Frameworks to Encode User Preferences for Inferring Topic-sensitive Information Networks

Qingbo Hu* Sihong Xie* Shuyang Lin* Wei Fan† Philip S. Yu*‡

Abstract

The connection between online users is the key to the success of many important applications, such as viral marketing. In reality, we often easily observe the time when each user in the network receives a message, yet the users' connections that empower the message diffusion remain hidden. Therefore, given the traces of disseminated messages, recent research has extensively studied approaches to uncover the underlying diffusion network. Since topic related information could assist the network inference, previous methods incorporated either users' preferences over topics or the topic distributions of cascading messages. However, methods combining both of them may lead to more accurate results, because they consider a more comprehensive range of available information. In this paper, we investigate this possibility by exploring two principled methods: ***Weighted Topic Cascade*** (WTC) and ***Preference-enhanced Topic Cascade*** (PTC). WTC and PTC formulate the network inference task as non-smooth convex optimization problems and adopt coordinate proximal gradient descent to solve them. Based on synthetic and real datasets, substantial experiments demonstrate that although WTC is better than several previous approaches in most cases, it is less stable than PTC, which constantly outperforms other baselines with an improvement of 4%$\sim$10% in terms of the F-measure of inferred networks.

1 Introduction

Recently, "ALS Ice Bucket Challenge", which is an activity to promote the awareness of amyotrophic lateral sclerosis (ALS), went virally through social media. Because of the participation of numerous political figures and other celebrities, 1.2 million related videos were shared by Facebook users between June 1 and August 13, and Twitter users mentioned this activity 2.2 million times between July 29 and August 17[1]. Similar to the videos and tweets of "ALS Ice Bucket Challenge", we refer a piece of information that spreads through social networks as a ***contagion*** or ***cascade***.

*Department of Computer Science, University of Illinois at Chicago, Chicago, U.S.A. {qhu5, sxie6, slin38, psyu} @uic.edu

†Baidu Research Big Data Lab, Sunnyvale, U.S.A. fanwei03@baidu.com

‡Institute for Data Science, Tsinghua University, Beijing, China

[1]http://en.wikipedia.org/wiki/Ice_Bucket_Challenge

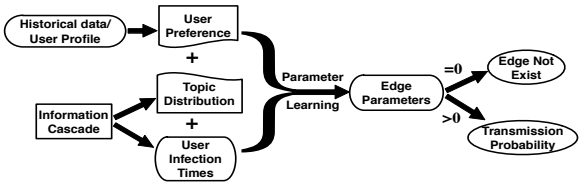


Figure 1: A sketch of proposed frameworks

1.1 Network Inference Problem.

Normally, it is easy for us to record the trace of a cascade: the timestamps when users share/post the message (also called ***infected***). On the contrary, the underlying network structure that carries the diffusion of a co-anchorntagion is usually hidden. An online blog user, for instance, seldom specifies his/her information source when s/he posts a blog. However, the embedded diffusion network is critical to many real applications, such as viral marketing [11]. As a result, inferring the latent diffusion network from the observed cascades, a.k.a. ***network inference problem***, becomes an important and prevailing task related to information networks.

1.2 Related Work.

Recently, the network inference problem has been extensively studied by numerous researchers. In 2010, based on the basic assumptions of *Independent Cascade* model [11], Rodriguez et al. proposed ***NetInf*** [7], initially utilizing ***survival analysis*** to solve the network inference problem. The method iteratively added "most likely" edges among people to unveil the hidden structure of the diffusion network. Similar idea was also used in [16], with an additional constraint that the diffusion networks should comply with tree structures. However, the inferred edges in both these models were presumed to be identical, meaning that they did not differentiate the strengths of different edges (a.k.a. the probabilities to infect the connected users). To compensate this disadvantage, [6] systematically studied the network inference problem under three distinct parametric models. The proposed ***NetRate*** method was able to output not only the structures of diffusion networks but also the strength on each edge. NetRate formulated the network inference problem as a convex optimization problem and utilized ***Stochastic Gradient Descent*** (SGD) method to effi-

ciently solve it. Afterwards, various approaches were proposed to either improve the effectiveness or the efficiency of the NetRate [4, 8, 19, 20], while other non-survival analysis based methods to solve similar problems include [1, 14, 17].

In reality, topic related data that can be used to assist the inference are often available. On the one hand, a user's preference over different topics is usually retrievable from his/her profile or historical data. For instance, in an online social network, a user's preference can be explicitly extracted from the interested topics s/he has marked or implicitly approximated by the topic distributions of the contagions that previously infected him/her. To utilize the preferences of users, ***MoNet*** [19] added an extra weighting term to NetRate model, which penalized the edges that connected users with very different preferences. On the other hand, the topic distribution of a cascade can also be extracted from the cascade's textual information through methods such as ***Latent Dirichlet Allocation*** (LDA) [2]. ***Topic Cascade*** [4] model employs this type of information to infer topic-sensitive diffusion networks. In other words, the strength of inferred edges by Topic Cascade could change with respect to different topics. This might better fit real networks: a user may have a significant impact on one of his/her friends due to his/her authority related to one topic, but it does not mean s/he is an expert in all domains. Experiments showed that MoNet and Topic Cascade could infer diffusion networks more accurately comparing to NetRate.

1.3 Motivations and Proposed Methods. Although MoNet and Topic Cascade utilize the preferences of users and the topic distributions of cascades, respectively, neither of them is capable of incorporating both types of information. Intuitively, it is possible to better infer the network by considering a more comprehensive range of available information, which is exactly the motivation of this paper.

Neglecting either users' preferences or the topic distributions of cascades might cause potential problems. On the one hand, without utilizing the preferences of users, if we observe that a cascade belongs to a topic and has two sequentially infected users, Topic Cascade might obtain an edge between them with respect to the topic. However, the user that gets infected later might be actually influenced by a third user, who has similar preference over topics. Here we accept the presumption that users with similar preferences have a higher chance to influence each other, which is confirmed by MoNet [19]. On the other hand, without incorporating the topic distributions of cascades, the strength of an inferred connection by MoNet is uniform across all topics. In other words, there is no topic distribution over the inferred edges. However, the fact that topic-sensitive networks better mimic real diffusion networks is shown by Topic Cascade [4].

Furthermore, how to judiciously synthesize the user preferences and the topic distribution of cascades remains unexplored. In fact, it is usually much harder to infer a topic-sensitive network, since it has more parameters and the model's additional regularizers require more time-consuming algorithms to get a solution. Therefore, although numerous possible combination methods could be used to incorporate both types of information, incorporating them without increasing a model's complexity is still challenging. Inspired by MoNet, we first introduce ***Weighted Topic Cascade*** (WTC), which is a framework that directly transforms user preference vectors into a weighting term and adds it to Topic Cascade. However, comparing to the improvement that MoNet brings to NetRate, the experiments show that WTC does not significantly prevail over Topic Cascade in some cases. An intuitive explanation to this phenomenon is that after transforming the preferences of users into one single scalar, WTC might lose the abundant topic modulated information contained in the original preference vectors. In order to overcome the drawback, we propose the second framework, namely ***Preference-enhanced Topic Cascade*** (PTC), to directly combine the user preference vectors and the topic distributions of cascades. Extensive experiments show that PTC constantly outperforms WTC and other baselines on both synthetic and real datasets. Fig. 1 provides a sketch of these two proposed frameworks. We sum up the major contributions of this paper as follows:

- To the best of our knowledge, we are the first ones to explore principled methods to combine both user preferences and the topic distributions of cascades when inferring networks. To achieve it, we design two frameworks, ***Weighted Topic Cascade*** (WTC) and ***Preference-enhanced Topic Cascade*** (PTC).
- We successfully adopt coordinate gradient descent [18] with proximal mapping [15] to learn parameters for the proposed frameworks.
- Based on substantial experiments on both synthetic and real datasets, we compare the performance of WTC and PTC to several state-of-the-art models. We show that although WTC outperforms previous models in many cases, it is less stable than PTC, which constantly outperforms other methods with an improvement of 4%$\sim$10% on the F-measure of inferred networks.
- Our work offers important insights for future research that attempts to adopt the rich topic information of users and cascades to infer topic-sensitive networks. Models attempting to transform topic modulated vectors into weighting scalars, like WTC, has the risk of losing the abundant information in the original vectors and thus hurt the performance of such models.

We organize the rest of the paper as follows: Section 2 introduces preliminary and background knowledge. The description of proposed frameworks are in Section 3. Section 4

Notation	Meaning
$\boldsymbol{t^c}$	The infection time of each user to cascade c
$\Delta^c_{j,i}$	The difference between the infection time of user i and j with respect to cascade c
$f(\cdot)$	Transmission likelihood
$S(\cdot)$	Survival function
$H(\cdot)$	Hazard function
N	Number of users in the network
K	Number of topics
T	The length of observation time window
$\boldsymbol{u_i}$	User preference vector of user i
$\boldsymbol{v_c}$	The topic distribution of cascade c
$w(\cdot)$	The weight function in WTC: $\mathbb{R}^K \times \mathbb{R}^K \to \mathbb{R}$
$h(\cdot)$	The mapping function in PTC: $\mathbb{R}^K \times \mathbb{R}^K \to \mathbb{R}^K$

Table 1: Important Notations

and 5 present the experimental evaluation of the frameworks on synthetic and real data, respectively. Finally, Section 6 concludes this article.

2 Preliminaries

2.1 Notations. First of all, we introduce some fundamental concepts and analysis tightly related to our research. We denote a diffusion network by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of users/nodes and $\mathcal{E}$ is the set of edges. The edges in $\mathcal{E}$ are directed, since the influence between two users are usually asymmetric. A cascade refers to a contagious message that spreads through the edges in the network. Usually, we are able to observe the time when a node is infected by the cascade. Thus, if we denote $|\mathcal{V}| = N$, each cascade can be represented as a vector of length N: $\boldsymbol{t}^c = (t^c_1, t^c_2, ..., t^c_N)$, where t^c_i is the infection time of the i^{th} node to the cascade c, and $t^c_i \in [0, T^c] \cup \{\infty\}$. T^c is the length of the observation time window for cascade c, and ∞ marks the nodes that are not infected within the time window. Each cascade has a separate clock, and it is set to 0 at the time point when the first node is infected. For simplicity, we usually fix the time windows of all cascades to the same length, a.k.a. $T^c = T, \forall c$.

The topic distribution of a cascade c is denoted as a nonnegative normalized vector: $\boldsymbol{v_c} = (v_{c,1}, v_{c,2}, ..., v_{c,K})$, where K is the number of topics, $v_{c,i} \geq 0, \forall i, c$ and $\sum_{i=1}^K v_{c,i} = 1, \forall c$. More particularly, $v_{c,i}$ can be interpreted as the probability that c belongs to the i^{th} topic. Accordingly, the j^{th} node's preference over topics is also denoted as a nonnegative normalized vector: $\boldsymbol{u_j} = (u_{j,1}, u_{j,2}, ..., u_{j,K})$, where $u_{j,i} \geq 0, \forall i, j$ and $\sum_{i=1}^K u_{j,i} = 1, \forall j$. Similarly, $u_{j,i}$ can be interpreted as the probability that an infected cascade of the j^{th} node belongs to the i^{th} topic. Furthermore, we assume that $\boldsymbol{v_c}$ and $\boldsymbol{u_j}$ are in the same topic space, meaning that each indexed topic has the same meaning in $\boldsymbol{v_c}$ and $\boldsymbol{u_j}$. For clearness, some important notations are listed in Table 1.

2.2 Survival Analysis for Network Inference. This subsection revisits survival analysis for inferring diffusion networks from observed cascades, which contains the preliminary knowledge to understand NetRate [6], MoNet [19], Topic Cascade [4] and our methods. The standard survival analysis does not adopt additional features, such as user preferences or the topic distributions of cascades. Similar to [4, 19], we attempt to add such additional information, while preserving the model's soundness and flexibility.

For a cascade $\boldsymbol{t}^c = (t^c_1, t^c_2, ..., t^c_N)$, we assume the influence of each node can only propagate forward time. In other words, an infected node will not have further influence on any node who is already infected. With respect to a specific cascade, **transmission likelihood** refers to the conditional likelihood that an infected node, say i, is infected by another node j, given the fact that j is infected. The transmission likelihood is a function of $\Delta^c_{j,i}$ and $\alpha_{j,i}$, where $\Delta^c_{j,i}$ is the time difference between i's infection time and j's ($\Delta^c_{j,i} = t^c_i - t^c_j$) and $\alpha_{j,i}$ is a non-negative model parameter associated with edge $j \to i$. If the directed edge $j \to i$ exists, $\alpha_{j,i}$ is positive, otherwise $\alpha_{j,i} = 0$. Therefore, the values of parameters determine the structure of the diffusion network, as well as the strength of each edge.

The form of the transmission likelihood is normally chosen from **exponential** distribution, **power law** distribution or **Rayleigh** distribution. In this paper, we select the Rayleigh distribution as the likelihood function for the purpose of demonstration and analysis, because it is shown to be the best among the three forms in most cases [4, 6, 19]. However, the proposed frameworks can naturally be extended to other function forms, since they are essentially independent from the choice of the form of transmission likelihood. The Rayleigh transmission likelihood is defined as follows:

(2.1)
$$f(t^c_i | t^c_j; \alpha_{j,i}) = \begin{cases} \alpha_{j,i} \Delta^c_{j,i} \cdot e^{-\frac{1}{2}\alpha_{j,i}(\Delta^c_{j,i})^2} & t^c_j < t^c_i \\ 0 & \text{otherwise} \end{cases}$$

Survival function, denoted by $S(\cdot)$, refers to the probability that a node j does **not** infect i by time t_i, given that the node j gets infected at time t_j. Thus, we have $S(t_i|t_j; \alpha_{j,i}) = 1 - \int_{t_j}^{t_i} f(t|t_j; \alpha_{j,i}) dt$. **Hazard function**, denoted by $H(\cdot)$, on the other hand, refers to the instantaneous infection rate, which is: $H(t_i|t_j; \alpha_{j,i}) = \frac{f(t_i|t_j; \alpha_{j,i})}{S(t_i|t_j; \alpha_{j,i})}$. With a little calculus, we can show that if the transmission likelihood takes the form of Rayleigh distribution, the corresponding survival and hazard functions are:

$$S(t^c_i | t^c_j; \alpha_{j,i}) = e^{-\alpha_{j,i} \cdot \frac{(\Delta^c_{j,i})^2}{2}}$$
$$H(t^c_i | t^c_j; \alpha_{j,i}) = \alpha_{j,i} \cdot \Delta^c_{j,i}$$

As a result, we can compute the likelihood of a particular cascade $\boldsymbol{t}^c = (t^c_1, t^c_2, ..., t^c_N)$. First, we name those nodes of which infection times are before one node as its **parents**. We assume that infections of nodes in one cascade are conditionally independent from each other given their own parents. Additionally, we further assume that every node can

only get infected once by the first parent who successfully infects it. This assumption originates from the principles of the Independent Cascade model [11]. Therefore, the conditional likelihood of an infected node i is the summation of the likelihood of numerous disjoint events (the node is infected by one of its parents yet survives the others):

$$\mathcal{L}_i(\boldsymbol{t}^c;\boldsymbol{A}) = \sum_{\{j:t_j^c<t_i^c\}} f(t_i^c|t_j^c;\alpha_{j,i}) \prod_{\{k:k\neq j,t_k^c<t_i^c\}} S(t_i^c|t_k^c;\alpha_{k,i})$$

where $\boldsymbol{A}$ is the matrix storing all $\alpha_{j,i}$, a.k.a. $\boldsymbol{A_{i,j}} = \alpha_{j,i}$

Because of the conditional independence, the likelihood of all infected nodes in the cascade will be:

$$\mathcal{L}(\boldsymbol{t}^c;\boldsymbol{A}) = \prod_{\{0<t_i\leq T\}} \sum_{\{j:t_j^c<t_i^c\}} f(t_i^c|t_j^c;\alpha_{j,i}) \prod_{\{k:k\neq j,t_k^c<t_i^c\}} S(t_i^c|t_k^c;\alpha_{k,i}) \tag{2.2}$$

Noticing that $f(t_i^c|t_j^c;\alpha_{j,i}) = S(t_i^c|t_j^c;\alpha_{j,i}) \cdot H(t_i^c|t_j^c;\alpha_{j,i})$, we can substitute it in the Eq. (2.2):

$$\mathcal{L}(\boldsymbol{t}^c;\boldsymbol{A}) = \prod_{\{0<t_i\leq T\}} \sum_{\{j:t_j^c<t_i^c\}} H(t_i^c|t_j^c;\alpha_{j,i}) \prod_{\{k:t_k^c<t_i^c\}} S(t_i^c|t_k^c;\alpha_{k,i})$$

Additionally, those uninfected nodes (a.k.a. $t_i^c = \infty$) should be considered, since they are informative as well. Therefore, we need to include more terms, which is the likelihood that uninfected nodes survive from the influence of infected nodes by T. The final likelihood of cascade c is:

$$\mathcal{L}(\boldsymbol{t}^c;\boldsymbol{A}) = \prod_{\{0<t_i\leq T\}} \prod_{\{m:t_m^c>T\}} S(T|t_i^c;\alpha_{i,m}) \prod_{\{j:t_j^c<t_i^c\}} S(t_i^c|t_j^c;\alpha_{j,i}) \sum_{\{k:t_k^c<t_i^c\}} H(t_i^c|t_k^c;\alpha_{k,i}) \tag{2.3}$$

Moreover, if we consider that each cascade is independent from each other, likelihood of all cascades will be:

$$\prod_c \mathcal{L}(\boldsymbol{t}^c;\boldsymbol{A})$$

Finally, we learn the parameters ($\boldsymbol{A}$) by minimizing the negative log-likelihood of all cascades, which is exactly the convex optimization problem solved by NetRate [6]:

$$\begin{aligned} \underset{\boldsymbol{A}}{\text{minimize}} \quad & -\textstyle\sum_c \log\{\mathcal{L}(\boldsymbol{t}^c;\boldsymbol{A})\} \\ \text{subject to} \quad & \alpha_{i,j} \geq 0, i, j = 1, 2, ..., N \end{aligned} \tag{2.4}$$

3 Description of Proposed Frameworks

3.1 Weighted Topic Cascade (WTC). ***Weighted Topic Cascade*** (WTC) model aims to infer topic-sensitive networks by incorporating both user preferences and topic distributions of cascades to the optimization problem presented in Eq. (2.4). Due to the nature of topic-sensitive networks, we allow the value of parameter $\alpha_{j,i}$ associated with edge $j \rightarrow i$ to change with regard to different topics. Therefore, we use a non-negative vector $\boldsymbol{\alpha_{j,i}}^\top = \left(\alpha_{j,i}^{(1)}, \alpha_{j,i}^{(2)}, ..., \alpha_{j,i}^{(K)}\right)$ to replace $\alpha_{j,i}$, where $\alpha_{j,i}^{(k)}$ indicates the edge's strength on the k^{th} topic. Corresponding to the Rayleigh function in the Eq. (2.1), WTC employs the following transmission likelihood function:

$$f(t_i^c|t_j^c) = \begin{cases} w\left(\boldsymbol{u_i},\boldsymbol{u_j}\right) \cdot \left[\boldsymbol{\alpha_{j,i}}^\top \cdot \boldsymbol{v_c} \cdot \Delta_{j,i}^c\right] \cdot \\ \exp\left[-\frac{1}{2}\boldsymbol{\alpha_{j,i}}^\top \cdot \boldsymbol{v_c} \cdot \left(\Delta_{j,i}^c\right)^2\right] & t_j^c < t_i^c \\ 0 & \text{otherwise} \end{cases} \tag{3.5}$$

where $w(\cdot)$ is a weighting function, which takes two user preference vectors as input and generates a non-negative scalar: $w : \mathbb{R}^K \times \mathbb{R}^K \rightarrow \mathbb{R}$. Generally speaking, $w(\cdot)$ computes the similarity between $\boldsymbol{u_i}$ and $\boldsymbol{u_j}$. In other words, if $\boldsymbol{u_i}$ and $\boldsymbol{u_j}$ are more similar, $w(\cdot)$ should output a larger value. Otherwise, $w(\cdot)$ emits a smaller value, which provides a penalty to the edge connecting node i and j. Moreover, in order to guarantee Eq. (3.5) is still a probability, $w(\cdot)$ may need to include a normalization factor to ensure the transmission likelihood sums up to 1. We provide users with the flexibility to select the $w(\cdot)$ function to fit various datasets, since the choice of $w(\cdot)$ might be dataset-dependent. However, we found that the weighting term in MoNet [19] generally works well in many cases, comparing to some other simple functions, such as the dot product of $\boldsymbol{u_i}$ and $\boldsymbol{u_j}$. Thus, in practice, we use the following $w(\cdot)$:

$$w\left(\boldsymbol{u_i},\boldsymbol{u_j}\right) = e^{\boldsymbol{u_i}\cdot\boldsymbol{u_j}} \cdot Z^{-1}$$

where $\boldsymbol{u_i} \cdot \boldsymbol{u_j}$ is the dot product of $\boldsymbol{u_i}$ and $\boldsymbol{u_j}$, and Z is a normalization factor: $\int_{\boldsymbol{u_i}}\int_{\boldsymbol{u_j}} e^{\boldsymbol{u_i}\cdot\boldsymbol{u_j}} d\boldsymbol{u_j} d\boldsymbol{u_i} = Z$. Later, we will show that in the parameter learning phase, Z can be canceled out and thus does not affect estimating each $\boldsymbol{\alpha_{i,j}}$.

Accordingly, the survival and hazard functions of WTC are readily to be computed:

$$\begin{aligned} S(t_i^c|t_j^c) =& 1 - \int_{\boldsymbol{u_i}}\int_{\boldsymbol{u_j}}\int_{t_j^c}^{t_i^c} f(t|t_j^c)dt\, d\boldsymbol{u_j} d\boldsymbol{u_i} \\ =& 1 - \int_{\boldsymbol{u_i}}\int_{\boldsymbol{u_j}} w\left(\boldsymbol{u_i},\boldsymbol{u_j}\right) d\boldsymbol{u_j} d\boldsymbol{u_i} \cdot \\ & \int_{t_j^c}^{t_i^c} \boldsymbol{\alpha_{j,i}}^\top \cdot \boldsymbol{v_c} \cdot (t - t_j^c) \cdot \exp\left[-\boldsymbol{\alpha_{j,i}}^\top \cdot \boldsymbol{v_c} \cdot \frac{(t-t_j^c)^2}{2}\right] dt \\ =& 1 - Z \cdot Z^{-1} \cdot \left(1 - \exp\left[-(\boldsymbol{\alpha_{j,i}})^\top \cdot \boldsymbol{v_c} \cdot \frac{(\Delta_{j,i}^c)^2}{2}\right]\right) \\ =& \exp\left[-(\boldsymbol{\alpha_{j,i}})^\top \cdot \boldsymbol{v_c} \cdot \frac{(\Delta_{j,i}^c)^2}{2}\right] \\ H(t_i^c|t_j^c) =& f(t_i^c|t_j^c)/S(t_i^c|t_j^c) = w\left(\boldsymbol{u_i},\boldsymbol{u_j}\right) \cdot \boldsymbol{\alpha_{j,i}}^\top \cdot \boldsymbol{v_c} \cdot \Delta_{j,i}^c \end{aligned}$$

Moreover, we need to impose empirical regularizers on the optimization problem of Eq. (2.4) to ensure the sparsity of the inferred network by WTC. We choose lasso regularizers to achieve this. At last, WTC aims to solve the

	Transmission Likelihood $f(t_i\|t_j)$ $(t_j < t_i, f(t_i\|t_j) = 0$ otherwise)	**Log Survival Function** $\log S(t_i\|t_j)$	**Hazard Function** $H(t_i\|t_j)$
NetRate	$\alpha_{j,i}(t_i - t_j)e^{-\frac{1}{2}\alpha_{j,i}(t_i-t_j)^2}$	$-\alpha_{j,i}\frac{(t_i-t_j)^2}{2}$	$\alpha_{j,i}(t_i - t_j)$
MoNet	$\frac{e^{\boldsymbol{u_i}\cdot\boldsymbol{u_j}}}{Z}\cdot\alpha_{j,i}(t_i - t_j)e^{-\frac{1}{2}\alpha_{j,i}(t_i-t_j)^2}$	$-\alpha_{j,i}\frac{(t_i-t_j)^2}{2}$	$\frac{e^{\boldsymbol{u_i}\cdot\boldsymbol{u_j}}}{Z}\cdot\alpha_{j,i}(t_i - t_j)$
Topic Cascade	$\boldsymbol{\alpha_{j,i}}^\top\cdot\boldsymbol{v_c}\cdot(t_i - t_j)e^{-\frac{1}{2}\boldsymbol{\alpha_{j,i}}^\top\cdot\boldsymbol{v_c}\cdot(t_i-t_j)^2}$	$-\boldsymbol{\alpha_{j,i}}^\top\cdot\boldsymbol{v_c}\cdot\frac{(t_i-t_j)^2}{2}$	$\boldsymbol{\alpha_{j,i}}^\top\cdot\boldsymbol{v_c}\cdot(t_i - t_j)$
WTC	$w(\boldsymbol{u_i},\boldsymbol{u_j})\cdot\boldsymbol{\alpha_{j,i}}^\top\cdot\boldsymbol{v_c}\cdot(t_i - t_j)$ $\exp\left\{-\frac{1}{2}\boldsymbol{\alpha_{j,i}}^\top\cdot\boldsymbol{v_c}\cdot(t_i - t_j)^2\right\}$	$-\boldsymbol{\alpha_{j,i}}^\top\cdot\boldsymbol{v_c}\cdot\frac{(t_i-t_j)^2}{2}$	$w(\boldsymbol{u_i},\boldsymbol{u_j})\cdot\boldsymbol{\alpha_{j,i}}^\top\cdot\boldsymbol{v_c}\cdot(t_i - t_j)$
PTC	$\boldsymbol{\alpha_{j,i}}^\top\cdot[\boldsymbol{v_c}\circ h(\boldsymbol{u_i},\boldsymbol{u_j})]\cdot(t_i - t_j)$ $\exp\left\{-\frac{1}{2}\boldsymbol{\alpha_{j,i}}^\top\cdot[\boldsymbol{v_c}\circ h(\boldsymbol{u_i},\boldsymbol{u_j})]\cdot(t_i - t_j)^2\right\}$	$-\boldsymbol{\alpha_{j,i}}^\top\cdot[\boldsymbol{v_c}\circ h(\boldsymbol{u_i},\boldsymbol{u_j})]\cdot\frac{(t_i-t_j)^2}{2}$	$\boldsymbol{\alpha_{j,i}}^\top\cdot[\boldsymbol{v_c}\circ h(\boldsymbol{u_i},\boldsymbol{u_j})]\cdot(t_i - t_j)$

Table 2: Rayleigh transmission likelihood, log survival functions and hazard functions used by different models

following optimization problem:

$$\begin{aligned}(3.6)\quad &\underset{\boldsymbol{A}}{\text{minimize}} && -\textstyle\sum_c \log\{\mathcal{L}(\boldsymbol{t}^c;\boldsymbol{A})\} + \lambda\sum_i\sum_j\|\boldsymbol{\alpha_{j,i}}\|_2\\ &\text{subject to} && \boldsymbol{\alpha_{j,i}}\succeq\mathbf{0}, i, j = 1, 2, ..., N\end{aligned}$$

3.2 Preference-enhanced Topic Cascade (PTC). As we stated before, transforming user preference vectors into a single weighting term may lose the rich topic-wise information contained in the vectors. This might largely affect the performance of WTC. To compensate this, ***Preference-enhanced Topic Cascade*** (PTC) uses the following transmission likelihood function:

$$(3.7)\quad f(t_i^c|t_j^c) = \begin{cases}\boldsymbol{\alpha_{j,i}}^\top\cdot[\boldsymbol{v_c}\circ h(\boldsymbol{u_i},\boldsymbol{u_j})]\cdot\Delta_{j,i}^c\cdot & \\ \exp\left(-\frac{1}{2}\boldsymbol{\alpha_{j,i}}^\top\cdot[\boldsymbol{v_c}\circ h(\boldsymbol{u_i},\boldsymbol{u_j})]\cdot(\Delta_{j,i}^c)^2\right) & t_j^c < t_i^c\\ 0 & \text{otherwise}\end{cases}$$

where $\circ$ denotes the Hadamard product, and $h(\boldsymbol{u_i},\boldsymbol{u_j})$ is a function of which inputs are user preference vectors, and the output is a non-negative vector of length K: $h:\mathbb{R}^K\times\mathbb{R}^K\to\mathbb{R}^K$. If two users have large values on the i^{th} element of the preference vectors, we consider they are more probable to both infect and be infected on the i^{th} topic. As a result, function h tends to output a larger value on the i^{th} element as well. Similar to WTC, readers are free to test different h functions to find the optimal one. In practice, we found out that simply adopting the Hadamard product of user preference vectors turns out to be a good choice in terms of both efficiency and effectiveness:

$$h(\boldsymbol{u_i},\boldsymbol{u_j}) = \boldsymbol{u_i}\circ\boldsymbol{u_j}$$

Likewise, PTC's survival and hazard functions are:

$$\begin{aligned}S(t_i^c|t_j^c) =& \exp\left(-\boldsymbol{\alpha_{j,i}}^\top\cdot[\boldsymbol{v_c}\circ h(\boldsymbol{u_i},\boldsymbol{u_j})]\cdot\frac{(\Delta_{j,i}^c)^2}{2}\right)\\ H(t_i^c|t_j^c) =& \boldsymbol{\alpha_{j,i}}^\top\cdot[\boldsymbol{v_c}\circ h(\boldsymbol{u_i},\boldsymbol{u_j})]\cdot\Delta_{j,i}^c\end{aligned}$$

Afterwards, PTC solves the same formed optimization problem as WTC attempts to solve in Eq. (3.6). In order to provide a straightforward comparison, we summarize our frameworks and previous models in Table 2.

3.3 Parameter Learning From Table 2, it is obvious to see that the transmission likelihood, log survival function and hazard functions of PTC framework have the same form of Topic Cascade model's. The only difference is that PTC replaces $\boldsymbol{v_c}$ in Topic Cascade with $\boldsymbol{v_c}\circ h(\boldsymbol{u_i},\boldsymbol{u_j})$. Therefore, we can adopt the same learning process of Topic Cascade with simple substitutions to learn the parameters of PTC. For brevity, we omit the introduction of the parameter learning for PTC, which can be found in [4].

Next, we focus on solving the problem in Eq. (3.6) for WTC framework. Similar to previous models [4, 6, 19], we discover that the problem in Eq. (3.6) can be factorized into N subproblems, which can be addressed in parallel. The i^{th} subproblem only involves the vectors $\boldsymbol{\alpha_{*,i}}$. If ℓ_i denotes the negative log-likelihood terms in Eq. (3.6) that involve $\boldsymbol{\alpha_{*,i}}$, the i^{th} subproblem can be stated as follows:

$$\begin{aligned}(3.8)\quad &\underset{\{\alpha_{j,i}\}_{j=1}^N}{\text{minimize}} && \ell_i(\{\alpha_{j,i}\}_{j=1}^N) + \lambda\textstyle\sum_j\|\boldsymbol{\alpha_{j,i}}\|_2\\ &\text{subject to} && \boldsymbol{\alpha_{j,i}}\succeq\mathbf{0}, j = 1, 2, ..., N\end{aligned}$$

Because of the group lasso regularizers, the problem in Eq. (3.8) is non-smooth. However, it is separable with respect to the subscription j in $\boldsymbol{\alpha_{j,i}}$. Therefore, we adopt the block coordinate descent framework for non-differentiable function [18] to perform parameter estimation. The algorithm runs in several rounds, and in each round, we iterate j to sequentially estimate each $\boldsymbol{\alpha_{j,i}}$.

The estimated $\boldsymbol{\alpha_{j,i}}$ in the l^{th} iteration, denoted by $\boldsymbol{\alpha_{j,i}^l}$, minimizes ℓ_i with respect to $\boldsymbol{\alpha_{j,i}}$ with other model parameters fixed to their current values:

$$(3.9)\quad \boldsymbol{\alpha_{j,i}^l} = \underset{\boldsymbol{\alpha_{j,i}}\succeq\mathbf{0}}{\text{argmin}}\ell_{j,i}(\boldsymbol{\alpha_{j,i}}) + \lambda\|\boldsymbol{\alpha_{j,i}}\|_2$$

where $\ell_{j,i}(\boldsymbol{\alpha_{j,i}}) = \ell_i\left(\{\boldsymbol{\alpha_{1,i}^l},...,\boldsymbol{\alpha_{j-1,i}^l},\boldsymbol{\alpha_{j,i}},\boldsymbol{\alpha_{j+1,i}^{(l-1)}},...,\boldsymbol{\alpha_{N,i}^{(l-1)}}\}\right)$.

Since Eq. (3.9) is still non-differentiable, we adopt proximal gradient algorithm to solve it. First, by applying WTC's log survival and hazard functions to Eq. (3.9), we can compute the gradient of $\ell_{j,i}(\boldsymbol{\alpha_{j,i}})$ with regard to $\boldsymbol{\alpha_{j,i}}$:

$$\begin{aligned}(3.10)\quad \nabla\ell_{j,i}(\boldsymbol{\alpha_{j,i}}) =& \frac{\partial\ell_{j,i}(\boldsymbol{\alpha_{j,i}})}{\boldsymbol{\alpha_{j,i}}} = \sum_{\{c|t_j^c<t_i^c\}}\frac{(\Delta_{j,i}^c)^2}{2}\cdot\boldsymbol{v_c} -\\ &\sum_{\{c|t_i^c<=T\}}\frac{w(\boldsymbol{u_i},\boldsymbol{u_j})\cdot\Delta_{j,i}^c\cdot\boldsymbol{v_c}}{S_j^c + w(\boldsymbol{u_i},\boldsymbol{u_j})\cdot\boldsymbol{\alpha_{j,i}}^\top\cdot\boldsymbol{v_c}\cdot\Delta_{j,i}^c}\end{aligned}$$

where S_j^c is defined as:

$$S_j^c = \sum_{k<j, t_k^c < t_i^c} w(\boldsymbol{u_i}, \boldsymbol{u_j}) \cdot \boldsymbol{\alpha_{k,i}^l}^\top \cdot \boldsymbol{v_c} \cdot \Delta_{k,i}^c + \sum_{k>j, t_k^c < t_i^c} w(\boldsymbol{u_i}, \boldsymbol{u_j}) \cdot \boldsymbol{\alpha_{k,i}^{l-1}}^\top \cdot \boldsymbol{v_c} \cdot \Delta_{k,i}^c$$

Notice that the normalization factor Z of $w(\boldsymbol{u_i}, \boldsymbol{u_j})$ disappears in Eq. (3.10) after taking derivative. This means in practice, we do not need to actually compute Z for the parameter learning. This is important to the efficiency of WTC, since computing Z might be extremely time consuming.

Theoretically, $\nabla \ell_{j,i}(\boldsymbol{\alpha_{j,i}})$ needs to be Lipschitz continuous. One can easily prove this by showing that the Hessian of $\ell_{j,i}(\boldsymbol{\alpha_{j,i}})$ is upper bounded by a constant, say L:

$$\frac{\partial^2 \ell_{j,i}(\boldsymbol{\alpha_{j,i}})}{\partial \boldsymbol{\alpha_{j,i}}^2} = \sum_{\{c | t_i^c <= T, t_j^c < t_i^c\}} \frac{w(\boldsymbol{u_i}, \boldsymbol{u_j}) \cdot (\Delta_{j,i}^c)^2 \cdot \boldsymbol{v_c} \cdot \boldsymbol{v_c}^\top}{\left[S_j^c + w(\boldsymbol{u_i}, \boldsymbol{u_j}) \cdot \boldsymbol{\alpha_{j,i}}^\top \cdot \boldsymbol{v_c} \cdot \Delta_{j,i}^c\right]^2} \leq L$$

Furthermore, let $\boldsymbol{\alpha_{j,i}^*} = \boldsymbol{\alpha_{j,i}^{(l-1)}} - \gamma \cdot \nabla \ell_{j,i}(\boldsymbol{\alpha_{j,i}^{(l-1)}})$, where γ is the step size in the gradient descent. γ can be either a value in the interval of $(0, 1/L]$ or decided by line search [15]. Due to the convexity of $\ell_{j,i}(\boldsymbol{\alpha_{j,i}})$, $\boldsymbol{\alpha_{j,i}^*}$ is exactly the solution to Eq. (3.9) disregarding the non-negative constraint ($\boldsymbol{\alpha_{j,i}} \succeq \mathbf{0}$) and the lasso regularizer. To obtain the final solution to Eq. (3.9), we adopt the proximal algorithm, yielding to the following proximal mapping:

$$\boldsymbol{\alpha_{j,i}^l} = \underset{\boldsymbol{\alpha_{j,i}} \succeq \mathbf{0}}{\operatorname{argmin}} \frac{1}{2\gamma} \|\boldsymbol{\alpha_{j,i}} - \boldsymbol{\alpha_{j,i}^*}\|_2^2 + \lambda \|\boldsymbol{\alpha_{j,i}}\|_2$$

Let $\mathbf{prox}_{\|\cdot\|_p}(\boldsymbol{\alpha_{j,i}^*}) = \underset{\boldsymbol{\alpha_{j,i}}}{\operatorname{argmin}} \frac{1}{2\gamma} \|\boldsymbol{\alpha_{j,i}} - \boldsymbol{\alpha_{j,i}^*}\|_2^2 + \lambda \|\boldsymbol{\alpha_{j,i}}\|_p$, where $\|\cdot\|_p$ denotes L-p norm, then the following equation always holds according to the Moreau decomposition [15]:

$$\mathbf{prox}_{\|\cdot\|_p}(\boldsymbol{\alpha_{j,i}^*}) = \boldsymbol{\alpha_{j,i}^*} - \mathbf{prox}_{\|\cdot\|_*}(\boldsymbol{\alpha_{j,i}^*})$$

where $\|\cdot\|_*$ is the dual norm of $\|\cdot\|_p$.

Notice that $\mathbf{prox}_{\|\cdot\|_*}(\boldsymbol{\alpha_{j,i}^*})$ is exactly to project $\boldsymbol{\alpha_{j,i}^*}$ on a dual norm ball of radius $\lambda \cdot \gamma$, which yields:

$$\mathbf{prox}_{\|\cdot\|_*}(\boldsymbol{\alpha_{j,i}^*}) = \begin{cases} \boldsymbol{\alpha_{j,i}^*} & \|\boldsymbol{\alpha_{j,i}^*}\|_* \leq \lambda \cdot \gamma \\ \boldsymbol{\alpha_{j,i}^*} \cdot \frac{\lambda \cdot \gamma}{\|\boldsymbol{\alpha_{j,i}^*}\|_*} & \|\boldsymbol{\alpha_{j,i}^*}\|_* > \lambda \cdot \gamma \end{cases}$$

By setting $p = 2$ and recalling that the dual of $\|\cdot\|_2$ is itself, we can obtain the final solution to Eq. (3.9):

(3.11)
$$\boldsymbol{\alpha_{j,i}^l} = \begin{cases} 0 & \|\boldsymbol{\alpha_{j,i}^*}\|_2 \leq \lambda \cdot \gamma \\ \left[\boldsymbol{\alpha_{j,i}^*} \cdot \left(1 - \frac{\lambda \cdot \gamma}{\|\boldsymbol{\alpha_{j,i}^*}\|_2}\right)\right]_+ & \|\boldsymbol{\alpha_{j,i}^*}\|_2 > \lambda \cdot \gamma \end{cases}$$

where the notation, $[\cdot]_+$, denotes a special function: it sets all negative elements in one vector to be zeros. This function guarantees that the constraint $\boldsymbol{\alpha_{j,i}} \succeq \mathbf{0}$ is not violated. At last, we conclude the parameter learning for WTC framework in Algorithm 1.

Algorithm 1 Weighted Topic Cascade (WTC)

Input:
Cascade records: $\boldsymbol{t^c}, \forall c$
User preferences: $\boldsymbol{u_i}, \forall i$
Cascade topic distributions: $\boldsymbol{v_c}, \forall c$
Maximal iterations: l_{max}
Output: $\boldsymbol{\alpha_{j,i}}, \forall i, j$

Initialize $\boldsymbol{\alpha_{j,i}^0}$ for all i and j
for all i=1 to N **do**
  $l = 0$
  while ($l \leq l_{max}$ and $\boldsymbol{\alpha_{i,i}^l}$ not converges) **do**
    $l = l + 1$
    for j = 1 to N and j $\neq$ i **do**
      update $\boldsymbol{\alpha_{j,i}^l}$ according to Eq. (3.11)
    end for
  end while
end for
Set $\boldsymbol{\alpha_{i,i}^l} = \mathbf{0}, \forall i$
return $\boldsymbol{\alpha_{j,i}^l}, \forall i, j$

4 Synthetic Data based Evaluation

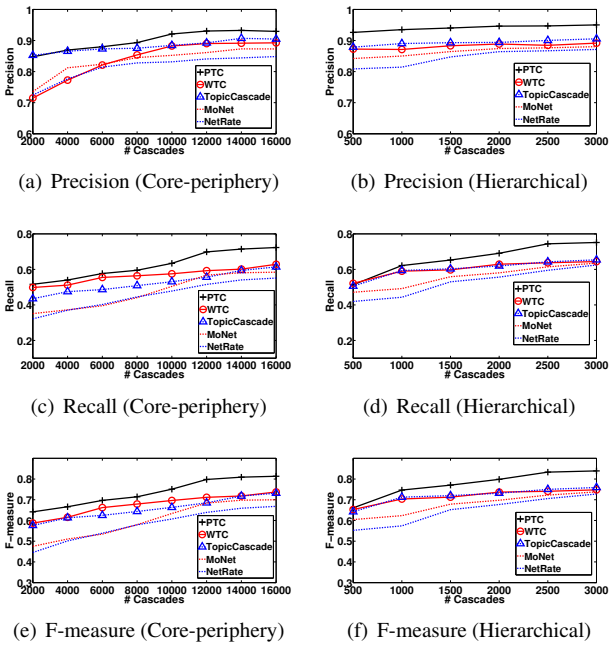
In this section, we use experiments based on synthetic data to provide both qualitative and quantitative evaluation of WTC and PTC. We compare our approaches with three state-of-the-art models: NetRate [6], MoNet [19] and Topic Cascade [4]. Two major reasons of introducing synthetic data to test the proposed frameworks are as follows: (1) for a real information network, the true values of the parameters associated with each edge are usually not obtainable. Synthetic network, on the contrary, offers us a more controllable environment to quantitatively evaluate the models in details. (2) When inferring networks by incorporating both user preferences and the topic distributions of cascades, the experiments in this section unveils the potential problems of compressing users' preferences into simple weighting terms.

4.1 Synthetic Network Generation. We use Kronecker Graph [13] to generate two synthetic networks. Kronecker Graph is selected due to that it well captures many important properties of real networks and therefore mimics a true network well. The first core-periphery network [5] is generated by using a 2-node core having parameters [0.9, 0.5; 0.5, 0.3]. The second hierarchical network [3] is generated by using a 2-node core having parameters [0.9, 0.1; 0.1, 0.9]. The final obtained two networks contain 512 nodes and 1207 edges, and 512 nodes and 512 edges, respectively.

The K in the experiments is set to 5, and we do not observe significant change of the result when K varies. However, one should notice that this might not be true for real datasets. For the i^{th} node, we randomly generate a normalized vector of length K to be $\boldsymbol{u_i}$. For each edge $j \rightarrow i$, we also randomly generate $\boldsymbol{\alpha_{j,i}}$, of which each element is drawn from the interval [0, 1).

4.2 Cascade Simulation. In order to initiate a cascade, we first need to generate its topic distribution. Similar to $\boldsymbol{u_i}$, $\boldsymbol{v_c}$ is also randomly generated. Then we randomly pick a node

in the network as the first infected node and set its infection time to 0. Afterwards, we simulate the diffusion process on the network as follows: we add the first infected node into a list storing the frontier of diffusion process. In each iteration, we remove the node with the earliest infection time from the list, and mark it has been infected. Then, we allow it to infect each of its not yet infected neighbors. According to the distribution defined by the transmission likelihood on the edge, we first draw the time difference between the infection moment of the node's neighbor and itself. The sum of the time difference and the node's infection time is exactly the infection time of its neighbor. If the result is no larger than the predefined T, we add this neighbor into the list and record its infection time. Of course, if one node is successfully infected by multiple parents, we consider the earliest infection time as its true infection time. This process continues until the list is empty and we set the infection time of all uninfected nodes to ∞. If the cascade only infects one node in the network, we simply discard this cascade. We select Eq. (3.7) as the transmission likelihood function to generate cascades due to that comparing to other models' transmission likelihood functions, it best fits real datasets (we will show this in Section 4).



(a) Precision (Core-periphery) (b) Precision (Hierarchical)

(c) Recall (Core-periphery) (d) Recall (Hierarchical)

(e) F-measure (Core-periphery) (f) F-measure (Hierarchical)

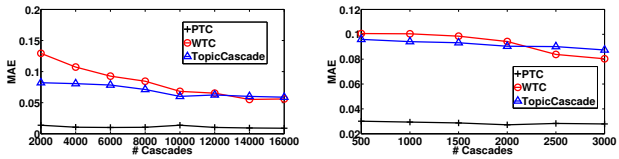
Figure 2: Performance on Synthetical Data

4.3 Experimental Results. We generate sets of cascades and feed different algorithms with the infection times of nodes. We also provide the generated user preferences and the topic distributions of cascades, if the algorithm requires them. Readers should notice that this is an ideal scenario that the true value of each $\boldsymbol{u_i}$ and $\boldsymbol{v_c}$ is known. However, for real datasets, $\boldsymbol{u_i}$ and $\boldsymbol{v_c}$ need to be approximated by methods such as LDA [2], and we might need to tune the value of K to get the best performance. In order to conduct a fair comparison, we set the convergence condition and maximal allowed iterations to be the same for different models.

We first examine the performance of different models to uncover the structures of diffusion networks, which reflects the quality of inferred edges. Based on the estimated results, if there exists any parameter associated with an edge is nonzero, we consider the edge inferred by the model. By comparing inferred edges to the ground truth, we can calculate the precision, recall and F-measure for each model when using different numbers of simulated cascades. The results on the core-periphery and hierarchical network are shown in Fig. 2. In general, because of the structural difference between the core-periphery and hierarchical network, we need more cascades to get good results on the core-periphery network. It is easy to observe that PTC outperforms all the other methods in terms of precision, recall and F-measure.

For the generated core-periphery network, Topic Cascade and PTC model can achieve good precision values even with fewer number of cascades. However, PTC performs better as the number of cascades increases. The other three models, on the other hand, need more cascades to reach their stable points. Roughly speaking, the recall values of all models monotonically increase along with the increase of the number of cascades. This outcome is straightforward, since if we generate more cascades, more edges will be included during the message dissemination process, which provides us the chance to unveil more unknown edges in the network. Comparing the performance of MoNet to NetRate's, we can clearly see the benefit of incorporating user preferences. However, this might not be true for a topic modulated model: WTC framework does not significantly outperforms Topic Cascade. What's worse, as suggested in the core-periphery network, the precision of WTC might be much worse than Topic Cascade when we do not observe many cascades. This indicates the potential problem of adopting WTC to infer networks with certain structures and fewer cascades. As mentioned previously, a potential explanation to this phenomena is that after WTC transforms user preference vectors into weighting scalars, the original topic-modulated information in the vectors might not be properly preserved. PTC framework, on the other hand, can avoid this since the $h(\cdot)$ function still has a topic-modulated output, which can be directly combined with the topic distributions of cascades.

Next, we conduct a more quantitive evaluation of different models: computing the errors of estimated parameter values with respect to the ground truth. In this experiment, we only compare the results generated by PTC, WTC and Topic Cascade model, since they differentiate the parameter values on various topics, which is exactly the scenario of



(a) MAEs (Core-periphery) (b) MAEs (Hierarchical)

Figure 3: Mean Absolute Errors (MAEs) on Synthetical Data

the generated ground truth. Therefore, if we denote an estimated parameter by $\hat{\alpha}_{i,j}^{(k)}$ and the ground truth by $\alpha_{i,j}^{(k)*}$, we can compute the Mean Absolute Error (MAE) as:

$$MAE = \frac{\sum_i \sum_j \sum_k (\hat{\alpha}_{i,j}^{(k)} - \alpha_{i,j}^{(k)*})}{N \cdot N \cdot K}$$

Fig. 3 displays the MAEs of three models on the two generated networks. Having in mind that the true parameter values are in the interval of [0, 1), the results of Fig. 3 demonstrate that PTC is more stable comparing to the other two models and its MAE is much smaller.

5 Real Data based Evaluation

This section introduces the experiments based on two public datasets to evaluate the proposed frameworks: a MemeTracker dataset[2] [4, 7, 12] and a Sina Weibo dataset[3] [21].

5.1 MemeTracker Data. The MemeTracker dataset records the times when different blog and news websites post "memes". A "meme" refers to a textual message (such as a short phrase or a long statement) that spreads widely in the network and is viewed as the cascade. Different websites are viewed as nodes in the network, because they can actively generate or repost memes. However, edges among them are not directly observable. Similar to [4, 7], we use the hyperlinks mentioned together with memes to reveal the information source of a website, since they point to related articles. Through this, we can formulate a hyperlink network among nodes and view it as the ground truth.

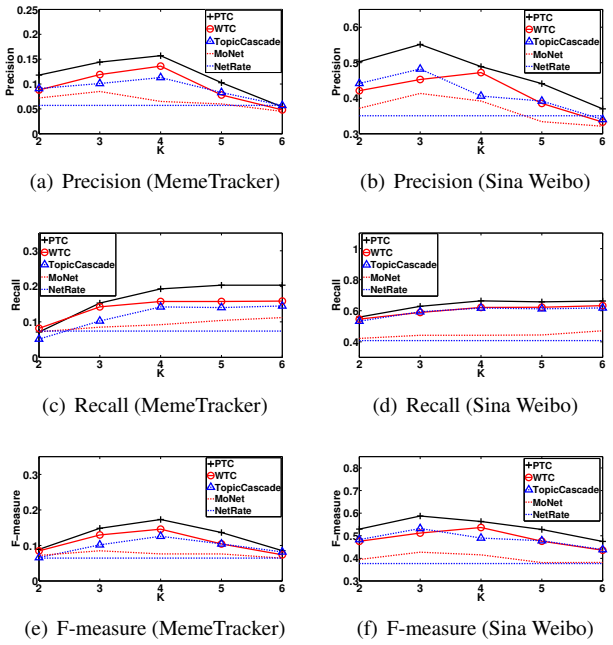
Based on the raw MemeTracker data, we first extract the 1,000 most active websites in the January of 2009 (a.k.a. websites that posted most memes). We record all their posted memes in that month, along with their infection times. If a website posted a meme more than once, we use the earliest time as the infection time. Afterwards, the observation window (T) for all cascades is set to 1 week, and we remove all the memes that do not have enough time to be observed. We also remove websites' memes that do not contain the information sources, since we do not have the corresponding diffusion edges to evaluate them. Finally, the obtained dataset consists of 1,000 nodes, 4,323 edges and 2,762 cascades. Moreover, to obtain the topic distributions of cascades, we apply JGibbLDA[4], a java implemented LDA model based on Gibbs sampling [9, 10], to the textual content of memes. We also gather the textual content of a node's all posted memes and use JGibbLDA to extract its preference.

5.2 Sina Weibo Data. Sina Weibo is a micro blogging website that is widely used in China, which is similar to the Twitter network. The users in the network are deemed as nodes and connected to each other through "follow" relationship, which allows a user to see the tweets posted by the people s/he has followed and further responds to it. Therefore, "follower-followee" relationships in the Sina Weibo form an information network that facilitates the spread of contagions, a.k.a. tweets. Similar to the MemeTracker dataset, we first find the 1,000 most active users in the August of 2012, which are the users who posted most tweets. We also retrieve all their posted tweets and times to be used as cascades. T is also set to 1 week in this case. Moreover, the "follower-followee" network of users is also retrievable from the original dataset. Likewise, a user's preference vector and the topic distribution of a cascade are also obtained by applying JGibbLDA to the textual content of a user's all posted tweets and the specific tweet, respectively. At last, we obtain a dataset containing 1,000 nodes, 2,686 edges and 2,281 cascades.

5.3 Experimental Results. Since the ground truths of the two real datasets only contain network structures, we are only able to evaluate the precision, recall and F-measure of inferred edges. By tuning the value of K, we present the comparison of different models in Fig. 4. Since NetRate does not incorporate topic related information, it is simply a horizontal line in Fig. 4. Moreover, each reported result is the best performance of the corresponding model under different choices of λ ($\lambda = \{0.05, 0.1, 0.2, 0.4, 0.8\}$).

In general, we found out that it is much harder to infer the MemeTracker network, probably because of that it has more edges and a more complex structure. In terms of the F-measure, different algorithms can reach their optimal values when $K = 3$ or $K = 4$, and it is clear to see that PTC framework constantly outperforms the other models. Although in most cases, WTC framework prevails over the previous methods, it seems that WTC is less stable comparing to PTC: in the Sina Weibo dataset, WTC is even worse than Topic Cascade when $K = 2$ and $K = 3$. We can also observe that the recall values of different models cease to increase or only increase marginally when K reaches a certain threshold. Meanwhile, the number of false positive edges grows, which largely hurts the precision. In practice, K can be decided by techniques like cross-validation.

[2] http://snap.stanford.edu/data/memetracker9.html
[3] http://arnetminer.org/Influencelocality#b2352
[4] http://jgibblda.sourceforge.net/



(a) Precision (MemeTracker) (b) Precision (Sina Weibo)

(c) Recall (MemeTracker) (d) Recall (Sina Weibo)

(e) F-measure (MemeTracker) (f) F-measure (Sina Weibo)

Figure 4: Performance on Real Data

6 Conclusion

Uncovering a hidden diffusion network from the traces of disseminated messages is a problem that has been extensively studied in recent years. Motivated by the discovery that topic related information often helps the network inference, previous work has incorporated either the user preferences over topics or the topic distributions of cascades. This paper explores methods to incorporate both of these two types of information for better inferring diffusion networks. To achieve this, we propose two principled frameworks: Weighted Topic Cascade (WTC) and Preference-enhanced Topic Cascade (PTC). The proposed frameworks formulate the network inference task as non-smooth convex optimization problems and adopt coordinate gradient descent with proximal algorithm to solve them. Based on substantial experiments on both synthetic and real datasets, we demonstrate that methods such as WTC, which compress user preferences into additional weighting terms, may potentially lose the rich topic modulated information that contained in the preference vectors. PTC, on the other hand, is more stable and constantly outperforms WTC and several state-of-the-art algorithms on both synthetic and real datasets.

7 Acknowledgement

This work is supported in part by NSF through grants CNS-1115234, and OISE-1129076 and Huawei grant.

References

[1] B. Abrahao, F. Chierichetti, R. Kleinberg, and A. Panconesi. Trace complexity of network inference. In *KDD '13*, pages 491–499. ACM, 2013.

[2] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning research*, 3:993–1022, 2003.

[3] A. Clauset, C. Moore, and M. E. Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98–101, 2008.

[4] N. Du, L. Song, H. Woo, and H. Zha. Uncover topic-sensitive information diffusion networks. In *AISTATS '13*, pages 229–237, 2013.

[5] P. Erdős and A. Rényi. On the evolution of random graphs. *Magyar Tud. Akad. Mat. Kutató Int. Közl*, 5:17–61, 1960.

[6] M. Gomez-Rodriguez, D. Balduzzi, and B. Schölkopf. Uncovering the temporal dynamics of diffusion networks. In *ICML '11*, 2011.

[7] M. Gomez-Rodriguez, J. Leskovec, and A. Krause. Inferring Networks of Diffusion and Influence. In *KDD '10*, 2010.

[8] M. Gomez-Rodriguez, J. Leskovec, and B. Schölkopf. Structure and Dynamics of Information Pathways in On-line Media. In *WSDM '13*, 2013.

[9] T. Griffiths. Gibbs sampling in the generative model of latent dirichlet allocation. 2002.

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

[11] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *KDD '03*, pages 137–146. ACM, 2003.

[12] J. Leskovec, L. Backstrom, and J. Kleinberg. Meme-tracking and the dynamics of the news cycle. In *KDD '09*, pages 497–506. ACM, 2009.

[13] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani. Kronecker graphs: An approach to modeling networks. *Journal of Machine Learning Research*, 11:985–1042, 2010.

[14] P. Netrapalli and S. Sanghavi. Learning the graph of epidemic cascades. *SIGMETRICS '12*, 40(1):211–222, 2012.

[15] N. Parikh and S. Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, pages 1–96, 2013.

[16] M. G. Rodriguez and B. Schölkopf. Submodular inference of diffusion networks from multiple trees. In *ICML '12*, pages 489–496, 2012.

[17] T. M. Snowsill, N. Fyson, T. De Bie, and N. Cristianini. Refining causality: who copied from whom? In *KDD '11*, pages 466–474. ACM, 2011.

[18] P. Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications*, 109(3):475–494, 2001.

[19] L. Wang, S. Ermon, and J. E. Hopcroft. Feature-enhanced probabilistic models for diffusion network inference. In *ECML PKDD'12*, pages 499–514. Springer-Verlag, 2012.

[20] S. Wang, X. Hu, P. S. Yu, and Z. Li. Mmrate: Inferring multi-aspect diffusion networks with multi-pattern cascades. In *KDD '14*, pages 1246–1255. ACM, 2014.

[21] J. Zhang, B. Liu, J. Tang, T. Chen, and J. Li. Social influence locality for modeling retweeting behaviors. In *IJCAI '13*, pages 2761–2767. AAAI Press, 2013.