodologies →Topic modeling; •Mathematics of computing →*Bayesian networks;* **•Networks →***Social media networks;*

KEYWORDS

temporal community detection, topic model, data mining, social network

ACM Reference format:
Yuting Jia, Ying Huang, Yaowei Huang, Bo Wang, Xinbing Wang. 2016. Community Detection in Weighted Temporal Text Networks. In *Proceedings of ACM Conference, Washington, DC, USA, July 2017 (Conference'17),* 9 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

From blogs to video-sharing sites to social networks and still others, online social media have experienced rapid growth over the past half-decade [17]. To extract meaningful knowledge from the available large amounts of data, the network structure inside it proves to be of utmost importance [12]. One way to understand network is to identify groups of nodes which share same properties or functions, which is known as *community detection* [27]. Detecting network communities allows us to discover functionally related objects [13, 14, 25], infer missing attribute values [1, 8], and predict unobserved connections [5].

Most of the real-life networks we are interested in are weighted temporal text networks, such as online social media (e.g. Twitter) and the academic citation/coauthor network (e.g. DBLP digital library), as shown in 1(a). The nodes, which in real-life network often represent users, are associated with several posts. These posts can be tweets in Twitter, papers published in conference and journal etc. The edge between users is associated with a weight. Naturally, each post is designated with a time stamp which indicates the publish time. In addition, each link is also characterized by several time stamps illustrating the time of interactions, which is detailedly formulated in §3.1.

However, it is difficult to comprehensively model such networks. In terms of edge weight, for simplification, existing methods often treat edges equally [18, 22, 25]. And in the few works which take it into consideration, the edge weight is only used to calculate and maximize/minimize some measure of graph, by which the community can be detected [6, 10, 21]. However, the edge weight is an important feature of the network and supposed to be explicitly involved into the model. Therefore, we intend to introduce the edge weight into the generative model.

Another issue is how to utilize the temporal information. A large amount of works divide the period of time into slices artificially and focus on the subnetwork in each time slice [7, 10, 19]. However, time is intrinsically continuous so that the determination of the slice size is tricky. Instead of discretizing time, we employ continuous distribution to model the time attribute of both posts and links.

In order to tackle the problems mentioned above within a unified framework, we propose a probabilistic generative model based on Bayesian network, named *Custom Temporal Community Detection* (**CTCD**), which models the edge weight, post time stamps and interaction time stamps directly. Taking both weight information and temporal information in to account, CTCD is able to extract temporal community affiliations for each user, the influence strength across communities and temporal interested topics in each community at the same time. The expected output of CTCD is demonstrated in 1(b). An efficient Gibbs-sampling-based inference algorithm is designed and corresponding parallel implementation for adapting to large-scale social networks is introduced. Besides, our approach is proved to achieve high accuracy for prediction task. And the extracted knowledge reveals interesting developing pattern of communities in the network.

To summarize, our contributions are as follows:

1. Novel Perspective. We propose a new idea that model multiple interactions between two users as a weighted edge with two bags of time stamps, which are generated within a topic model

Conference'17, Washington, DC, USA

DOI: 10.1145/nnnnnnn.nnnnnnn

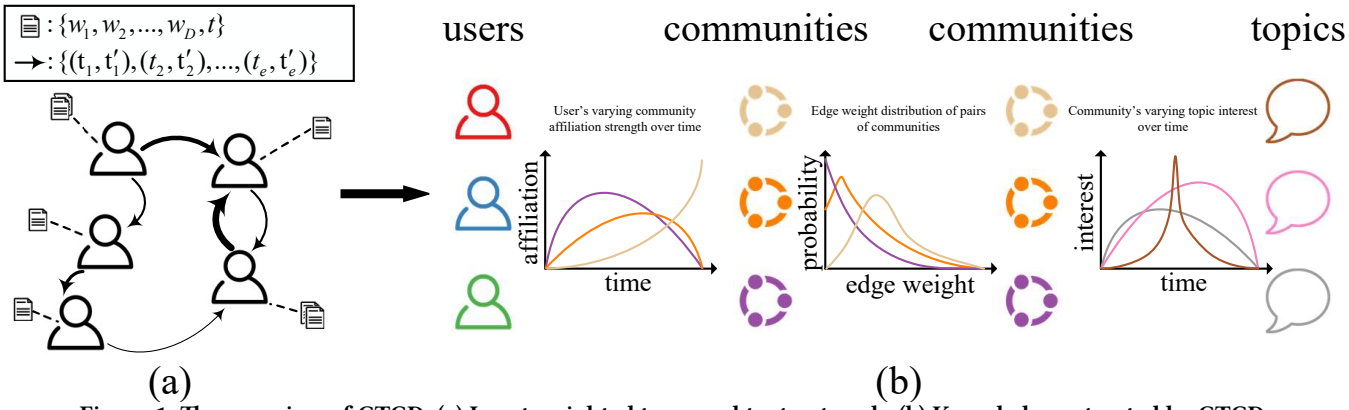


Figure 1: The overview of CTCD: (a) Input weighted temporal text network. (b) Knowledge extracted by CTCD.

framework. By modeling time in this way, our model is able to extract temporal community affiliation strength for each user, but not static community affiliation strength in nearly all existing models.

2. Comprehensive Model. Our model incorporates all semantic, network and temporal information. Temporal community affiliations for each user, the influence strength across communities and temporal interested topics can be uncovered at the same time. With the knowledge extracted, we are able to spot the community shift of the individual user and track the development of the communities over time.

3. Scalable Inference. We propose an inference method using Gibbs sampling which reaches linear complexity. Then to adapt to large-scale social dataset, we develop the parallel implementation and propose several methods to deal with the synchronization problem between processes.

The rest of this paper is organized as follows: §2 summarizes the related literature; §3 formulates the problem and introduces the model. §4 describes the inference method and the parallel implementation. §5 develops some user behavior prediction approaches. §6 evaluates the model proposed. And finally, we conclude this work in §7.

2 RELATED WORK

2.1 Weighted Network

Most of real-life networks are weighted network. For example, in social networks, the user enjoys different degree of intimacy of relations with friends in his contact list and therefore the link is expected to be characterized with various weight. However, it is difficult to model such weight of the edge. To simplify the analysis or design, they are often formulated as two-value networks (1 denotes edge existence, otherwise 0) in previous community detection works [18, 22, 25], which leads to loss of information. Recently, there has been some researches on community detection in weighted networks [6, 15, 21]. But all these methods are based on local property of network like cut, node-strength or local update like label propagation. We introduce a new method which considers both edge weight and global property of network in CTCD. By the empirical result of comparison between CTCD and corresponding unweighted version model, CTCD-Ber, we can get a view of the improvement brought by taking edge weight into account.

2.2 Temporal Community Detection

Temporal community detection is an important branch of community detection research [2, 4, 10], which aims to identify the process that communities emerge, grow, combine, and decay over time. Most existing works separate the whole network into several subnetworks by splitting time into slices, then exploit some statical community detection method on each subnetwork and finally merge sub-results with some smoothness [7, 10, 19]. However, discretization of time always begs the question of selecting the slice size, and inevitably the size is too small for some regions and too large for others.

In this paper, we encode the time stamps of interactions between users as the attribute of edges and model them in a generative and continuous manner.

3 CUSTOM TEMPORAL COMMUNITY DETECTION METHOD

3.1 Problem Formulation

Notation. In this paper, all vectors are column vectors and denoted by lower-case bold letters like $\boldsymbol{\theta}$ and $\boldsymbol{\pi}$. Calligraphic letters are used to represent sets like $\mathcal{U}$ and $\mathcal{V}$. And for simplicity, the corresponding normal letters are used to denote their size like $V = |\mathcal{V}|$. The details of notations in this paper are listed in table 1.

Definition 3.1. (**Weighted Temporal Network**) Consider a real-life network $\mathcal{G} = (\mathcal{U}, \mathcal{E})$ where $\mathcal{U}$ is a set of U users and $\mathcal{E}$ is a set of E weighted edges.

A directed edge $(i, i', e_{ii'}, \boldsymbol{s}, \boldsymbol{s'}) \in \mathcal{E}$ means the existence of interaction(s) between user i and i', the weight of the interaction(s) is $e_{ii'}$, and this edge contains two bags of time stamps, $\boldsymbol{s}$ and $\boldsymbol{s'}$, which represent the **out** and **in** time stamps of each interaction between user i and i' respectively. In this paper, we set the weight as the number of interactions (i.e. the number of pairs of time stamps of interactions) between two users. Correspondingly, if there is no any interaction between user i and i', $e_{ii'}$ equals to zero and these two bags of time stamps are both empty.

Note that interaction represents different connecting activities in different networks. One interaction means one retweeting, coauthoring or citing behavior. Between two users there may be several interactions. For each interaction, there will be two time stamps,

Table 1: Notations used in this paper

Symbol	Description
U, C, K, V	number of users, communities, and topics and size of vocabulary
D, W, E, S	number of posts, words, edges, and interactions(pairs of time stamps)
D_i, E_i	number of posts by user i, and out edges from user i
$W_{ij}, S_{ii'}$	number of words in post j by user i, and interactions between user i and i'
d_{ij}	the jth post of user i
$e_{ii'}$	the weight of the edge between user i and i'
t_{ij}	the time stamp of post d_{ij}
$s_{ii'm}, s'_{ii'm}$	the 'out' and 'in' time stamp of mth interaction in edge $e_{ii'}$
w_{ijl}	the lth word in post d_{ij}
c_{ij}, z_{ij}	community and topic associated with post d_{ij}
$g_{ii'}, g'_{ii'}$	communities associated with user i and i' in edge $e_{ii'}$
$\boldsymbol{\pi}_i$	multinomial distribution over communities specific to user i
$\boldsymbol{\theta}_c$	multinomial distribution over topics specific to community c
$\boldsymbol{\phi}_k$	multinomial distribution over words specific to topic k
ψ_{ck}	Beta distribution over time specific to community c and topic k
$\eta_{cc'}$	Poisson distribution over edge weight from community c to c'
δ_{ig}	Beta distribution over **out** time specific to user i and community g
$\gamma_{i'g'}$	Beta distribution over **in** time specific to user i' and community g'

out and in, and they may be different. For example, when investigating the citing interaction as shown in Figure 2, we define the direction of the edge from the cited user to the citing user. The out time stamp, s, denotes the publish time of the cited paper and, correspondingly, the in time stamp, s', represents the publish time of the citing paper. This setting is based on the fact that in citing interactions, it is user i at time s who influences user i' at time s'. So the time stamp of this interaction should not be only s'. However, in some other kinds of interaction activities like coauthoring, the two time stamps of one interaction are supposed be the same, which are both the publish time of the collaborated paper.

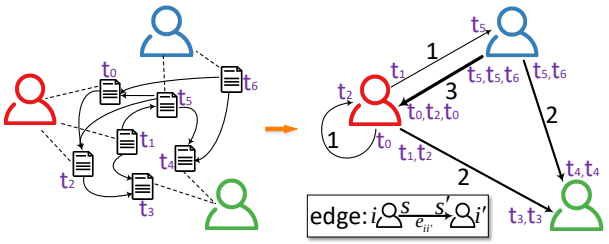


Figure 2: The example of citation interaction modeling

Definition 3.2. (**Semantic Information**) Each user is associated with D_i posts, denoted as $\mathcal{D}_i$. And each post d_{ij} is a bag of words along with a time stamp t_{ij}.

In text network, the available post content information enables us to extract semantic topics as well as detecting communities. Thus, a community can be characterized not only by link structures, but also by its topic composition. Associating each community with a distribution over topics makes the communities more interpretable. Therefor we define communities as follows:

Definition 3.3. (**Community**) A community $c \in C$ has two components: (1) a multinomial distribution over topics, $\boldsymbol{\theta}_c$, where each θ_{ck} represents the probability that community c publishes posts covering topic k; (2) $\boldsymbol{\eta}_c$, where each $\eta_{cc'}$, containing two parameters $\eta_{cc'0}$ and $\eta_{cc'1}$, represents a Poisson distribution of the weight of edges from community c to community c'.

In the real life networks, users generally enjoy multiple roles and get involved in different communities [24]. Therefore a *mixed-membership* approach is adopted. Furthermore, the user's membership of communities is not invariable. The participation and influence of one user in a specific community constantly changes, and a distribution over time is assigned to each community user pair to depict the dynamic feature of the the membership. for each user i, (1) there is a multinomial distribution over communities, $\boldsymbol{\pi}_i$, where each π_{ic} represents the affiliation strength from user i to community c; (2) for each community c, there are two Beta distributions over time, δ_{ic} and γ_{ic}, which model the change of user i's influence degree and participation degree over time in community c.

Definition 3.4. (**Topic**) A topic $k \in \mathcal{K}$ is a multinomial distribution over a collection of words, $\boldsymbol{\phi}_k$, where each ϕ_{kw} represents the probability for word w generated from topic k.

Definition 3.5. (**Community-Topic Temporal Distribution**) For each community c and topic k, the popularity of topic k in community c obey a Beta distribution over time, ψ_{ck}.

3.2 Model Structure

To integrate the text, time and network information into a unified framework, we propose the probabilistic generative model, CTCD (Custom Temporal Community Detection). The graphical model representation of CTCD is shown in 3. Though the model seems complicated on the whole, it consists of three simple component in essential : the user affiliation component models users' affiliation to communities which act as bridge connecting the other two components, the text-time component discovers the topics and acquires variation of topic popularity within communities, the network-time component considers the link structure and spots the membership change of users.

3.2.1 User affiliation component. Generally, users belong to multiple communities in the real world. We associate each user i with a probability vector $\boldsymbol{\pi}_i$, which denotes the probability distribution of user i's affiliation to communities. For each post $d_{ij} \in \mathcal{D}$, we assign it to a community c_{ij}, which indicates the affiliation of user i when he publishes his jth post. And for each edge $e_{ii'} \in \mathcal{E}$, we associate it with two communities $g_{ii'}$ and $g'_{ii'}$, which respectively represents the affiliations of user i and i' when the interaction happens.

3.2.2 Text-time component. Each post d_{ij} contains a bag of words $\{w_{ij1}, w_{ij2}, ..., w_{ij|d_{ij}|}\}$, where $|d_{ij}|$ represents the length of post d_{ij}. Traditional topic models assume that a document is a mixture of topics and each words in the document is generated from a hidden topic. It's necessary to view the document as a topic mixture when the it is long and possible to exhibit several different topics. However, when the posts are short texts, such as tweets, micro-blogs and titles of papers, they are usually about a single topic and assign the posts with one topic is enough.

Besides a bag of words, each post d_{ij} also has a time stamp t_{ij}. To model this temporal information, we employ a Beta distribution ψ_{ck} over time to model the temporal variation specific to each

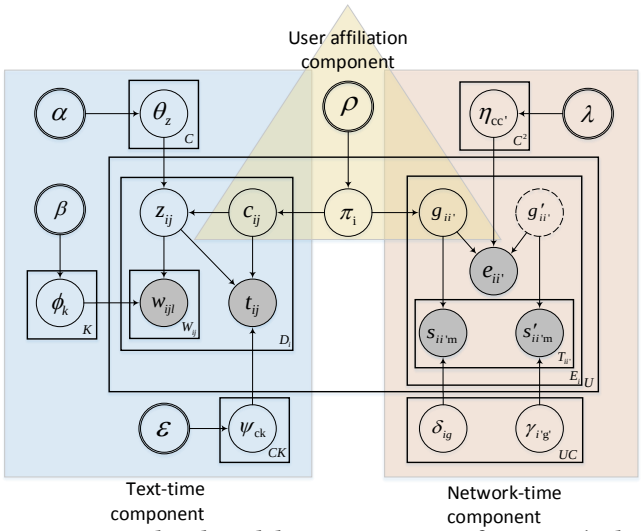


Figure 3: Graphical Model Representation of CTCD. The latent variable g' is represented by dashed circle because it is drawn from π_i, which is not shown in graph.

community c and each topic k. Thus, the publishing time of post d_{ij}, t_{ij}, is sampled from Beta distribution $\psi_{c_{ij}z_{ij}}$.

3.2.3 Network-time component. Each edge (i, i') contains the weight $e_{ii'}$ and two bags of time stamps, $\boldsymbol{s}_{ii'}$ and $\boldsymbol{s'}_{ii'}$, which respectively represent the out and in time stamps of interactions between user i and i'. We use pairwise community Poisson distribution $\boldsymbol{\eta}$ to model the weight of edges between pairs of users. For each edge (i, i'): (1) the weight $e_{ii'}$ is drawn from $\eta_{g_{ii'}g'_{ii'}}$ which represents the influence strength between community $g_{ii'}$ and $g'_{ii'}$; (2) each out time stamp $s_{ii'm}$ is sampled from Beta distribution δ_{ig}; (3) and each in time stamp $s'_{ii'm}$ is sampled from Beta distribution $\gamma_{i'g'}$.

Social networks are usually sparse, so we only model positive edges for speedup: the variables $g_{ii'}$ and $g'_{ii'}$ exist if and only if $(i, i') \in \mathcal{E}$, i.e., $e_{ii'} > 0$. As in [17], the negative edges can be avoided by a Bayesian fashion: we use a Gamma prior(λ_0, λ_1) on each $\eta_{cc'}$, and set $\lambda_0 = 0.1$ and $\lambda_1 = K \cdot ln(N_{neg}/C^2)$, where $N_{neg} = U(U-1) - |\mathcal{E}|$ represents the number of negative edges and K represents a tunable weight. By the Bayesian fashion, the time complexity is reduced to be linear with the number of edges.

3.3 Generative Process

The whole generative process is summarized in Algorithm 1. For user i, when he publishes a post d_{ij}, first he selects a community c_{ij} by his community distribution $\boldsymbol{\pi}_i$, and then selects a topic z_{ij} by topic interest $\boldsymbol{\theta}_{c_{ij}}$ specific to community c_{ij}. With the selected community c_{ij} and topic z_{ij}, words are generated from the topic's word distribution $\boldsymbol{\phi}_{z_{ij}}$, and the time stamp is generated from temporal distribution $\psi_{c_{ij}z_{ij}}$ specific to c_{ij} and z_{ij}. On the other hand, when he interacts with another user i', first they select communities $g_{ii'}$ and $g'_{ii'}$ respectively, and then the weight of edge (i, i'), $e_{ii'}$, is drawn from community-level influence strength distribution $\eta_{g_{ii'}g'_{ii'}}$. With the selected communities, each out time stamp $s_{ii'm}$ is sampled from temporal influence strength $\delta_{ig_{ii'}}$ specific to user i and community $g_{ii'}$, and each in time stamp $s'_{ii'm}$ is sampled from temporal participation strength $\gamma_{i'g_{ii'}}$ specific to user i' and community $g'_{ii'}$.

Algorithm 1 Generative Process for CTCD

1. For each community c=1,2,...,C,
   (1) Sample the distribution over topics, $\theta_c|\alpha \sim Dir(\alpha)$.
   (2) For each community c'=1,2,...,C,
      (a) Sample the probability of community-community link, $\eta_{cc'}|\lambda \sim Gamma(\lambda_0, \lambda_1)$.
   (3) For each topic k=1,2,...,K,
      (a) Sample the distribution over time stamps of document, $\psi_{ck}|\epsilon \sim Beta(\epsilon_0, \epsilon_1)$.
2. For each topic k=1,2,...,K,
   (1) Sample the distribution over words, $\phi_k|\beta \sim Dir(\beta)$.
3. For each user i=1,2,...,U,
   (1) Sample the distribution over communities, $\pi_i|\rho \sim Dir(\rho)$.
   (2) For each document j=1,2,...,
      (a) Sample community indicator, $c_{ij}|\pi_i \sim Mul(\pi_i)$.
      (b) Sample topic indicator, $z_{ij}|\theta_{c_{ij}} \sim Mul(\theta_{c_{ij}})$.
      (c) Sample time stamp, $t_{ij}|\psi_{c_{ij}z_{ij}} \sim Beta(\psi_{c_{ij}z_{ij}0}, \psi_{c_{ij}z_{ij}1})$.
      (d) For each word l=1,2,...,
         (i) Sample word, $w_{ijl}|\phi_{z_{ij}} \sim Mul(\phi_{z_{ij}})$.
   (3) For each out edge (i, i'),
      (a) Sample community indicator, $g_{ii'}|\pi_i \sim Mul(\pi_i)$.
      (b) Sample community indicator, $g'_{ii'}|\pi_{i'} \sim Mul(\pi_{i'})$.
      (c) Sample edge weight, $e_{ii'}|\eta_{g_{ii'}g'_{ii'}} \sim Poi(\eta_{g_{ii'}g'_{ii'}})$.
      (d) For each pair of time stamp m=1,2,...,
         (i) Sample out time stamp, $s_{ii'm}|\delta_{ig_{ii'}} \sim Beta(\delta_{ig_{ii'}0}, \delta_{ig_{ii'}1})$.
         (ii) Sample in time stamp, $s'_{ii'm}|\gamma_{ig'_{ii'}} \sim Beta(\gamma_{ig'_{ii'}0}, \gamma_{ig'_{ii'}1})$.

4 INFERENCE & IMPLEMENTATION

4.1 Inference Method

Exact inference for CTCD is difficult because of intractable latent variables, so we adopt collapsed Gibbs sampling method [16] to achieve an approximate inference. Gibbs sampling is a Markov chain Monte Carlo (MCMC) algorithm for obtaining a sequence of observations which are approximated from a specified multivariate probability distribution, when direct sampling is difficult. As with other MCMC algorithms, Gibbs sampling generates a Markov chain of samples of latent variables, like c and z in CTCD, each of which is correlated with nearby samples. And then the samples can be used to infer the distribution of interests, i.e., $\{\boldsymbol{\pi}, \boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\eta}\}$ in CTCD, while Beta distributions $\{\boldsymbol{\psi}, \boldsymbol{\delta}, \boldsymbol{\gamma}\}$ are fitted in inference process.

The whole inference process is summarized as Algorithm 2. In the sampling part of each iteration of Gibbs sampler, for each post d_{ij}, sample both the corresponding community indicator c_{ij} and topic indicator z_{ij}. And for each edge (i, i'), sample the corresponding community indicators $g_{ii'}$ and $g'_{ii'}$. In the fitting part, fit the Beta distributions $\boldsymbol{\psi}$ according to the sampled $\boldsymbol{c}$ and $\boldsymbol{z}$, and fit the Beta distributions $\boldsymbol{\delta}$ and $\boldsymbol{\gamma}$ according to the sampled $\boldsymbol{g}$ and $\boldsymbol{g'}$. Due to the space limitation, we only show the sampling and fitting formulas here, omitting the details of derivation process.

4.1.1 Simpling c_{ij} for post d_{ij}.

$$P(c_{ij} = c|z_{ij} = k, t_{ij} = t, \boldsymbol{c}_{-ij}, \boldsymbol{g}, \boldsymbol{z}_{-ij}, \boldsymbol{t}_{-ij}, .) \propto \frac{n_i^{(c)} + \rho}{n_i^{(\cdot)} + C\rho} \frac{n_c^{(k)} + \alpha}{n_c^{(\cdot)} + K\alpha} \frac{t_{ij}^{\psi_{ck0}-1} \cdot (1-t_{ij})^{\psi_{ck1}-1}}{B(\psi_{ck0}, \psi_{ck1})} \quad (1)$$

where $n_i^{(c)}$ denotes the number of posts and edges (both in and out) of user i associated with community c; $n_c^{(k)}$ denotes the number

Algorithm 2 Inference for CTCD

initialize community and topic assignment randomly for all tokens
for $iter$ = 1 to N_{iter} **do**
  // sampling part start
  for i = 1 to U **do**
    for j = 1 to D_i **do**
      draw c_{ij} from $P(c_{ij}|z_{ij}=k, t_{ij}=t, \boldsymbol{c}_{-ij}, \boldsymbol{g}, \boldsymbol{z}_{-ij}, \boldsymbol{t}_{-ij}, .)$
      draw z_{ij} from $P(z_{ij}|c_{ij}=c, t_{ij}=t, \boldsymbol{c}_{-ij}, \boldsymbol{z}_{-ij}, \boldsymbol{t}_{-ij}, \boldsymbol{w}, .)$
    for i' = 1 to U **do**
      if $e_{ii'} > 0$ **then**
        draw $g_{ii'}$ and $g'_{ii'}$ from $P(g_{ii'}, g'_{ii'}|e_{ii'}, \boldsymbol{g}_{-ii'}, \boldsymbol{c}, \boldsymbol{e}, .)$
  // sampling part end
  // fitting part start
  for c = 1 to C **do**
    for k = 1 to K **do**
      update ψ_{ck} by Eq.(4)
  for i = 1 to U **do**
    for c = 1 to C **do**
      update δ_{ic} by Eq.(5)
      update γ_{ic} by Eq.(6)
  // fitting part end
compute the posterior estimates of $\boldsymbol{\pi}$, $\boldsymbol{\theta}$, $\boldsymbol{\phi}$ and $\boldsymbol{\eta}$

of posts associated with both community c and topic z; $B(x,y) = \int_0^1 t^{x-1}(1-t)^{y-1}dt$ denotes the beta function. The marginal counters are represented by dots, e.g., $n_i^{(\cdot)}$ denotes the number of posts and edges (both in and out) of user i associated with any community. All the counters are counted with the post d_{ij} excluded.

4.1.2 Sampling z_{ij} for post d_{ij}.

$$P(z_{ij}=k|c_{ij}=c, t_{ij}=t, \boldsymbol{c}_{-ij}, \boldsymbol{z}_{-ij}, \boldsymbol{t}_{-ij}, \boldsymbol{w}, .) \propto \frac{n_c^{(k)}+\alpha}{n_c^{(\cdot)}+K\alpha} \frac{t_{ij}^{\psi_{ck0}-1}\cdot(1-t_{ij})^{\psi_{ck1}-1}}{B(\psi_{ck0},\psi_{ck1})} \frac{\prod_{v=1}^{V}\prod_{q=0}^{n_{ij}^{(v)}}(n_k^{(v)}+q+\beta)}{\prod_{q=0}^{n_{ij}^{(\cdot)}}(n_k^{(\cdot)}+q+V\beta)} \quad (2)$$

where v denotes the vth word in vocabulary; V denotes the size of vocabulary; $n_{ij}^{(v)}$ denotes the number of words v in post d_{ij}; $n_k^{(v)}$ denotes the number of words v associated with topic k. The marginal counters are represented by dots and all the counters are counted with the post d_{ij} excluded.

4.1.3 Sampling $g_{ii'}$ and $g'_{ii'}$ for edge (i,i').

$$P(g_{ii'}=c, g'_{ii'}=c'|e_{ii'}, \boldsymbol{g}_{-ii'}, \boldsymbol{c}, \boldsymbol{e}, .) \propto \frac{n_i^{(c)}+\sigma}{n_i^{(\cdot)}+C\sigma}\frac{n_{i'}^{(c')}+\rho}{n_{i'}^{(\cdot)}+C\rho}\frac{(\lambda_1+n_{cc'})^{(\lambda_0+n^s_{cc'})}\prod_{m=0}^{e_{ii'}-1}(\lambda_0+n^s_{cc'}+m)}{(\lambda_1+n_{cc'}+1)^{(\lambda_0+n^s_{cc'}+e_{ii'})}} \prod_{s_{ii'm}}\frac{s_{ii'm}^{\delta_{ic0}-1}(1-s_{ii'm})^{\delta_{ic1}-1}}{B(\delta_{ic0},\delta_{ic1})}\prod_{s'_{ii'm}}\frac{s'^{\gamma_{i'c'0}-1}_{ii'm}(1-s'_{ii'm})^{\gamma_{i'c'1}-1}}{B(\gamma_{i'c'0},\gamma_{i'c'1})} \quad (3)$$

where $n_{cc'}$ and $n^s_{cc'}$ respectively denote the number of edges and interactions associated with community c and c' with edge (i,i') excluded.

4.1.4 Fitting ψ_{ck} for community c and topic k.

$$\psi_{ck0} = m_{ck}\left(\frac{m_{ck}(1-m_{ck})}{v_{ck}^2}-1\right); \psi_{ck1} = (1-m_{ck})\left(\frac{m_{ck}(1-m_{ck})}{v_{ck}^2}-1\right) \quad (4)$$

where m_{ck} and v_{ck}^2 denote the mean and variance of time stamps of posts associated with community c and topic v.

4.1.5 Fitting δ_{ic} and γ_{ic} for user i and community c.

$$\delta_{ic0} = m_{ic}\left(\frac{m_{ic}(1-m_{ic})}{v_{ic}^2}-1\right); \delta_{ic1} = (1-m_{ic})\left(\frac{m_{ic}(1-m_{ic})}{v_{ic}^2}-1\right) \quad (5)$$

where m_{ic} and v_{ic}^2 denote the mean and variance of **out** time stamps of user i associated with community c.

$$\gamma_{ic0} = m'_{ic}\left(\frac{m'_{ic}(1-m'_{ic})}{v'^2_{ic}}-1\right); \gamma_{ic1} = (1-m'_{ic})\left(\frac{m'_{ic}(1-m'_{ic})}{v'^2_{ic}}-1\right) \quad (6)$$

where m'_{ic} and v'^2_{ic} denote the mean and variance of **in** time stamps of user i associated with community c.

With the samples, then we can infer the distribution of interests, i.e., $\{\boldsymbol{\pi}, \boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\eta}\}$ in CTCD. For any single sample, we can estimate the unknown distributions as follows:

$$\hat{\pi}_{ic} = \frac{n_i^{(c)}+\rho}{n_i^{(\cdot)}+C\rho}; \quad \hat{\theta}_{ck} = \frac{n_c^{(k)}+\alpha}{n_c^{(\cdot)}+K\alpha}; \quad \hat{\phi}_{kw} = \frac{n_k^{(w)}+\beta}{n_k^{(\cdot)}+V\beta} \quad (7)$$

However, the distribution $\boldsymbol{\eta}$ cannot be estimated like the others. The distributions like $\boldsymbol{\pi}$ can be inferred as Eq.(7) because of the property of multinomial distribution (i.e. $\pi_{ic} = P(c_{i(D_i+1)}=c|\boldsymbol{c}, \boldsymbol{g}, i)$ where $c_{i(D_i+1)}$ denotes the community indicator for an unseen post of user i), while $\boldsymbol{\eta}$ is the Poisson distribution which can only be represented as the formula:

$$P(e|g_e=c, g'_e=c', \boldsymbol{g}, \boldsymbol{g}', \lambda) = \frac{(\lambda_1+n_{cc'})^{(\lambda_0+n^s_{cc'})}\prod_{m=0}^{e-1}(\lambda_0+n^s_{cc'}+m)}{(\lambda_1+n_{cc'}+1)^{(\lambda_0+n^s_{cc'}+e)}} \quad (8)$$

where e is an unseen edge; g_e and g'_e are the two community indicators of edge e.

4.2 Time Complexity

Now we analyze the time complexity of our inference algorithm. It is shown that the time complexity of the whole inference algorithm is $O(W+S)$, which means that this algorithm scales linearly w.r.t. the size of dataset, i.e. the number of words and interactions. There are two parts of the whole algorithm, sampling and fitting, which will be analyzed separately.

Firstly we analyze the sampling part. To sample community indicator c_{ij}, Eq.(1) need to be calculated. Because all the counters can be counted in advance and updated in a constant time when each c_{ij} is resampled, with all the counters, Eq.(1) can be calculated in constant time. Sampling all $\boldsymbol{c}$ takes $O(D)$ time. But to sample topic indicator z_{ij}, when calculating Eq.(2), linear time in terms of the length of post d_{ij} costs. So sampling all $\boldsymbol{z}$ takes $O(W)$ time. Finally we sample community indicator $g_{ii'}$ and $g'_{ii'}$ according to Eq.(3). Calculating Eq.(3) costs $O(e_{ii'}+2S_{ii'})$ time, which can be also written as $O(S_{ii'})$. Because all the negative edges are modeled by a Bayesian fashion, the time for sampling all $\boldsymbol{g}$ and $\boldsymbol{g}'$ is reduced from quadratic ($O(U^2)$) to linear ($O(S)$). So the sampling part totally takes $O(W+S)$ time.

In the fitting part, counting all the counters takes $O(D+S)$. Updating ψ takes $O(CK)$ and updating δ and γ takes $O(UC)$. Considering that usually C and K won't be so large that $CK < D$ and $UC < S$, the fitting part totally takes $O(D+S)$ time.

4.3 Parallel Implementation

To apply CTCD to large-scale social data, parallelizing the inference algorithm is inevitable. In the fitting part, all fittings are totally independent which can be parallelized directly. But in the sampling part, after each iteration of sampling, the counters have to be updated and the process synchronization will be the bottleneck [9, 20]. Note that, as [23], we don't refit the Beta distributions after each resampling, but refit once at the end of each iteration. Now we propose three method to solve this synchronization problem.

1. Batch Sampling. At each iteration, first calculate all the counters, and in sampling, all the counters stay no-change. As the result, there is no critical section and synchronization problem because the counters are not updated. This method means that all the indicators are resampled simultaneously so that they will be updated based on same counters. But according to [3], the stochastic updating will converge much faster than batch updating. So the batch sampling may also take more iterations to convergence, which means more time for inference.

2. Exact Synchronization. This time we do not sample all indicators in batch again, but sample one, update counters once. In the synchronization, if we want to keep counters exactly correct, we have to: get lock, update counters and release locks. Although all processes share a common global lock is okay, requiring and releasing a common lock means single processing updating, which is easily to be the bottleneck. So we propose a parallel updating method for speedup. It should be noticed that all the indicators are relative to communities or topics. So we set one lock l_c (or l_k) for each community c and each topic k. If the community indicator of a post is changed from c to c', it will require lock l_c and l'_c and then update corresponding counters, and similar to topics.

3. Nearly-Exact Synchronization. There are three types of counters, specific to users, only specific to communities/topics and specific to words. (1) For counters specific to users like $n_i^{(c)}$, we can avoid the synchronization problem by parallelizing as user level, i.e., different processes sample indicators for different users. (2) For counters only specific to communities or topics like $n_c^{(k)}$, typically the value will be very large because of the low dimension of communities and topics. So the conflict is small enough to ignore. (3) For counters specific to words like $n_k^{(v)}$, the value will be small, but because of the large vocabulary, the probability for different processes to update indicators for a same word will be small enough to ignore. In a word, we can update the counter directly, without any lock, because the error is too small to affect the result much.

These three different parallel implementations are all practicable, and we implement all of them. The comparison of these three implementations are shown in §6.4

5 USER BEHAVIOR PREDICTION

5.1 User Interaction Prediction

User interaction prediction is defined to estimate the probability of an interaction. As in Def 3.1, an interaction can be once retweeting, coauthor, citing and so on. Thus, given two users i and i', and two time stamps s and s', the probability of an interaction from user i to user i' is computed as follow:

$$\begin{aligned} &P(i, i', s, s' | \boldsymbol{\pi}, \boldsymbol{\eta}, \boldsymbol{\delta}, \boldsymbol{\gamma}) \\ &= \sum_g \sum_{g'} P(g|i) P(g'|i') P(e_{ii'} > 0 | g, g') P(s|i, g) P(s'|i', g') \\ &= \sum_g \sum_{g'} \pi_{ig} \pi_{i'g'} [1 - (\frac{\lambda_1 + n_{cc'}}{\lambda_1 + n_{cc'} + 1})^{(\lambda_0 + n^s_{cc'})}] \\ &\qquad b(s; \delta_{ig0}, \delta_{ig1}) b(s'; \gamma_{i'g'0}, \gamma_{i'g'1}) \end{aligned} \tag{9}$$

where $b(x; \alpha, \beta)$ represents the probability density function of Beta distribution. In this formula, we use $P(e_{ii'} > 0)$ because if there exists the interaction, the weight of the edge from user i to i' will be at least 1, i.e., $e_{ii'} > 0$.

5.2 Post Time Prediction

Predicting the time stamps of posts [23] is to estimate the publishing time stamp of an unseen post. Given the words $\boldsymbol{w}$ and the author i of a post d, the time stamp can be chosen as the candidate giving maximum likelihood:

$$\begin{aligned} \hat{t}_d &= arg\max_t P(d|t, i, \boldsymbol{\pi}, \boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\psi}) \\ &= arg\max_t \sum_c P(c|i, \boldsymbol{\pi}) \sum_k P(k|c, \boldsymbol{\theta}) P(\boldsymbol{w}|k, \boldsymbol{\phi}) P(t|c, k, \boldsymbol{\psi}) \\ &= arg\max_t \sum_c \pi_{ic} \sum_k \theta_{ck} b(t; \psi_{ck0}, \psi_{ck1}) \prod_w \phi_{kw} \end{aligned} \tag{10}$$

Because the Beta distribution is a continuous curve, which is not proper for participating calculation. We can select multiple candidate time stamps from the whole time range, and use the discrete value of Beta distribution to calculate.

6 EXPERIMENTS

In this section, we first quantitatively evaluate the performance of CTCD for extracting communities and topics. We then qualitatively demonstrate the interesting knowledge about the development pattern of communities by some case studies. Finally we compare the efficiency and performance of different parallel methods.[1]

6.1 Experiment Setup

6.1.1 Datasets. Our experiments are conducted on two real temporal text networks, Academia and Weibo.

Academia. This dataset is extracted from data provided by website Acemap (http://acemap.sjtu.edu.cn), which has merged several data sources such as DBLP, Microsoft Academic Graph, ACM Digital Library and IEEEXplore. we select out 80271 authors who have published at least one paper on top journals or conferences in computer science region. The posts of each author are the titles of his papers. And each interaction between two users means once citation between them. Note that posts and interactions which time is earlier than Jan 1, 2000 are removed. Finally there are 2.4m posts 18m words, 10m edges and 22m interactions in the network and the vocabulary size is 0.1m.

Weibo. Weibo is one of the most popular micro-blog platform in China. We select 25088 active, interacted users and crawled micro-blogs they published in recent 5 years. In this dataset, each post represents one micro-blog and each interaction denotes once

[1]The code of CTCD and datasets used in the experiments can be downloaded at https://github.com/SamJia/KDD2017-Model-Dataset

retweeting. There are 21m posts, 125m words, 0.6m edges and 3.7m interactions in the network and the vocabulary size is 0.7m.

6.1.2 Baselines. We compare CTCD with another 4 baseline methods. Cluster Affiliation Model for Big Networks (BIGCLAM) [25], Poisson Mixed-Topic Link Model (PMTLM) [28], COmmunity Level Diffusion (COLD) [18] and Enhanced User-Temporal Model with Burst-weighted Smoothing (EUTB) [26]. BIGCLAM is one of representatives of community detection methods for pure network based on affiliation graph model. PMTLM utilizes network and posts while COLD and EUTB consider network, posts and temporal information.

In CTCD, the weight of edges are considered and modeled by Poisson distribution, now we propose a Bernoulli version CTCD (CTCD-Ber) for exhibiting the importance of considering edge weight. In CTCD-Ber, $\boldsymbol{\eta} \sim Beta(\lambda_0, \lambda_1)$ and $e_{ii'} \sim Bernoulli(\eta_{g_{ii'}g'_{ii'}})$, where $e_{ii'}$ is an indicator for edge (i, i') which can only be 0 or 1. In this situation, the sampling formula for $g_{ii'}$ and $g'_{ii'}$ changes to

$$P(g_{ii'} = c, g'_{ii'} = c' | e_{ii'}, \boldsymbol{g}_{-ii'}, \boldsymbol{c}, \boldsymbol{e}, .) \propto \frac{n_i^{(c)} + \sigma}{n_i^{(\cdot)} + C\sigma} \frac{n_{i'}^{(c')} + \rho}{n_{i'}^{(\cdot)} + C\rho} \frac{\lambda_0 + n_{cc'}}{\lambda_0 + \lambda_1 + n_{cc'}}$$
$$\prod_{s_{ii'm}} \frac{s_{ii'm}^{\delta_{ic0}-1}(1 - s_{ii'm})^{\delta_{ic1}-1}}{B(\delta_{ic0}, \delta_{ic1})} \prod_{s'_{ii'm}} \frac{s'^{\gamma_{i'c'0}-1}_{ii'm}(1 - s'_{ii'm})^{\gamma_{i'c'1}-1}}{B(\gamma_{i'c'0}, \gamma_{i'c'1})} \tag{11}$$

The Bernoulli distribution $\boldsymbol{\eta}$ can be inferred as

$$\hat{\eta}_{cc'} = \frac{\lambda_0 + n_{cc'}}{\lambda_0 + \lambda_1 + n_{cc'}} \tag{12}$$

And the probability of interaction from user i to i' changes to:

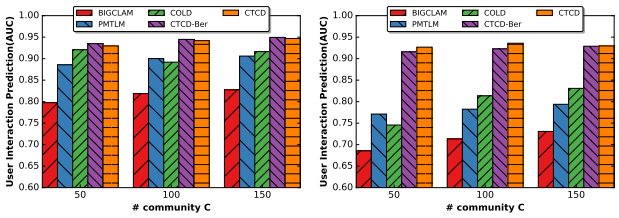
$$\begin{aligned} &P(i, i', s, s' | \boldsymbol{\pi}, \boldsymbol{\eta}, \boldsymbol{\delta}, \boldsymbol{\gamma}) \\ &= \sum_g \sum_{g'} P(g|i) P(g'|i') P(e_{ii'} = 1 | g, g') P(s|i, g) P(s'|i', g') \\ &= \sum_g \sum_{g'} \pi_{ig} \pi_{i'g'} \eta_{gg'} b(s; \delta_{ig0}, \delta_{ig1}) b(s'; \gamma_{i'g'0}, \gamma_{i'g'1}) \end{aligned} \tag{13}$$

6.2 Quantitative Evaluation

In this section, we quantitatively evaluate our model based on the user behavior prediction methods proposed in §5.

6.2.1 User interaction Prediction. Since here is no pre-defined threshold for user interaction prediction, we adopt *area under the receiver operating characteristic curve* (AUC) as the evaluation metric. When using normalized units, the AUC is equal to the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one (assuming 'positive' ranks higher than 'negative') [11]. In 5-fold cross validation, each time we randomly select 20% positive interactions and 1% negative interactions to calculate AUC.

Figure 4 shows the AUC value for five models. CTCD-Ber outperforms others in Academia dataset, while CTCD outperforms in Weibo dataset. There are 10m edges and 22m interactions in Academia, on the other hand, 0.6m edges and 3.7m interactions in Weibo. So the average weight of edges in Weibo (6.2) is much larger than Academia (2.2), which benefits the Poisson distribution and leads better performance for CTCD in Weibo dataset. Moreover, AUC of BIGCLAM is much lower than all other models, showing that pure network structure is not such reliable and incorporating



(a) Academia Dataset (b) Weibo Dataset

Figure 4: the AUC of User Interaction Prediction

information improves the accuracy of models significantly. Both CTCD and CTCD-Ber are much better than other models, which means the importance of considering temporal community.

Table 2: The Accuracy of Post Time Prediction

Dataset	Academia			Weibo		
# community C	50	100	150	50	100	150
EUTB	.529	.541	.552	.453	.432	.402
COLD	.496	.506	.502	.841	.839	.838
CTCD-Ber	**.624***	**.628***	**.629***	.863	.866	.864
CTCD	.621	.627	**.629***	**.875***	**.872***	**.873***

6.2.2 Post Time Prediction. Table 2 shows the prediction accuracy for the four models. For each number of communities, the same number of topics are set. And for calculating the accuracy, we set the time tolerance as 20%, i.e., the prediction is seen as correct if the difference between the predicted time and the real time is less than 20% of the whole time range. Note that here 5-fold cross validation is used.

From Table 2, we can know that: (1) CTCD-Ber and CTCD are both outperform in Academia dataset, while CTCD outperform in Weibo dataset because of the same reason mentioned in last section §6.2.1; (2) the average accuracy of Weibo dataset is much higher than Academia dataset. Because compared to Weibo where topics vary fast, the topics of Academia dataset is much stable so that the time of posts are much harder to predict; (3) the obvious improvement from COLD to CTCD shows that significant superiority of continuous distributions to model time.

6.3 Qualitative Evaluation

6.3.1 Capture the User Flow between Communities. After obtaining temporal community affiliations through CTCD, we are able to capture trend on the community level by analyzing the change of people's activeness in particular community. As mentioned before, the beta distribution δ model the affiliation from influence aspect and γ from the participation aspect. For the reason that the way modeling influence using citations is biased to old documents and therefore not comprehensive, we only utilize the affiliation strength from the perspective of participation.

One interesting phenomenon is that there are a certain amount of people who show high involvement in one community at first and then appear to be more active in another community. If the number of people who expose affiliation shift between two communities is large, it indicates that there is a user flow from one community

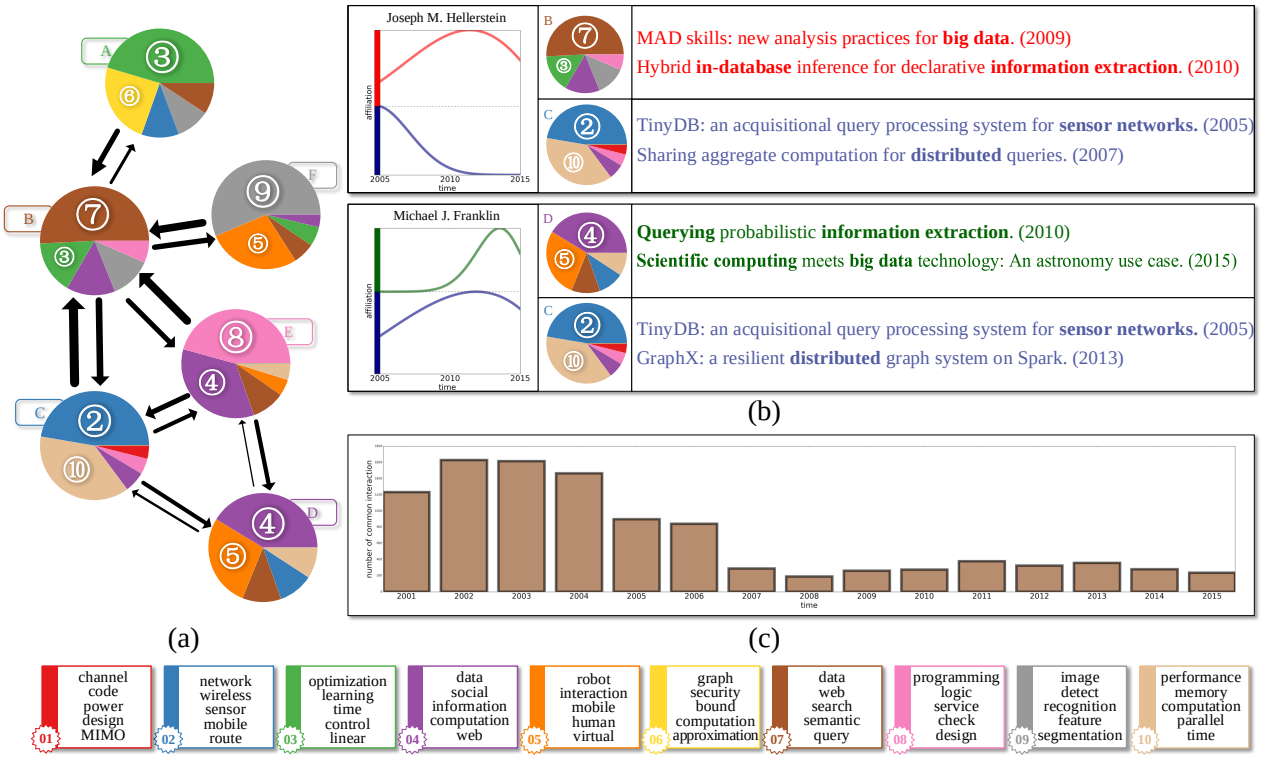
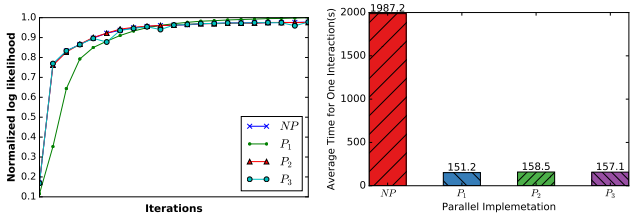


Figure 5: Dynamic pattern:(a) The user flow among communities. We represent the communities in the form of pie chart. The sectors of colors correspond to the topics with the same color stated at bottom. (b) Temporal affiliation strength. For each user, the top 2 dominant communities he belongs to are present as well as representative papers. (c) Change of common interactions over time. For certain year, the number of common interactions between two users indicates how many times their publications within that year reference or are cited by the same authors.

to another. We investigate such user flow on Academia dataset, setting both the topic number and community number ten.

Part of the user flow we discovered is displayed in Figure 5(a). Since the topic is more interpretable, we present the community in the form of combination of topics. According to the result, the community B whose interested topics cover data mining, optimization and information retrieval can be viewed as a rising community. Because the input flow of community B far outweighs the output flow and these people from community A C E F all show great tendency to join community B. Thus, we expect community B to be larger and popular. In addition, compared to community B, community C which mainly focuses on wireless sensor network and community E whose major interest is logic programming are losing heat. Such user flow diagram enable us to track the development of the community.

6.3.2 Analysis of Individual User. Zooming into individual user, the change of community affiliation strength over time can be investigated. In Figure 5(b), we pick up two users in the Academia dataset, plot affiliation strength of the top two dominant communities each user belongs to and list some representative papers of the two users. Joseph M. Hellerstein and Michael J. Franklin co-author several times and share high degree of affiliation to community C. Concluded from the affiliation strength plot, up to 2010, Joseph M. Hellerstein has almost left the community C while Michael J. Franklin is still in it. This finding is in accordance with the statistics as demonstrated in Figure 5(c) – the number of common interacted-authors of these two scholars fell from 1280 per year during 2001-2006 to 287 per year during 2007-2015.



(a) Log Likelihood (b) Iteration Time

Figure 6: Comparison of Different Parallel Implementation (*NP*: *No Parallel*; P_1: *Batch Sampling*; P_2: *Exact Synchronization*; P_3: *Nearly-Exact Synchronization*)

6.4 Parallel Implementations Comparison

In this section, we compare three different parallel implementations described in §4.3. The log-likelihood and iteration time are recorded on Academia dataset.

6.4.1 Log-Likelihood Comparison. To evaluate the sampling performance for different parallel implementations, we calculate the

log-likelihood for generating the whole training data in the inference iterations, as shown in Figure 6(a). Compared to other methods, P_1(*Batch Sampling*) converge slower, but the converged log-likelihood is higher. This result is in accordance with the features of batch update and stochastic update that batch update converge slower but can reach extreme point, while stochastic update converge faster but may never converge (keep oscillating around the extreme point). The trace of log-likelihood of P_2(*Exact Synchronization*) is exactly same to NP(*No Parallel*), because the synchronization is exactly done, which means the sampling process of P_2 is same to single process version. The log-likelihood of P_3(*Nearly-Exact Synchronization*) is similar to P_2, except few points where log-likelihood of P_3 is a little smaller. This difference should be due to the errors of approximate counter update.

6.4.2 Iteration Time Comparison. Figure 6(b) gives the averaged time cost per iteration of different parallel implementations. The three parallel implementations (i.e. P_1, P_2, P_3) are all take 16 cores. The time cost per iteration of different implementations are nearly same, and the speedup is greater than 12 compared to NP. P_1 is fastest because it need not update counters and the following one is P_3, which update counters without lock. Although P_1 is faster than the other two parallel implementations, because it take more iterations for convergence, it isn't the most efficient choice. Considering both efficiency and performance, we should adopt P_3 (or P_2) for first several iterations and then switch to P_1 for last several iterations.

7 CONCLUSION

In this paper, we study the problem of temporal community detection in temporal text network. We present CTCD (Custom Temporal Community Detection), a probabilistic generative model considering network, text and time, to extract temporal community affiliations for each user, the influence strength across communities and temporal topic interest in each community in a unified way. An inference method for CTCD based on Gibbs sampling is given. What's more, to adopt to large-scale dataset, we propose the scalable parallel implementation, in which 3 methods to deal with the synchronization problem is given and compared. The extracted knowledge of CTCD can be used to address various practical problem, such as user behavior prediction and user's community shift analysis. The result of experiments on two real-world large-scale temporal text network demonstrates the effectiveness of CTCD.

REFERENCES

[1] Ramnath Balasubramanyan and William W. Cohen. 2014. Block-LDA: Jointly Modeling Entity-Annotated Text and Entity-Entity Links. In *Handbook of Mixed Membership Models and Their Applications*. 255–273.

[2] Marya Bazzi, Mason A. Porter, Stacy Williams, Mark McDonald, Daniel J. Fenn, and Sam D. Howison. 2016. Community Detection in Temporal Multilayer Networks, with an Application to Correlation Networks. *Multiscale Modeling & Simulation* 14, 1 (2016), 1–41.

[3] Léon Bottou and Olivier Bousquet. 2008. The Tradeoffs of Large Scale Learning. In *Advances in Neural Information Processing Systems*. Vol. 20. 161–168.

[4] Deepayan Chakrabarti, Ravi Kumar, and Andrew Tomkins. 2006. Evolutionary Clustering. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '06)*. 554–560.

[5] Jonathan Chang and David M. Blei. 2010. Hierarchical Relational Models for Document Networks. *The Annals of Applied Statistics* 4, 1 (2010), 124–150.

[6] Duanbing Chen, Mingsheng Shang, Zehua Lv, and Yan Fu. 2010. Detecting overlapping communities of weighted networks via a local algorithm. *Physica A: Statistical Mechanics and its Applications* 389, 19 (2010), 4177 – 4187.

[7] Yudong Chen, Vikas Kawadia, and Rahul Urgaonkar. 2013. Detecting Overlapping Temporal Community Structure in Time-Evolving Networks. *CoRR* abs/1303.7226 (2013). http://arxiv.org/abs/1303.7226

[8] Michele Coscia, Giulio Rossetti, Fosca Giannotti, and Dino Pedreschi. 2012. DEMON: A Local-first Discovery Method for Overlapping Communities. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '12)*. 615–623.

[9] Padhraic Smyth David Newman and Mark Steyvers. 2006. Scalable Parallel Topic Models. *Journal of Intelligence Community Research and Development* (2006).

[10] N. Du, X. Jia, J. Gao, V. Gopalakrishnan, and A. Zhang. 2015. Tracking Temporal Community Strength in Dynamic Networks. *IEEE Transactions on Knowledge and Data Engineering* 27, 11 (Nov 2015), 3125–3137.

[11] Tom Fawcett. 2006. An Introduction to ROC Analysis. *Pattern Recogn. Lett.* 27, 8 (June 2006), 861–874.

[12] Santo Fortunato. 2010. Community detection in graphs. *Physics Reports* 486, 3fi?!5 (2010), 75 – 174.

[13] Mario Frank, Andreas P. Streich, David Basin, and Joachim M. Buhmann. 2012. Multi-assignment Clustering for Boolean Data. *J. Mach. Learn. Res.* 13, 1 (Feb. 2012), 459–489.

[14] M. Girvan and M. E. J. Newman. 2002. Community structure in social and biological networks. *PNAS* 99, 12 (2002), 7821–7826.

[15] S. Gregory. 2010. Finding overlapping communities in networks by label propagation. *New Journal of Physics* 12, 10 (Oct. 2010), 103018. arXiv:physics.soc-ph/0910.5516

[16] T. L. Griffiths and M. Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences* 101, Suppl. 1 (April 2004), 5228–5235.

[17] Qirong Ho, Rong Yan, Rajat Raina, and Eric P. Xing. 2012. Understanding the Interaction between Interests, Conversations and Friendships in Facebook. *CoRR* abs/1211.0028 (2012).

[18] Zhiting Hu, Junjie Yao, Bin Cui, and Eric Xing. 2015. Community Level Diffusion Extraction. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data (SIGMOD '15)*. 1555–1569.

[19] Vikas Kawadia and Sameet Sreenivasan. 2012. Sequential detection of temporal communities by estrangement confinement. *Scientific Reports* 2 (Nov. 2012). http://www.nature.com/srep/2012/121109/srep00794/full/srep00794.html

[20] Zhiyuan Liu, Yuzhou Zhang, Edward Y. Chang, and Maosong Sun. 2011. PLDA+: Parallel Latent Dirichlet Allocation with Data Placement and Pipeline Processing. *ACM Transactions on Intelligent Systems and Technology, special issue on Large Scale Machine Learning* (2011).

[21] Z. Lu, Y. Wen, and G. Cao. 2013. Community detection in weighted networks: Algorithms and applications. In *2013 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. 179–184.

[22] Gergely Palla, Imre Dernyi, Ills Farkas, and Tams Vicsek. 2005. Uncovering the overlapping community structure of complex networks in nature and society. *Nature* 435, 7043 (June 2005), 814–818.

[23] Xuerui Wang and Andrew McCallum. 2006. Topics over Time: A non-Markov Continuous-time Model of Topical Trends. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '06)*. 424–433.

[24] Jierui Xie, Stephen Kelley, and Boleslaw K. Szymanski. 2011. Overlapping Community Detection in Networks: the State of the Art and Comparative Study. *CoRR* abs/1110.5813 (2011).

[25] Jaewon Yang and Jure Leskovec. 2013. Overlapping Community Detection at Scale: A Nonnegative Matrix Factorization Approach. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining (WSDM '13)*. 587–596.

[26] H. Yin, B. Cui, H. Lu, Y. Huang, and J. Yao. 2013. A unified model for stable and temporal topic detection from social media data. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*. 661–672.

[27] Hongyi Zhang, Irwin King, and Michael Lyu. 2015. Incorporating Implicit Link Preference Into Overlapping Community Detection. (2015).

[28] Yaojia Zhu, Xiaoran Yan, Lise Getoor, and Cristopher Moore. 2013. Scalable Text and Link Analysis with Mixed-topic Link Models. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '13)*. 473–481.