Knowl Inf Syst
DOI 10.1007/s10115-016-0955-7

REGULAR PAPER

Uncovering and predicting the dynamic process of information cascades with survival model

Linyun Yu[1] · **Peng Cui**[1] · **Fei Wang**[2] · **Chaoming Song**[3] · **Shiqiang Yang**[1]

Received: 17 March 2016 / Revised: 18 March 2016 / Accepted: 2 April 2016

Abstract Cascades are ubiquitous in various network environments. Predicting these cascades is decidedly nontrivial in various important applications, such as viral marketing, epidemic prevention, and traffic management. Most previous works have focused on predicting the final cascade sizes. As cascades are dynamic processes, it is always interesting and important to predict the cascade size at any given time, or to predict the time when a cascade will reach a certain size (e.g., the threshold for an outbreak). In this paper, we unify all these tasks into a fundamental problem: *cascading process prediction*. That is, given the early stage of a cascade, can we predict its cumulative cascade size at any later time? For such a challenging problem, an understanding of the micromechanism that drives and generates the macrophenomena (i.e., the cascading process) is essential. Here, we introduce behavioral dynamics as the micromechanism to describe the dynamic process of an infected node's neighbors getting infected by a cascade (i.e., one-hop sub-cascades). Through data-driven analysis, we find out the common principles and patterns lying in the behavioral dynamics and propose the novel NEtworked WEibull Regression model for modeling it. We also propose a novel method for predicting cascading processes by effectively aggregating behavioral dynamics and present a scalable solution to approximate the cascading process with a theoretical guarantee. We evaluate the proposed method extensively on a large-scale social network dataset. The results demonstrate that the proposed method can significantly outperform other state-of-the-art methods in multiple tasks including cascade size prediction, outbreak time prediction, and cascading process prediction.

Keywords Information cascades · Social network · Dynamic processes prediction

✉ Linyun Yu
yuly12@mails.tsinghua.edu.cn

[1] Tsinghua National Laboratory for Information Science and Technology, Department of Computer Science and Technology, Tsinghua University, Beijing, China

[2] School of Software Engineering, Beijing University of Technology, Beijing, China

[3] Department of Physics, University of Miami, Coral Gables, FL, USA

1 Introduction

In social networks, people can often observe the actions from their neighbors. When they decide to make the same choice as others have made earlier, these local actions can lead to interesting macroinformation-spreading dynamics, also known as cascades. In real life, the spreading of information is ubiquitous and can be found everywhere, from social media and marketing to epidemics and traffics. Predicting these cascades is very important for many applications. For example, in the field of online marketing, knowing the future trend of products may allow for the development of smarter strategies to maximize profit.

There is a growing body of research on information cascades due to their potential in various important applications such as viral marketing, epidemic prevention, and traffic management. Most works have focused on characterizing these information cascades and discovering their patterns in structures, contents, and temporal dynamics.

Recently, there has been heightened research interest regarding the predictive modeling of information cascades. Earlier works focused on predicting the final size of information cascades (the number of nodes that take part in the cascade) based on the content, and behavioral and structural features [3,6]. As only large cascades are of interest in most real applications, Cui et al. [6] proposed a data-driven approach for predicting whether the final size will surpass the threshold for an outbreak. More recently, Cheng et al. [3] investigated the problem of continuously predicting whether the cascade size will double in the future. However, these were all regarding the cascade size, which is only part of the story. Information cascade is a dynamic process, that is, it changes all the time. Meanwhile, the temporal scale is critical for understanding the cascading mechanism. Furthermore, it is decidedly nontrivial to predict when a cascade breaks out, or, more ambitiously, to predict the evolving process of a cascade (i.e., the cascading process, as shown in 1a). In this paper, we will move one step forward by asking: Is the cascading process predictable? That is, given the early stage of an information cascade, can we predict its cumulative cascade size at any later time?

It is apparent that the targeted problem is far more challenging than those in previous works. The commonly used cascade-level macro features for size prediction, such as the content, increasing speed, and structures in the early stage, are not distinctive and predictive enough to determine future the cascade sizes. A fundamental technique to address this problem is to look into the *micro*-mechanism of cascading processes. Intuitively, an information cascading process can be decomposed into multiple local (one-hop) sub-cascades. When a node gets involved in a cascade, one or more of its offspring nodes will also get involved in the cascade with a temporal scaling. If the dynamic process of these sub-cascades can be accurately modeled, the cascade process can be straightforwardly predicted by an additive function of these local sub-cascades.

Here, we will exploit *behavioral dynamics* as the micromechanism to represent the above-mentioned dynamic process of local sub-cascades. Given a node becoming involved in a cascade at t_0, its behavioral dynamic aims to capture the changing process of the cumulative number of its offspring nodes that will become involved in the cascade over time. By definition, this is a nondecreasing counting process and can be well represented by survival model [17]. Few research works have exploited the survival theory to model how the occurrence of an event at a node affects the time of its occurrence at other nodes (i.e., the diffusion rate, or the probability of a node taking part in the cascade at any time), and their results have demonstrated the superiority of continuous-time survival model to uncover temporal processes [11,18]. However, the targeted problem of these works was to uncover the hidden diffusion networks, which presumes the parameters of the survival function on each edge to

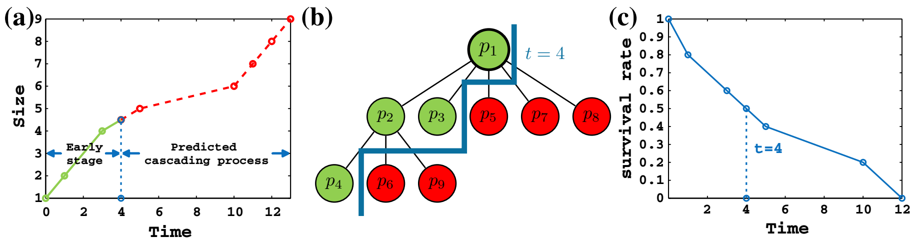


Fig. 1 Illustrations of cascading process prediction. **a** The early stage of a cascading process before t. **b** The partially observed cascade, where nodes in *green* (*red*) represent the observed (unobserved) nodes involved before (*after*) t. **c** The cascading re-tweeting dynamics of the followers of p_1 with respect to time (color figure online)

be fixed. This causes the unexpected result that all cascades with the same root node (or early involved nodes) will be anticipated to have the same cascading processes, which makes these models inapplicable for our problem.

In this paper, we propose a novel method for cascading process prediction, as shown in Fig. 1. Given the early stage of a cascading process before t in Fig. 1a, we illustrate the partially observed cascade as shown in Fig. 1b, where nodes in green (red) represent the observed (unobserved) nodes involved before (after) t. Given the behavioral dynamics of node p_1 represented by its survival rates, and the number of its offspring nodes that have become involved before t, we can predict the cumulative number of its offspring nodes that partake in the cascade at any time $t' > t$. After conducting similar predictions for all the observed nodes, the cascading process after t can be predicted using an additive function over all local predictions via behavioral dynamics.

However, the modeling of behavioral dynamics and the prediction of the cascading process based on continuous-time survival theory entails many challenges. First, it is unclear which distribution form the behavioral dynamics follows. Although Exponential and Rayleigh distributions are commonly used to characterize the temporal scaling of pairwise interactions, the behavioral dynamics in this paper are a reflection of collective behaviors and have been proven to be inconsistent with these simple distributions for real data. Second, the parameters in survival models are difficult to interpret, which limits the generalizability of the learned model. Given the distribution form of the data, the parameters of the survival model can always be learned from real data in a maximum likelihood manner. However, it is unclear what these parameters stand for and the learned model cannot be generalized to out-of-sample nodes (i.e., nodes whose behavioral dynamic data are not included in the data). Third, the predictive models based on survival theory are computationally expensive due to the continuous-time characteristic, which makes them infeasible in real applications. How to design an effective and interpretable model for behavioral dynamics modeling and a scalable solution for cascading process prediction are still open issues.

In this paper, we conduct extensive statistical analysis on large-scale real data and find that although the behavioral dynamics cannot be well captured by simple distributions such as Exponential or Rayleigh distributions, the general form of the Exponential and the Rayleigh, or the Weibull distribution, can reasonably preserve the characteristics of the behavioral dynamics. Also, we discover strong correlations between the parameters of a node's behavioral dynamics and its neighbor nodes' behavioral features. We thus propose a NEtworked WEibull Regression (NEWER) model for parameter learning of behavioral dynamics. In particular, besides the maximum likelihood estimation term, we also assume that the para-

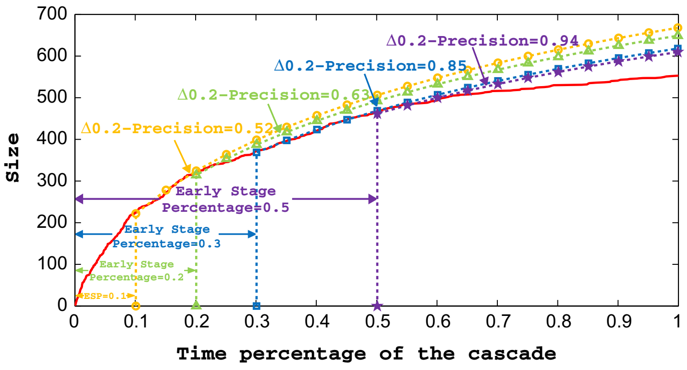


Fig. 2 Showcase of cascading process prediction for a real cascade. The *red line* represents the ground truth cascading process. The *other lines* are prediction results based on different early-stage information (color figure online)

meters of a node can be regressed by the behavioral features of its neighbor nodes and thus impose networked regularizers to improve the interpretability and generalizability of the model. Based on the behavioral dynamics, we further propose an additive model for cascading process prediction. To enable scalability, we will propose two efficient sampling strategies which can make approximations with a theoretical guarantee under two different scenarios.

We extensively evaluate the proposed method using a complete dataset from a population-level social network in China, including over 320 million users, 1.2 billion edges and 340 million cascades. In all the testing scenarios, the proposed method significantly outperforms other baseline methods. Figure 2 is a showcase of cascading process prediction using the proposed method. We show that by accurately modeling the behavioral dynamics of social network users, we can predict the entire cascading process with only early 30 % of the cascade information and get an average precision of 0.85 with a 20 % error tolerance. Furthermore, accurate predictions of the final cascade size and cascade outbreaking time are implied in the predicted cascading process.

The main contributions of this paper are:

1. Informed by the cascading size prediction works, we move one step further in attempting to solve the cascading process prediction problem, which implies several vital problems such as cascade size prediction, outbreaking time prediction, and evolving process prediction.
2. We find out the common principles and patterns lying in behavioral dynamics and accordingly propose a novel NEtworked Weibull Regression model for behavioral dynamics modeling, which significantly improves the interpretability and generalizability of traditional survival models.
3. We propose a novel method for predicting macrocascading processes by aggregating microbehavioral dynamics and propose a scalable solution to approximate the cascading process with a theoretical guarantee.

2 Related works

Prediction regarding cascades Recently, many methods have been proposed for making predictions regarding cascades. Most have focused on predicting the future size of a cascade, often via selecting vital nodes and placing sensors on them. For example, Cohen et al. [4] focused on exploring the topological characteristics of the cascade. Cui et al. [6] proposed optimizing the size prediction problem using dynamic information. Cheng et al. [3] introduced temporal features into the problem and predicted the growing size of a cascade. However, the features described in these works are complex and difficult to measure. Rather than attempt to predict the cascade size, we focus on predicting the cascading process, simultaneously considering both time and volume information. In addition, our work successfully solves the problem by first creating a link from temporal features to survival models and then finding the correlations with other kinds of features.

Survival model The survival model is a method of analysis according to the time duration before one or more events occur. Recently, researchers have started modeling information diffusion using continuous models. Myers et al. [15] proposed CONNIE to infer the diffusion network based on convex programming while leaving the transmission rate to be fixed. Later, Rodriguez et al. [18] proposed NETRATE, which allows the transmission rate to be different along different edges. Subsequently, Rodriguez et al. [11] provided an additive model and a multiplicative model to describe information propagation based on survival theory. Most of these works focused on discovering the rules and patterns to the edges in a social network. Therefore, they are hard to extend to making predictions for cascades since the correlation between transmission rates on edges is small. In contrast, we focus more on predictive modeling by grouping correlated edges so that we can make predictions for edges based on the information of other edges.

Influence modeling and maximization Influence modeling and maximization aims to evaluate the importance of users in social networks. This was first proposed by Domingos et al. [8] to identify early starters that trigger a large cascade. Then, Kempe et al. [13] proposed the stochastic cascade model to formalize the problem, and Chen et al. [2] proposed a scalable solutions. Recently, the approach has been extended to add opinion effects [1,10], topic effects [7], or time decay effects [19] into the models. Our work is distinct from existing works because rather than quantifying the influence on nodes, we predict the cascading process.

Comparing to the preliminary version [21], this one comprises a substantial amount of additional algorithmic and experimental efforts and contributions. Key points of differences lie in the following aspects: First, as the sampling strategy proposed in our conference paper can only solve the cascade size prediction problem with linear complexity, we extend the sampling section and give another sampling strategy to provide a chance to solve the cascading process prediction problem with linear complexity using balanced binary search tree. Second, we collect, sample, and publish a cascading process dataset from Tencent Weibo, one of the largest Twitter-style Web sites in China. To our knowledge, this is the first public cascading process dataset in social networks. Besides cascading process analysis, it also provides opportunities to understand and model the human activity patterns in social networks using this dataset. Third, we report a series of statistical tests that examine the efficiency of the new sampling strategy for cascading process prediction, and find that the method reduces the calculation number by several magnitudes, and the advantage is more obvious as the cascade final size increases.

3 Preliminaries

This section presents the discovered patterns and validated hypotheses to support the model design and solution.

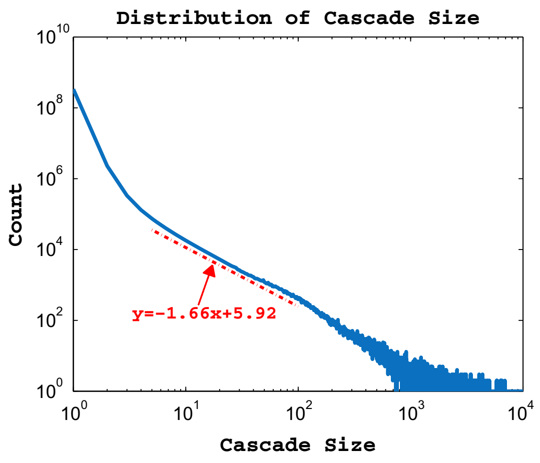


Fig. 3 Distribution of cascade size. The *straight red line* is the linear fitting result for the *blue curve*, which shows that the size distribution fits a Power-Law distribution (color figure online)

3.1 Dataset description

The data in this paper were collected from Tencent Weibo, one of the largest Twitter-style Web sites in China. We collected all the cascades generated in the 10 days between Nov 15 and Nov 25 2011. There were in total 320 million users and 340 million cascades[1] with their explicit cascading processes. Figure 3 shows the distribution of the cascade size, illustrating that it follows a Power-Law distribution. As we aim to predict the cascading process, a dataset[2] was sampled from the data by filtering out cascades with sizes less than 5 and maintaining the remaining 0.46 million cascades for statistical analysis and experiments.

Each cascade in the dataset is constituted of a number of triads, e.g., <userid u, time t, target userid r >, which means that user u re-tweet the post from user r at the time t. Due to privacy limitations, all user IDs were anonymized. To the best of our knowledge, this is the first public cascading process dataset for social networks. In addition to cascading process analysis, this dataset also provides opportunities to understand and model human activity patterns in social networks.

3.2 Characteristics of behavioral dynamics

As mentioned above, behavioral dynamics play a central role in uncovering and predicting cascade processes. Here, we investigate the characteristics of behavioral dynamics to inform the modeling of behavioral dynamics. By definition, the behavioral dynamics of a user capture the changing process of the cumulative number of his/her followers that re-tweet a post after the user re-tweets the post. Then, the behavioral dynamics of a user can be straightforwardly represented by averaging the size growth curve of all sub-cascades that spread to the user and his/her followers. However, Fig. 4 shows that the size growth curves vary significantly for different sub-cascades of the same user, indicating that such a representation is not suitable for characterizing behavioral dynamics. Here, we normalize the size growth process by the final cascade size and adopt the survival function to describe the behavioral dynamics where the survival rate represents the percentage of nodes that have not been, but will be, infected. As

[1] Here, the cascades are information cascades. When a user re-tweets/generates a post, several of his/her followers will further re-tweet the post and so on to form an information cascade.

[2] The dataset is complete and publicly available at http://www.thumedia.org.

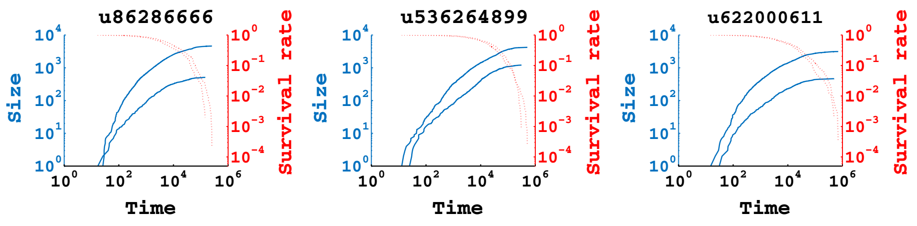


Fig. 4 The size growth curves and their corresponding survival function for 3 users

Table 1 Parametric models

Model	Density function	Survival function	Hazard function	ks-statistic in Weibo
Exponential	$\lambda_i e^{-\lambda_i t}$	$e^{-\lambda_i t}$	λ_i	0.2741
Power law	$\frac{\alpha_i}{\delta}\left(\frac{t}{\delta}\right)^{-\alpha_i-1}$	$\left(\frac{t}{\delta}\right)^{-\alpha_i}$	$\frac{\alpha_i}{t}$	0.9893
Rayleigh	$\alpha_i t e^{-\alpha_i \frac{t^2}{2}}$	$e^{-\alpha_i \frac{t^2}{2}}$	$\alpha_i t$	0.7842
Weibull	$\frac{k_i}{\lambda_i}\left(\frac{t}{\lambda_i}\right)^{k_i-1} e^{-\left(\frac{t}{\lambda_i}\right)^{k_i}}$	$e^{-\left(\frac{t}{\lambda_i}\right)^{k_i}}$	$\frac{k_i}{\lambda_i}\left(\frac{t}{\lambda_i}\right)^{k_i-1}$	**0.0738**

Weibull distribution in bold performs much better than other distributions

shown in Fig. 4, a user's survival function is quite stable for different sub-cascades, although their size growth patterns may vary.

Then, can we use the behavioral dynamics represented by the survival function to predict the size growth curve of a sub-cascade? We provide a positive answer with the assistance of early-stage information. For example, if we know the sub-cascade size at an early time t_0, then the survival function can be straightforwardly transformed from the percentage dimension into the size dimension.

3.3 Parameterizing behavioral dynamics

For the ease of computation and modeling, we need to parameterize the behavioral dynamics in our case. In state of the art, Exponential and Rayleigh distributions are often used to describe the dynamics of user behaviors in different settings [9,12]. Here, we test these distribution hypotheses on real data and find that they cannot sufficiently capture both the shape and scale characteristics of behavioral dynamics. Thus, we turn to the general form of Exponential and Rayleigh distributions, the Weibull distribution [16], and find it to be adequate for parameterizing behavioral dynamics. To quantify the effect of parameterization, we calculate the Kolmogorov–Smirnov (KS) statistic for the four candidate distributions as shown in Table 1. This statistic shows that the Weibull distribution performs much better than Exponential and Rayleigh distributions. The improvement is attributed to the high degree of freedom of the Weibull distribution as it has two parameters λ and k, to control the scale and shape of the behavioral dynamics, respectively.

3.4 Covariates of behavioral dynamics

If sub-cascades for all users are sufficient, the parameters of behavioral dynamics can be directly learned from data. However, this suffers from several drawbacks, including that (1) some users may have no or very sparse sub-cascades in the training dataset, which makes these

Table 2 Behavioral and structural features for users

Behavioral features	
$inflowrate$	The number of the posts a user received in a certain period
$outflowrate$	The number of the posts a user sent in a certain period
$inflowrate^{avg}_{F(i)}$	Average inflow rate of followers to the user, or $\frac{\sum_{j\in F(i)} retweet(j)\cdot inflow(j)}{\sum_{j\in F(i)} retweet(j)}$ where $F(i)$ describes The followers of user i (and the same as following)
$retweetrate^{avg}_{F(i)}$	Average re-tweet rate of followers to the user, or $\frac{\sum_{j\in F(i)} retweet(j)\cdot retweet\ rate(j)}{\sum_{j\in F(i)} retweet(j)}$
Structural features	
$follower\ number$	Number of followers of the user
$followee\ number$	Number of users this user follows

users' behavioral dynamics inaccurate or even unknown, and (2) it is difficult to interpret the parameters directly learned from data, which limits our insights into the behavioral dynamics. To address these, we investigate the covariates of behavioral dynamics. As the behavioral dynamics of a user captures the collective responses of his/her followers, we assume that the parameters of the user's behavioral dynamics should be correlated with the behavioral features of his/her followers (network neighbors). Hence, we extract a set of behavioral features for each user as listed in Table 2.[3] For each user with enough sub-cascades in our dataset, we learn their λ and k directly from the data. Then, we calculate the correlations between the learned parameters and their followers' collective behavioral features. The examples given in Fig. 5 indicate obvious correlations between the learned parameters with these behavioral features. Therefore, we can use these behavioral features as covariates to regress the parameters of behavioral dynamics.

3.5 From behavioral dynamics to cascades

After validating that the behavioral dynamics can potentially be accurately modeled and predicted, the key problem is whether we can derive the macrocascading process from microbehavioral dynamics. Intuitively, the cascading process cannot be perfectly predicted at early stage by behavioral dynamics. Given any time t, we can only use the behavioral dynamics of the users that were involved before t to predict the cascading process after t. Consequently, the prediction coverage is restricted to all the followers of these users, while users beyond this scope are neglected. These uncovered users may potentially affect the performance of the cascading process prediction.

Fortunately, we observe the following two interesting phenomena in real data.

Minor dominance Although each user has behavioral dynamics, the behavioral dynamics of different users make significantly different contributions to the cascading process. Intuitively, the behavioral dynamics of an active user with 1 million followers should contribute much more than that of an inactive user with 5 followers, and the data support this concept.

[3] We think that a follower with a different re-tweet number will have different effects to the user, and so we modify the weights of each term $follower\_avg\_inflow\_rate$ and $follower\_avg\_retweet\_rate$.

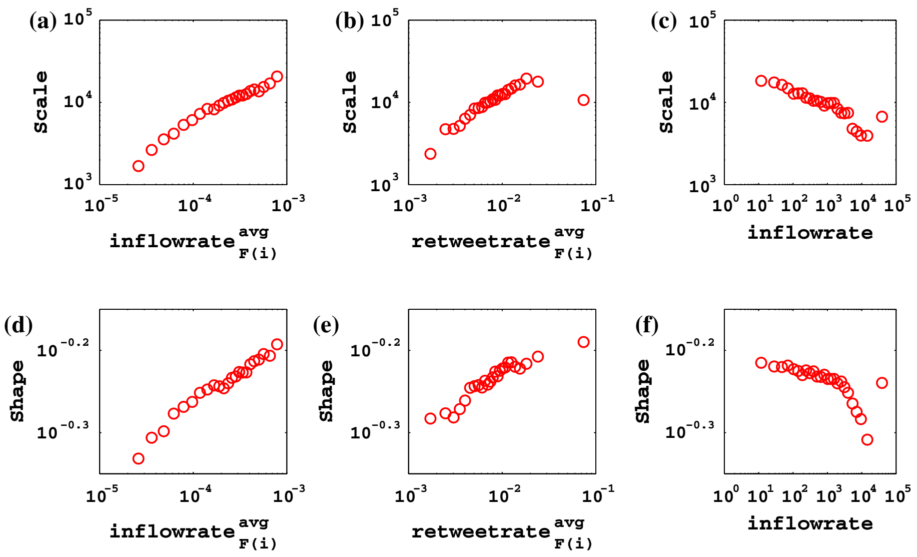


Fig. 5 Correlations between the survival function parameters and the behavioral features

According to Fig. 6a, it can be observed that the behavioral dynamics of a very small number of nodes dominate the cascading process. This supports the idea of just using the behavioral dynamics of only these dominant nodes for cascading process prediction.

Early-stage dominance Enlightened by the minor dominance phenomenon, we further ask whether the dominant nodes are prone to joining cascades in early stages. Figure 6b depicts the time distribution of these dominant nodes joining cascades, and we can see that most of these nodes actually join cascades in the very early stages.

Taking these two phenomena into account, it is safe to design a model exploiting the behavioral dynamics of infected nodes in the early stage to predict the cascading process.

4 Methodology

This section introduces the NEtworked WEibull Regression (NEWER) and cascade prediction methods in detail.

4.1 Problem statement

Given a network $G = \langle U, A \rangle$, where U is a collection of nodes and A is a set of pairwise directed/undirected relationships. An event (e.g., tweet) can be originated from one node and spread (e.g., by re-tweeting) to its neighboring nodes. A cascade is typically formed by repeating this process. Therefore, a cascade can be represented by a set of nodes $C = \{u_1, u_2, ...u_m\}$, where u_1 is the root node. In a cascade, each node will be infected by the event only once, so it is tree-structured. For every node u_i in the cascade, we denote its parent node as $rp(u_i)$. The time stamp that u_i gets infected is $t(u_i)$ and $t(u_i) \leq t(u_{i+1})$. Then the partial

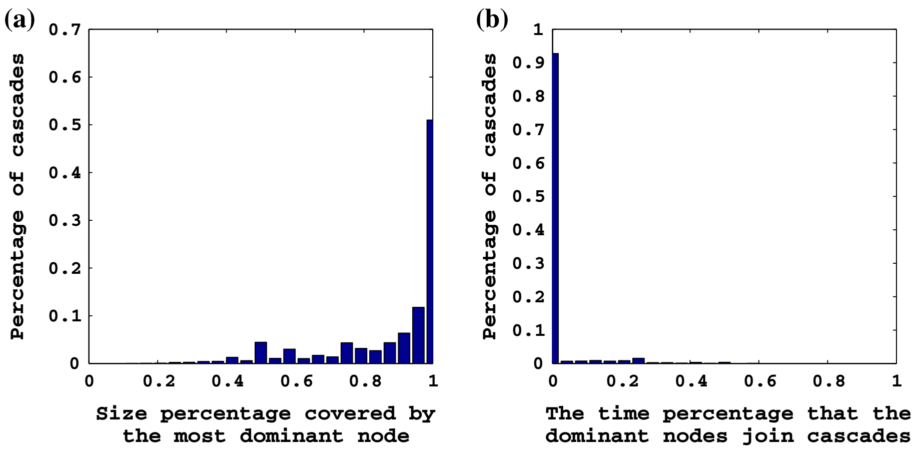


Fig. 6 Minor dominance and early-stage dominance in information cascades

cascade before time t is denoted by $C_t = \{u_i | t(u_i) \leq t\}$, and its size $size(C_t) = |C_t|$ where $|.|$ is the cardinality of a set. Then the cascade prediction problem can be defined as follows.

Cascade prediction Given the early stage of a cascade C_t, predict the cascade size $size(C_{t'})$ $(t' \geq t)$.

4.2 Survival analysis

Survival analysis is a branch of statistics that deals with the analysis of the time duration before one or more events occur, such as death in biological organisms and failure in mechanical systems [14]. It is a useful technique for cascade prediction. More concretely, let τ_0 be a nonnegative continuous random variable representing the waiting time until the occurrence of an event with probability density function $f(t)$. The survival function

$$S(t) = Pr\{\tau_0 \geq t\} = \int_t^{\infty} f(t) \tag{1}$$

encodes the probability that the event occurs after t. The hazard rate is defined as the event rate at time t, conditional on survival until time t or later $(\tau_0 \geq t)$, i.e.,

$$\lambda(t) = \lim_{\mathrm{d}t \to 0} \frac{Pr(t \leq \tau_0 < t + \mathrm{d}t | \tau_0 \geq t)}{\mathrm{d}t} = \frac{f(t)}{S(t)} \tag{2}$$

$S(t)$ and $\lambda(t)$ are the two core quantities in survival analysis.

4.3 NEtworked WEibull regression (NEWER) model

The Weibull distribution is commonly used in survival analysis. In a network scenario, if we consider the time that an event (e.g., re-tweet) happened on a node as a survival process, we can fit a Weibull distribution to the survival time of node i. Then, its corresponding density, survival and hazard functions are, respectively,

$$f_i(t) = \frac{k_i}{\lambda_i}\left(\frac{t}{\lambda_i}\right)^{k_i-1} \exp^{-\left(\frac{t}{\lambda_i}\right)^{k_i}} \tag{3}$$

$$S_i(t) = \exp^{-\left(\frac{t}{\lambda_i}\right)^{k_i}} \tag{4}$$

$$h_i(t) = \frac{k_i}{\lambda_i}\left(\frac{t}{\lambda_i}\right)^{k_i-1} \tag{5}$$

where $t > 0$ is the average event occurrence time to node i, and $\lambda_i > 0$ and $k_i > 0$ are the scale and shape parameter of the Weibull distribution, respectively. In the following, we will assume that the network nodes are users and that the event is re-tweeting.

Likelihood of re-tweeting dynamics Supposing that there are N users in total, T_i is a set of m_i time stamps and each element $T_{i,j}$ indicates the j-th re-tweet time stamp to the post of the ith user. We sort these time stamps in increasing order so that $T_{i,j+1} \geqslant T_{i,j}$. We assume that $T_{i,j} \geq 1$ and $T_{i,m_i} > 1$. Then, the likelihood of the event data can be written as follows:

$$\begin{aligned} L(\lambda, k) &= \prod_{i=1}^{N}\prod_{j=1}^{m_i}\left(h_i(T_{i,j}) \cdot S_i(T_{i,j})\right) \\ &= \prod_{i=1}^{N}\prod_{j=1}^{m_i}\left(k_i \cdot T_{i,j}^{k_i-1} \cdot \lambda_i^{-k_i} \cdot e^{-T_{i,j}^{k_i} \cdot \lambda_i^{-k_i}}\right) \end{aligned} \tag{6}$$

$$\log L(\lambda, k) = \sum_{i=1}^{N} l_i(\lambda_i, k_i) \tag{7}$$

where $l_i(\lambda_i, k_i) = m_i \log k_i + (k_i - 1)\sum_{j=1}^{m_i} \log T_{i,j} - m_i k_i \log \lambda_i - \lambda_i^{-k_i} \sum_{j=1}^{m_i} T_{i,j}^{k_i}$.

As discovered in Sect. 3.4, the survival characteristics of a user are correlated with his/her behavioral features. Then, we can parameterize those parameters in the personalized Weibull distributions using those behavioral features. More formally, let x_i be an r dimensional feature vector for user i. We can parameterize λ_i and k_i with the following linear functions:

$$\log \lambda_i = \log x_i * \beta \tag{8}$$

$$\log k_i = \log x_i * \gamma \tag{9}$$

where β and γ are r-dimensional parameter vectors for λ and k. We attempt to determine the scale and shape parameter for every user so that the likelihood of the observed data is maximized. At the same time, we can also get the parameter vectors for out-of-sample extensions.

We use Eqs. (8) and (9), respectively, to replace λ_i and k_i in the log-likelihood function in Eq. (7) to solve the parameters. To further enhance the interpretability, we also add $\ell 1$ sparsity regularizers on β and γ to enforce model sparsity. Combining everything, we can obtain the *NEtworked WEibull Regression* (NEWER) formulation which aims to minimize the following objectives:

$$F(\lambda, k, \beta, \gamma) = G_1(\lambda, k) + \mu G_2(\beta, \lambda) + \eta G_3(\gamma, k) \tag{10}$$

$$G_1(\lambda, k) = -\log L(\lambda, k) \tag{11}$$

$$G_2(\lambda, \beta) = \frac{1}{2N}\left\|\log \lambda - \log X \cdot \beta\right\|^2 + \alpha_\beta \left\|\beta\right\|_1 \tag{12}$$

$$G_3(k, \gamma) = \frac{1}{2N} \|\log k - \log X \cdot \gamma\|^2 + \alpha_\gamma \|\gamma\|_1 \tag{13}$$

4.4 Cascading process prediction

With the NEWER model, we can learn the behavioral dynamics for each user. Given the behavioral dynamics of a user, once we get the early stage information of a sub-cascade triggered by this user, we can transform the rate dimension to size dimension and make predictions on the remained dynamic process of the sub-cascade. After predicting all the sub-cascades, we can straightforwardly aggregate these sub-cascades into the whole cascade.

5 Solution

5.1 Optimization for NEWER

To minimize $F(\lambda, k, \beta, \gamma)$ in Eq. (10), we first prove that the function is lower bounded. We have the following theorem.

Theorem 1 $F(\lambda, k, \beta, \gamma)$ *has a global minimum.*

Proof See the appendix.

With this theorem, the following coordinate descent strategy can be used to solve the problem with guaranteed convergence. At each iteration, we solve the problem with one group of variables with others fixed.

$$\begin{cases} For\ it = 1, \ldots, it_{\max} \\ \lambda^{[it+1]} = \text{argmin}_\lambda F(\lambda, k^{[it]}, \beta^{[it]}, \gamma^{[it]}) \\ k^{[it+1]} = \text{argmin}_k F(\lambda^{[it+1]}, k, \beta^{[it]}, \gamma^{[it]}) \\ \beta^{[it+1]} = \text{argmin}_\beta F(\lambda^{[it+1]}, k^{[it+1]}, \beta, \gamma^{[it]}) \\ \gamma^{[it+1]} = \text{argmin}_\gamma F(\lambda^{[it+1]}, k^{[it+1]}, \beta^{[it+1]}, \gamma) \end{cases} \tag{14}$$

To solve the subproblem with respect to λ or k, we can use Newton's method. For the sub-problem with respect to β and γ, we can use the standard LASSO solver [20].

5.2 Base model for cascading process prediction

We will first present a basic model to achieve this goal.

The entire flow of the basic model is illustrated in Algorithm 1:

When a new node u_i is added to the cascade at $t(u_i)$, the algorithm will launch a process to estimate the final size of the sub-cascade that u_i will generate, with a temporal size counter $replynum(u_i)$ and a survival function $S_{u_i}(t)$ starting at $t(u_i)$. If u_i is involved by others, the algorithm also increases the temporal size of the re-tweet set of its parent $rp(u_i)$ by one.

After all the pre-deadline information is collected, the result will be finalized by aggregating all the values estimated by every sub-cascade process. Since the post number is at most $|V|$ (all nodes in the network are involved into the cascade), the value of death rate

Algorithm 1 Basic Model

```
Input:
    Set of users $U$ involved in the cascade $C$ before time $t_{limit}$, survival functions of users $S_{u_j}(t)$, predicting
    time $t_e$;
Output:
    Size of cascade $size\left(C_{t_e}\right)$;
1: for all user $u_i \in U$ do
2:     creates a subcascade process with $replynum(u_i) = 0$
3:     if $u_i$ is not root node then
4:         $replynum(rp(u_i)) = replynum(rp(u_i)) + 1$
5:     end if
6: end for
7: $sum = 1$
8: for all user $u_i \in U$ do
9:     $deathrate(u_i) = \max\left(1 - S_{u_i}(t_{limit} - t(u_i)), \frac{1}{|V|}\right)$
10:    $fdrate(u_i) = \max\left(1 - S_{u_i}(t_e - t(u_i)), \frac{1}{|V|}\right)$
11:    $sum = sum + \frac{replynum(u_i) \cdot fdrate(u_i)}{deathrate(u_i)}$
12: end for
13: return $size\left(C_{t_e}\right) = sum$
```

$deathrate(u_i)$ and final death rate $fdrate(u_i)$ (complement to their survival rates) at line 9 and line 10 is set to be $1/|V|$ when it is lower than $1/|V|$.

Complexity analysis Only constant time operations are involved in the two for-loops. Therefore, the complexity of the algorithm is $O(n)$ where n is the number of users in the cascade.

6 Sampling models for efficient prediction

Real applications often need to predict the cascading process of all cascades dynamically in a streaming environment so that the process can be continuously monitored, which imposes more burden on prediction tasks. Although the basic model solves the estimation problem, it is still computationally expensive and does not fit with a streaming environment in real time.

To make the algorithm scalable, an instinctive idea is to reduce the number of recalculations for sub-cascades by using previous calculations with an acceptable approximation error. Here, we propose two models: The first is to make a fixed-time-query prediction, and the second is to make an arbitrary-time-query prediction.

6.1 Fixed-Time-Query Sampling Model

For fixed-time-query prediction, the prediction time t_e in the prediction task remains unchanged. An example is to predict the size of the cascade in one hour starting from any time point when we make a prediction. For such settings, the final death rate $fdrate(u_i)$ for every sub-cascade will not change. Thus, we can utilize the following two facts to make the estimation process more efficient:

1) For a sub-cascade generated by u_i, the estimation of the size will always be zero if there is no user involved in it, which means that we can ignore the calculation.
2) If we do not re-estimate the final number of a sub-cascade (when there is no new user involved in it), the temporal size counter $replynum(u_i)$ and final death rate $edrate(u_i)$ will not change, but the death rate $deathrate_{u_i}(t)$ will increase over time. Suppose that

the previous time stamp of the sub-cascade set estimation is t_0, this will cause a relative error rate of $1 - \frac{deathrate_{u_i}(t_0)}{deathrate_{u_i}(t_1)} \leq \frac{deathrate_{u_i}(t_1)}{deathrate_{u_i}(t_0)} - 1$ at t_1.

By exploiting these two rules, the relative error rate will be at most ϵ if we sample $S_u^{-1}(1 - (1 + \epsilon) \cdot deathrate_{u_i}(t_0))$ as the next calculation time point and re-estimate the final number of the sub-cascade at that time. Then, we propose a sampling model as shown in Algorithm 2.

Complexity Analysis The following theorem analyzes the complexity of Algorithm 2.

Theorem 2 *With an overall $O(n \log(|V|))$ when counting to estimate the number of sub-cascades, the sampling model can approximate a fixed-time-size of the cascade estimated by the basic model at any time with a relative error rate of at most ϵ.*

Proof For each approximation request, we only need to report the number directly; for every new sub-cascade, the initial operation number is also constant, and we need to perform the threshold adjustment at most $O(\log_{1+\epsilon}(|V|))$ times for sub-cascades which have new users involved in, since the lower bound of $deathrate$ is $\frac{1}{|V|}$ and the upper bound is 1 (all the people are involved in the cascade). Above all, the final complexity is $O(t) + O(n) + O(n \log_{1+\epsilon}(|V|)) = O(t + n \log_{1+\epsilon}(|V|))$ for each cascade (with n users and t requests). If we put this algorithm into an online environment, the complexity will be $O(T + N \log_{1+\epsilon}(|V|)) \sim O(T)$ for all cascades with N users in total[4] (we see $\log_{1+\epsilon}(|V|)$ as a constant with respect to T and $N \sim T$ as the number of users involved in cascades increases over time). □

6.2 Arbitrary-time-query sampling model

In arbitrary-time-query prediction, the prediction time t_e continuously changes in the prediction task. An example is to predict the size of the cascade in one hour from anytime when we make a prediction. Therefore, the final death rate for every sub-cascade will change over different prediction times, such that the sampling idea proposed in the Fixed-Time-Query Sampling Model cannot be straightforwardly transferred into Arbitrary-Time-Query Sampling Model.

To avoid the irregularity caused by different prediction times, the final death rate term to calculate the size of the sub-cascade at a specific time point is ignored in contrast to the first sampling stage described in the previous method. Instead, for every estimation time in the first sampling stage, we initiate a second-level sampling stage to present the future sub-cascading process as follows.

1. For one sub-cascade generated by u_i, when we need to present the sub-cascading process at time t_0, we choose t_0 as the first sampling point.
2. Once the latest sampling time point t_k of $S(u_i)$ is determined, we choose $t_{k+1} = S_u^{-1}\big(1 - (1 + \epsilon_2) \cdot deathrate_{u_i}(t_k)\big)$ as the newer sampling time point unless $(1 + \epsilon_2) \cdot deathrate_{u_i}(t_k)$ is no longer lower than 1.

It can be proved that the relative error rate caused by second-level sampling will be at most ϵ_2, while the size of the second-level sampling time point set for every sub-cascade will be $O\left(\log_{1+\epsilon_2}(|V|)\right)$ since the minimum value of $deathrate$ at the time point is $\frac{1}{|V|}$ (i.e., at least one user participated in the sub-cascade) and the maximum value is 1 (i.e., all the people are involved in the cascade).

[4] It will be counted multiple times if a specific user is involved in multiple cascades.

Algorithm 2 Fixed-Time-Query Sampling Model

```
Input:
    survival functions of users $S_{u_j}(t)$, and set of users $U$ in one cascade $C$(given dynamically);
Output:
    for every size prediction request to $t_e$ at $t_0$, output $size\left(C_{t_e}\right)$;
1:  $sum = 0$;
2:  $t = \emptyset$;
3:  $replynum = \emptyset$;
4:  $app = \emptyset$;
5:  $deathrate = \emptyset$;
6:  while request = model.acceptRequest do
7:      switch request.type do
8:          case APPROXIMATION
9:              return $size\left(C_{t_e}\right) = sum$
10:         case INVOLVED_USER
11:             INVOLVE(request)
12:         case THRESHOLD_CHANGE
13:             THRESCHANGE(request)
14: end while
15: function INVOLVE(request)
16:     $u_i$=request.user, $t_0$=request.time
17:     CREATESUBCASCADEPROCESS($u_i$, $t_0$, $t_e$)
18:     if $u_i$ is root node then
19:         $sum = 1$;
20:     else
21:         $t_{rep} = t_0 - t\left(rp(u_i)\right)$;
22:         $replynum(rp(u_i)) = replynum(rp(u_i)) + 1$;
23:         $sum = sum - app(rp(u_i))$;
24:         $deathrate(rp(u_i)) = \max\left(\frac{1}{|V|}, 1 - S_{rp(u_i)}(t_{rep})\right)$;
25:         $app(rp(u_i)) = \frac{replynum(rp(u_i)) \cdot fdrate(rp(u_i))}{deathrate(rp(u_i))}$;
26:         $sum = sum + app(rp(u_i))$;
27:         if $(1 + \epsilon_1) \cdot deathrate(rp(u_i)) \leq 1$ then
28:             $t_{new} = S_{rp(u_i)}^{-1}(1 - (1 + \epsilon) \cdot deathrate(rp(u_i)))$
29:                     $+t(rp(u_i))$;
30:             sendRequest(THRESHOLD_CHANGE,$rp(u_i)$,$t_{new}$);
31:         end if
32:     end if
33: end function
34: function CREATESUBCASCADEPROCESS(user, time, ptime)
35:     $t(user) = time$
36:     $app(user) = 0$
37:     $replynum(user) = 0$
38:     $fdrate(user) = \max\left(\frac{1}{|V|}, 1 - S_{rp(user)}(ptime - time)\right)$;
39: end function
40: function THRESCHANGE(request)
41:     $u_i = request.user$, $t_0$=request.time
42:     $sum = sum - app(u_i)$;
43:     $deathrate(u_i) = \max\left(\frac{1}{|V|}, 1 - S_{u_i}(t_0 - t(u_i))\right)$;
44:     $t_{new} = S_{u_i}^{-1}(1 - (1 + \epsilon) \cdot deathrate(u_i)) + t(u_i)$;
45:     sendRequest(THRESHOLD_CHANGE,$u_i$,$t_{new}$);
46:     $app(u_i) = \frac{replynum(u_i) \cdot fdrate(u_i)}{deathrate(u_i)}$;
47:     $sum = sum + app(u_i)$;
48: end function
```

For the ease of maintaining the sub-cascading process, we choose the following representation so that we can make adjustments using BBST (balanced binary search tree), which will reduce the complexity:

1. For the first sampling time point (t_0) in the sampling set of the sub-cascade, set the corresponding value to be $\frac{replynum(u_i)}{deathrate(t_0)} \cdot deathrate(t_1)$.
2. For the succeeding time point t_k, set the corresponding value to $\frac{replynum(u_i)}{deathrate(t_0)} \cdot (deathrate(t_{k+1}) - deathrate(t_k))$, which estimates the number of users involved in the sub-cascade within the time interval $[t_k, t_{k+1}]$.

The entire flow of the Arbitrary-Time-Query Sampling Model is illustrated in Algorithm 3.

Complexity Analysis

Theorem 3 *By using an overall* $O(n \log^2 |V| \cdot \log(n \log |V|))$ *time complexity algorithm, the model can approximate the cascading process estimated by the basic model with a relative error rate within* $\max(\epsilon_1, \epsilon_2)$.

Proof For every sub-cascade, the function THRESCHANGE() will be involved at most $O(\log_{1+\epsilon_1}(|V|))$ times, since the lower bound of $deathrate$ is $\frac{1}{|V|}$ and the upper bound is 1; for each re-tweet, the function INVOLVE() will be involved exactly once. For these two functions, the most time-consuming steps are the REMOVESUBCASCADEOLDEFFECT() and ADDSUBCASCADEEFFECT() functions. We need to adjust $O(\log |V|)$ nodes in the balanced binary search tree, while the tree size is $O(n \log |V|)$. Therefore, the overall time complexity is $O(\log(|V|) \log(n \log |V|))$.

Furthermore, the predicted size may decrease within a relative error rate of ϵ_1 during the first sampling stage, and it may increase within a relative error rate of ϵ_2 during the second sampling stage. Thus, the overall relative error rate will not exceed $\max(\epsilon_1, \epsilon_2)$. □

With the streaming models, for the task of final cascade size prediction task, we only need to set the prediction time t_e to be infinite so that the $deathrate$ of all sub-cascades will be 1 when using the Fixed-Time-Query Sampling Model. For the size prediction for arbitrary time point t, we can aggregate all corresponding values of all sampling time points lower than t in all sub-cascades in the BBST by using the Arbitrary-Time-Query Sampling Model. We can also solve the outbreak time prediction and process prediction problem by directly using the Arbitrary-Time-Query Sampling Model directly. In all cases, the streaming models will be much more efficient when compared to the base model.

7 Experiments

Experiments are performed on the dataset introduced in Sect. 3.1. Since the proposed method can be applied in a broad range of application scenarios, to evaluate the performances and fully demonstrate the advantages of the proposed methods, we conduct the experiments in three steps. First, we test our method on the size prediction problem, to calculate the final size of the cascade. Second, we test our method on the outbreak time prediction problem to find out when an outbreak cascade will reach the outbreak threshold. Finally, we test for our ultimate goal—the cascading process prediction problem—to predict the future growth of the cascade.

We also test the running speed of all the methods and find that the sampling models proposed in Sect. 6 are far more efficient than other methods under different kinds of problem

Algorithm 3 Arbitrary-Time-Query Sampling Model

```
Input:
    survival functions of users $S_{u_j}(t)$, and set of users $U$ in one cascade $C$(given dynamically);
Output:
    1. for every size prediction request to $t_e$ at $t_0$, output $size\left(C_{t_e}\right)$;
    2. the future cascading process when needed;
1: BalancedBinarySearchTree $tree.initialized()$;
2: TimeSamplingSets $tsss = \emptyset$;
3: $replynum = \emptyset$;
4: $deathrate = \emptyset$;
5: while request = model.acceptRequest do
6:     switch request.type do
7:         case APPROXIMATION
8:             return APPROX(request)
9:         case CASCADINGPROCESS
10:            return tree
11:        case INVOLVED_USER
12:            INVOLVE(request)
13:        case THRESHOLD_CHANGE
14:            THRESCHANGE(request)
15: end while
16: function APPROX(request)
17:     $t_e$=request.prediction_time
18:     return $size\left(C_{t_e}\right) = tree.allvalueKeyLowerThan(t_e) + 1$
19: end function
20: function INVOLVE(request)
21:     $u_i$=request.user, $t_0$=request.time
22:     CREATESUBCASCADEPROCESS($u_i$, $t_0$)
23:     if $u_i$ is not root node then
24:         $t_{rep} = t_0 - t\left(rp(u_i)\right)$;
25:         $replynum(rp(u_i)) = replynum(rp(u_i)) + 1$;
26:         REMOVESUBCASCADEOLDEFFECT($rp(u_i)$)
27:         ADDSUBCASCADEEFFECT($rp(u_i)$, $t_0$)
28:         if $(1 + \epsilon_1) \cdot deathrate(rp(u_i)) \leq 1$ then
29:             $t_{new} = S^{-1}_{rp(u_i)}(1 - (1 + \epsilon_1) \cdot deathrate(rp(u_i)))$
30:                 $+t(rp(u_i))$;
31:             sendRequest(THRESHOLD_CHANGE,$rp(u_i)$,$t_{new}$);
32:         end if
33:     end if
34: end function
35: function CREATESUBCASCADEPROCESS(user, time)
36:     $t(user) = time$
37:     $app(user) = 0$
38:     $replynum(user) = 0$
39:     $TimeSamplingSettssu = \emptyset$;
40:     $tsss(user) = tssu$;
41: end function
42: function THRESCHANGE(request)
43:     $u_i = request.user$, $t_0$=request.time
44:     REMOVESUBCASCADEOLDEFFECT($u_i$)
45:     ADDSUBCASCADEEFFECT($u_i$)$t_0$
46:     $t_{new} = S^{-1}_{u_i}(1 - (1 + \epsilon_1) \cdot deathrate(u_i))$
47:             $+t(u_i)$;
48:     sendRequest(THRESHOLD_CHANGE,$rp(u_i)$,$t_{new}$);
49: end function
```

```
50: function removeSubcascadeOldEffect(user)
51:     for all $elem \in tsss(user)$ do
52:         tree.remove(elem.Key, elem.Value);
53:     end for
54:     $tsss(user) = \emptyset$;
55: end function
56: function addSubcascadeEffect(user, time)
57:     $ds = dr = \max\left(\frac{1}{|V|}, 1 - S_{user}(time - t(user))\right)$;
58:     TimeSamplingSet $tssu = \emptyset$;
59:     $elem = \emptyset$
60:     $elem.Key = time$
61:     $elem.Value = replynum(user) \cdot (1 + \epsilon_2)$;
62:     $tssu.pushback(elem)$;
63:     $dr = dr * (1 + \epsilon_2)$;
64:     while $dr \leq 1$ do
65:         $do = dr$;
66:         $dr = dr * (1 + \epsilon_2)$;
67:         $t_{new} = S^{-1}_{rp(u_i)}(1 - dr)$;
68:         $elem.Key = t_{new}$
69:         $elem.Value = \frac{replynum(user)}{ds} \cdot (dr - do)$;
70:         $tssu.pushback(elem)$;
71:     end while
72:     for all $elem \in tsss(user)$ do
73:         tree.add(elem.Key, elem.Value);
74:     end for
75: end function
```

settings (the Fixed-Time-Query Sampling Model for the final cascade size prediction problem and the Arbitrary-Time-Query Sampling Model for the cascading process prediction problem). Furthermore, we will provide additional insights about the proposed method.

7.1 Baselines and evaluation metrics

Since we are the first to investigate the cascading process prediction problem, no previous models can be adopted as direct baselines. Hence, we implemented the following methods which can be potentially applied to our targeted problem as baselines:

- Cox Proportional Hazard Regression Model (Cox): This model assumes that the behavioral dynamics of all users have different scale parameters while sharing the same shape parameter. We use the same covariates as in our model and find the optimal scale parameters for all users and the shared shape parameter. We implement it as in [5].
- Exponential/Rayleigh Proportional Hazard Regression Model (Exponential/Rayleigh): Since the shape parameters of both Exponential and Rayleigh distributions are fixed values (1 for an Exponential distribution and 2 for a Rayleigh distribution), they are two special cases of the Cox model.
- Log-linear Regression Model (Log-linear): We refer to [3] which extracted 4 class features to characterize cascades, including node features, structural features of cascades, temporal features, and content features. In our case, we ignore the content features which are not covered in our dataset and also reported by [3] to be unimportant for cascade prediction. Then, we use a log-linear regression model to predict the cascade size.

It is noted that log-linear can only predict the cascade size and is not appropriate for time-related predictions, while Cox, Exponential and Rayleigh models are applied to all prediction

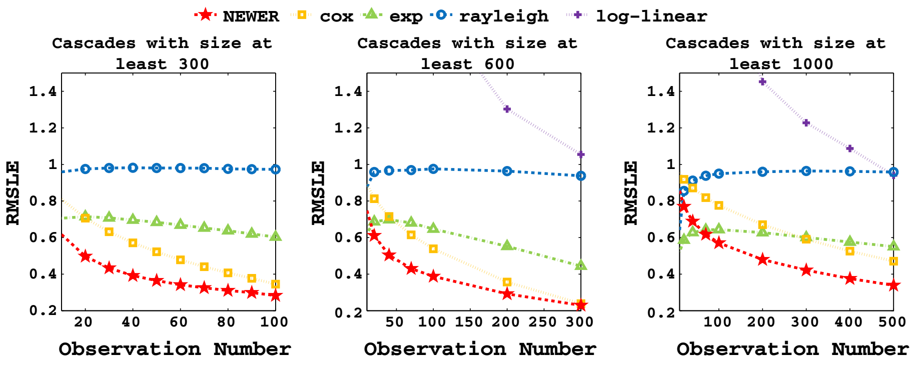


Fig. 7 *RMSLE* results for different methods with different numbers of observed nodes in cascades. We excluded the log-linear results in the first sub-figure since their *RMSLE* values all exceeded 1.5

tasks. Furthermore, the goals of the Cox, Exponential and Rayleigh models are to elucidate the behavioral dynamics. Then, we use the same cascade prediction model as in our method to conduct cascade-level predictions.

For each cascade, our dataset includes its complete cascading process as the ground truth. In Tencent Weibo, the activities of users are highly affected by daily cycle. They are highly active in daytime and quieter at midnight. Therefore, it is better to quantify the dynamics by event time instead of real time. The average number of events that occurred in the data per second is about 427, from which we can map the event time into real time.

Next, we use the following metrics to evaluate the performances.

- Root Mean Square Log Error (*RMSLE*): It is not appropriate to use standard RMSE to evaluate the prediction accuracy for power-law distributed data. For example, for a cascade with a ground truth size of 1000, it is significantly different to predict its size to be 2000 or 0, but these results give the same RMSE. Thus, we first calculate the logarithmic results for both the ground truth and the predicted value and then calculate RMSE for the logarithmic results to evaluate the accuracy of the proposed method and the baselines. The lower the RMSLE, the better the result.
- Precision with σ-Tolerance ($\Delta\sigma$-Precision): In real applications, a small deviation from the ground truth value is often acceptable. In our case, we regard a predicted value within the range of $(1+\sigma)^{\pm 1} groundtruth$ as tolerable, and the resulting precision is $\Delta\sigma$-Precision. The larger the value of $\Delta\sigma$-Precision, the better the result.

For parameter setting, there are 4 parameters in our method, including μ, η, α_β, and α_γ. We tune these parameters using grid searching, and the optimal parameters used in our experiments are $\mu = 10$, $\eta = 10$, $\alpha_\beta = 6 \times 10^{-5}$, $\alpha_\gamma = 8 \times 10^{-6}$.

7.2 Cascade size prediction

We randomly separate the cascades into 10 folds and conduct a 10-fold cross-validation by using 9 of them as training data and the other one as testing data. For cascades with sizes over k, we use the first $s(s < k)$ nodes as observed data, and the target is to predict the final cascade sizes.

The prediction performances for all the methods are shown in Fig. 7. It can be seen that the proposed method NEWER significantly outperforms other baselines in terms of *RMSLE* value for different sized datasets. When we set the observation number to 50, the RMSLE value of

Table 3 Running time of final cascade size prediction for different methods in different datasets using a server with 3.4 GHz Quad Core Intel i7-3770 CPU and 16 GB of memory

Method	Base model	Fixed-Time-Query Sampling Model ($\delta = 0.1$) (s)	Directed learning method (s)
Size ≥20	8.47×10^5s	10.73	899
Size ≥50	7.61×10^5s	8.62	899
Size ≥100	6.65×10^5s	7.09	898
Size ≥500	4.35×10^5s	4.33	891
Size ≥1000	3.4×10^5s	3.30	881

the NEWER model is 0.36 for all cascades with size over 300, while the RMSLE value for all the other methods is at least 0.53. The baseline with closest performance to NEWER is the Cox model. We can see that the margins of improvement from Cox to NEWER are more obvious in the dataset with a larger k. In a given dataset, the margins are more evident with a smaller s. These results demonstrate the significant advantage of NEWER in predicting large cascades at a very early stage.

Comparatively, the log-linear method does not achieve satisfactory results in this task. The main reason is that the coefficients in the log-linear model are highly biased toward the dominant number of small-sized cascades, which is also argued by [3]. In our method, we successfully overcome this bias by shifting from macrocascade-level features to microbehavioral dynamics. The substantial gain achieved by all behavioral dynamics-based methods (including NEWER, Cox, Exponential and Rayleigh) exemplifies the importance of this micromechanism for cascade prediction.

To demonstrate the efficiency of the proposed method, we also evaluate the computational cost of NEWER and Sampling-NEWER in a computational environment with 3.4 GHz Quad Core Intel i7-3770 and 16 GB of memory. We track the process of all cascades. The base cascade prediction model (Base) re-predicts the final size at every time point (in seconds), while the Fixed-Time-Query sampling model only re-predicts the final size when the observed sub-cascade sizes increase or reach the threshold change of the sub-cascade. As shown in Table 3, the sampling model (with a 10 percent performance degradation tolerance) is much more efficient than base model by almost 5 magnitudes.

7.3 Outbreak time prediction

Another interesting problem is to predict when a cascading outbreak will happen. For example, in the early stage of a cascade, can we predict when the cascade will reach a specific size? Without loss of generality, we set the outbreak size threshold to as 1000. We evaluate the prediction performance with different numbers of observed nodes in the cascades. As shown in Fig. 8, the NEWER model obtains the best performances for both *RMSLE* and $\Delta\sigma$-Precision metrics, and the average improvements in the early stage is approximately 70 %. Although the Exponential and Rayleigh models report better results than NEWER in the very early stage (fewer than 50 observation nodes), improvements in their performance with an increasing number of observed nodes are not as significant as for NEWER.

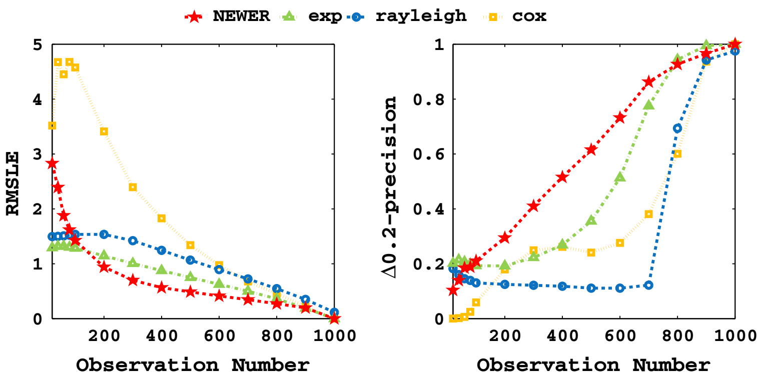
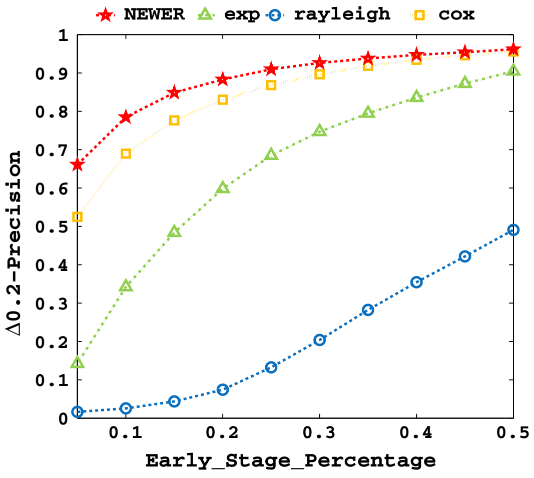


Fig. 8 Outbreak time prediction results for different methods with different number of observed nodes in cascades

Fig. 9 Cascading process prediction accuracy of different methods under different early-stage percentage settings



7.4 Cascading process prediction

The ultimate purpose of this paper is to predict the cascading process. For each cascade, we use Δt to represent the early stage window and $\hat{t}$ to represent its end time. Then, we use the cascade information during $[0, \Delta t]$ to predict the cascading process during $[\Delta t, \hat{t}]$. At any time $t \in [\Delta t, \hat{t}]$, we calculate whether the predicted cascade size at t is within the σ tolerance of the ground truth size at t. Then, we calculate the $\Delta\sigma$-Precision by integrating t to describe the prediction accuracy for this cascading process. Finally, we average the $\Delta\sigma$-Precision for all cascades and show the results in Fig. 9. Here, we vary the early-stage percentage (i.e., $\Delta t/\hat{t}$) from 0 to 50 % and discover that in all the settings of early-stage percentage, NEWER achieves the best performances in cascading process prediction. Moreover, the advantage of NEWER is clearer for lower early-stage percentages. When we set the early stage to be 15 % of the whole cascade duration, we can obtain a $\Delta 0.2$-Precision of 0.849, signifying the correct prediction of cascade sizes at 84.9 % time points, which indicates that the cascading process is predictable and that the proposed method is adequate and superior in cascading process prediction. Furthermore, changing the precision tolerance value σ will not affect the

Table 4 Computational numbers of cascade process prediction for different methods

Size	Base model	Arbitrary-Time-Query Sampling Model ($\epsilon_1 = 0.1$ and $\epsilon_2 = 0.1$)
20	1.4×10^{12}	4.2×10^{6}
50	3.5×10^{12}	7.6×10^{6}
100	6.9×10^{12}	1.4×10^{7}
500	3.5×10^{13}	3.4×10^{7}
Around 1000	6.9×10^{13}	4.2×10^{7}

relative results of all the methods in our experiments, and the precision value will be smaller when setting a smaller value of σ. For abbreviation, we only report the results of $\sigma = 0.2$, which is a reasonable tolerance for most application scenarios.

We also make a comparison on the cascading process prediction efficiency between the base model and the Arbitrary-Time-Query Sampling Model. We randomly select 10 cascades with fixed sizes, trace the cascading processes over one day, and let the model predict the one-day cascading processes dynamically. The base cascade prediction model (Base) re-predicts the size of every time point in the cascade (in seconds), while the Arbitrary-Time-Query Sampling Model only re-predicts the cascading process when the observed sub-cascade sizes increase or reach the first-level sampling time point of the sub-cascades. We evaluate the average computation number for both methods, with the results being shown in Table 4. It is obvious that the Arbitrary-Time-Query Sampling Model reduces the calculation number by several magnitudes, and this advantage becomes more obvious as the cascade size increases.

7.5 Out-of-sample prediction

In real applications, the interaction information between nodes is not always available, which means that some nodes' behavioral dynamics cannot be directly derived by maximum likelihood estimation from data. We call these nodes *out-of-sample nodes*. This is the main reason why we propose NEWER to incorporate the covariates of behavioral dynamics. To evaluate the performance of NEWER in handling such case, we simulate the scenario by hiding the interaction information of randomly selected 10 % of users as out-of-sample users and then predict the final sizes of the cascades that these users were involved at an early stage.

In the Cox model, the scale parameters in behavioral dynamics of out-of-sample users can be regressed by the covariates. For the shape parameter, we calculate the average value of shape parameters in observed users and apply this value to the shape parameters of out-of-sample users. In the NEWER model, both of shape and scale parameters can be regressed by covariates with the learned β and γ. We also employ the standard Weibull Regression (Wbl) as a baseline, which can be derived by simply setting μ and η to be 0 in Eq. 10. Then, we use the averaged shape and scale parameters of observed users as the parameters of out-of-sample users.

As shown in Fig. 10, the NEWER model can significantly and consistently outperform the Cox and Wbl models in out-of-sample prediction, which demonstrates that the discovered covariates from behavioral features of a user's networked neighbors can effectively be used to predict the user's behavioral dynamics. Furthermore, we visualize the regression coefficients β and γ in Fig. 11. It can be observed that the behavioral features of a user's followers play more important roles in predicting both scale and shape parameters for the user, while the user's structural features are less important.

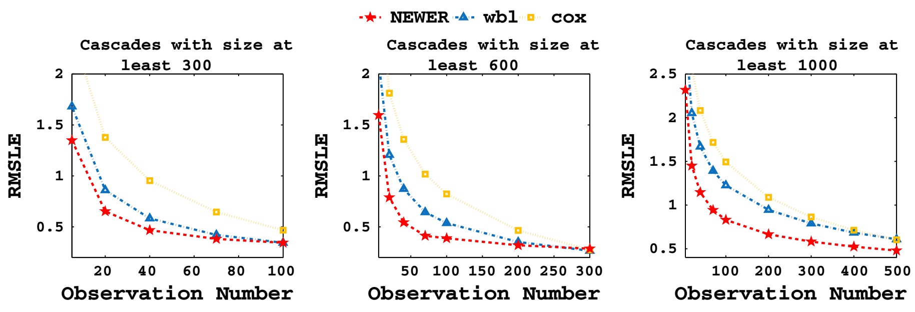


Fig. 10 Prediction result with unknown users

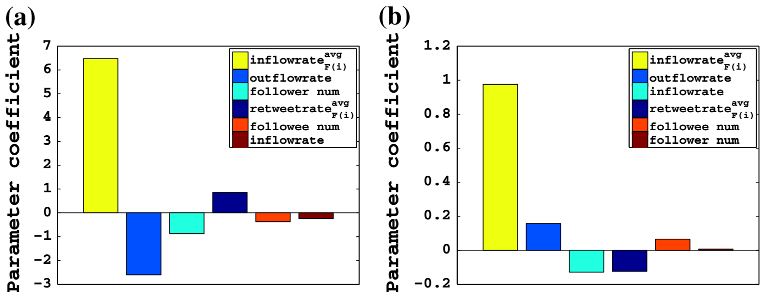


Fig. 11 Parameter coeffiecients for **a** scale parameter and **b** shape parameter

8 Conclusions and future work

In this paper, we raise an important and interesting question: Beyond predicting the final size of a cascade, is it possible to predict the entire cascading process based on early-stage information of cascades? To address this problem, we propose to uncover and predict the macrocascading process via microbehavioral dynamics. Through data-driven analysis, we find out the common principles and important patterns in the behavioral dynamics and propose a novel NEWER model for behavioral dynamics modeling with good interpretability and generalizability. After that, we propose a scalable method to aggregate microbehavioral dynamics into macrocascading processes. Extensive experiments on a large-scale real dataset demonstrate that the proposed method achieves the best results in various cascading prediction tasks, including cascade size prediction, outbreak time prediction, and cascading process prediction.

There are still a few limitations on the proposed method. First, although the overall performance is very well, our method cannot give the confidence value of the prediction. Hence, we cannot tell how confident we are for a specific prediction task. Second, our method can make good prediction results for most natural information cascading processes. However, the prediction results are not very well for some abnormal cascading processes, or the cascading processes generated by some specific patterns (by using machines, zombie followers, etc.). Both of these are very interesting and we will try to solve them in our future work.

Acknowledgments This work was supported by National Program on Key Basic Research Project, No. 2015CB352300; National Natural Science Foundation of China, Nos. 61370022, 61531006, 61472444 and 61210008; and International Science and Technology Cooperation Program of China, No. 2013DFG12870.

Thanks for the research fund of Tsinghua-Tencent Joint Laboratory for Internet Innovation Technology, and Beijing Key Laboratory of Networked Multimedia.

Appendix: Proof of Theorem 1

Proof It is evident that both $G_2(\beta, \lambda)$ and $G_3(\gamma, k)$ have global minimum value. Next we prove that $G_1(\lambda, k)$ also has global minimum value, or to prove $\log L(\lambda, k)$ has global maximum value.

Let $\lambda_i' = \lambda_i^{-k_i}$, $\log L'(\lambda', k) = \log L(\lambda, k) = \sum_{i=1}^{N} l_i'(\lambda_i', k_i)$ where $l_i'(\lambda_i', k_i) = m_i \log k_i + (k_i - 1)\sum_{j=1}^{m_i} \log T_{i,j} + m_i \log \lambda_i' - \lambda_i' \sum_{j=1}^{m_i} T_{i,j}^{k_i}$, the partial derivatives of the l_i' are given by:

$$\frac{\partial l_i'}{\partial \lambda_i'} = \frac{m_i}{\lambda_i'} - \sum_{j=1}^{m_i} T_{i,j}^{k_i}, \quad \frac{\partial^2 l_i'}{\partial \lambda_i'^2} = -\frac{m_i}{\lambda_i'^2} < 0 \tag{15}$$

$$\frac{\partial l_i'}{\partial k_i} = \frac{m_i}{k_i} + \sum_{j=1}^{m_i} \log T_{i,j} - \lambda_i' \sum_{j=1}^{m_i} T_{i,j}^{k_i} \log T_{i,j} \tag{16}$$

$$\frac{\partial^2 l_i'}{\partial k_i^2} = -\frac{m_i}{k_i^2} - \lambda_i' \sum_{j=1}^{m_i} T_{i,j}^{k_i} \left(\log T_{i,j}\right)^2 < 0 \tag{17}$$

Since $\frac{\partial^2 l_i'}{\partial \lambda_i'^2} < 0$ and $\frac{\partial^2 l_i'}{\partial k_i^2} < 0$, the conditional marginal posterior densities of parameters λ_i' and k_i are log-concave. Moreover, when $0 < k_i < 1, 0 < \lambda_i' < \min\left(\frac{m_i}{\sum_{i=1}^{m_i} T_{i,j}^{k_i}}, \frac{1}{\sum_{j=1}^{m_i} T_{i,j} \log T_i}\right)$,

$$\frac{\partial l_i'}{\partial \lambda_i'} = \frac{m_i}{\lambda_i'} - \sum_{j=1}^{m_i} T_{i,j}^{k_i} \geq \frac{m_i}{\frac{m_i}{\sum_{i=1}^{m_i} T_{i,j}^{k_i}}} - \sum_{j=1} m_i T_{i,j}^{k_i} = 0$$

$$\begin{aligned}\frac{\partial l_i'}{\partial k_i} &= \frac{m_i}{k_i} + \sum_{j=1}^{m_i} \log T_{i,j} - \lambda_i' \sum_{j=1}^{m_i} T_{i,j}^{k_i} \log T_{i,j} \\ &\geq \frac{m_i}{k_i} + \sum_{j=1}^{m_i} \log T_{i,j} - \frac{\sum_{j=1}^{m_i} T_{i,j}^{k_i} \log T_{i,j}}{\sum_{j=1}^{m_i} T_{i,j} \log T_i} \geq m_i + \sum_{j=1}^{m_i} \log T_{i,j} - 1 > 0\end{aligned} \tag{18}$$

when $k_i \geq \max\left(1, \frac{m_i}{\lambda_i' \sum_{j=1}^{m_i} T_{i,j} \log T_{i,j} - \sum_{j=1}^{m_i} \log T_{i,j}}\right)$ and $\lambda_i' \geq \max\left(1, \frac{m_i}{\sum_{j=1}^{m_i} T_{i,j}^{k_i}}\right)$,

$$\frac{\partial l_i'}{\partial \lambda_i'} = \frac{m_i}{\lambda_i'} - \sum_{j=1}^{m_i} T_{i,j}^{k_i} \leq \frac{m_i}{\frac{m_i}{\sum_{i=1}^{m_i} T_{i,j}^{k_i}}} - \sum_{j=1} m_i T_{i,j}^{k_i} = 0 \tag{19}$$

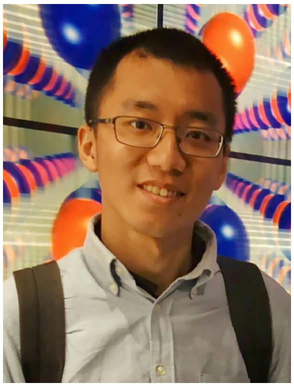
$$\begin{aligned}\frac{\partial l_i'}{\partial k_i} &= \frac{m_i}{k_i} + \sum_{j=1}^{m_i} \log T_{i,j} - \lambda_i' \sum_{j=1}^{m_i} T_{i,j}^{k_i} \log T_{i,j} \\ &\leq \frac{m_i}{\frac{m_i}{\lambda_i' \sum_{j=1}^{m_i} T_{i,j} \log T_{i,j} - \sum_{j=1}^{m_i} \log T_{i,j}}} + \sum_{j=1}^{m_i} \log T_{i,j} - \lambda_i' \sum_{j=1}^{m_i} T_{i,j}^{k_i} \log T_{i,j}\end{aligned}$$

$$= \lambda_i' \sum_{j=1}^{m_i} T_{i,j} \log T_{i,j} - \sum_{j=1}^{m_i} \log T_{i,j} + \sum_{j=1}^{m_i} \log T_{i,j} - \lambda_i' \sum_{j=1}^{m_i} T_{i,j}^{k_i} \log T_{i,j}$$
$$< 0 \tag{20}$$

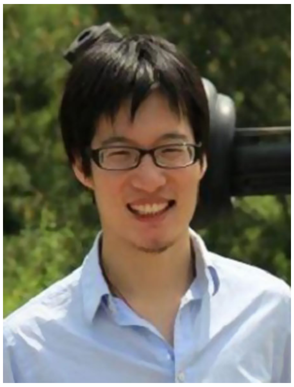
which means there should be a global maximum of l_i', so does $\log L$. □

References

1. Chen W, Collins A, Cummings R, Ke T, Liu Z, Rincon D, Sun X, Wang Y, Wei W, Yuan Y (2011) Influence maximization in social networks when negative opinions may emerge and propagate. In: SDM, vol. 11, pp 379–390. SIAM
2. Chen W, Wang Y, Yang S (2009) Efficient influence maximization in social networks. In: Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 199–208
3. Cheng J, Adamic L, Dow PA, Kleinberg JM, Leskovec J (2014) Can cascades be predicted? In: Proceedings of the 23rd international conference on World wide web. International World Wide Web Conferences Steering Committee, pp 925–936
4. Cohen R, Havlin S, Ben-Avraham D (2003) Efficient immunization strategies for computer networks and populations. Phys Rev Lett 91(24):247901
5. Cox D (1972) Regression models and life-tables. J R Stat Soc Ser B 34(2):187–220
6. Cui P, Jin S, Yu L, Wang F, Zhu W, Yang S (2013) Cascading outbreak prediction in networks: a data-driven approach. In: Proceedings of the 19th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 901–909
7. Cui P, Wang F, Liu S, Ou M, Yang S, Sun L (2011) Who should share what?: item-level social influence prediction for users and posts ranking. In: Proceedings of the 34th international ACM SIGIR conference on research and development in information retrieval, ACM, pp 185–194
8. Domingos P, Richardson M (2001) Mining the network value of customers. In: Proceedings of the seventh ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 57–66
9. Du N, Song L, Woo H, Zha H (2013) Uncover topic-sensitive information diffusion networks. In: Proceedings of the sixteenth international conference on artificial intelligence and statistics, pp 229–237
10. Gionis A, Terzi E, Tsaparas P (2013) Opinion maximization in social networks. arXiv preprint arXiv:1301.7455
11. Gomez-Rodriguez M, Leskovec J, Schölkopf B (2013) Modeling information propagation with survival theory. In: ICML
12. Gomez Rodriguez M, Leskovec J, Schölkopf B (2013) Structure and dynamics of information pathways in online media. In: Proceedings of the sixth ACM international conference on web search and data mining. ACM, pp 23–32
13. Kempe D, Kleinberg J, Tardos É (2003) Maximizing the spread of influence through a social network. In: Proceedings of the ninth ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 137–146
14. Miller RG Jr (2011) Survival analysis, vol 66. Wiley, London
15. Myers S, Leskovec J (2010) On the convexity of latent social network inference. In: Advances in neural information processing systems, pp 1741–1749
16. Pinder III JE, Wiener JG, Smith MH (1978) The weibull distribution: a new method of summarizing survivorship data. Ecology 59(1):175–179. doi:10.2307/1936645
17. Rodriguez G (2005) Parametric survival models. Lectures Notes, Princeton University
18. Rodriguez MG, Balduzzi D, Schölkopf B (2011) Uncovering the temporal dynamics of diffusion networks. arXiv preprint arXiv:1105.0697
19. Rodriguez MG, Schölkopf B (2012) Influence maximization in continuous time diffusion networks. arXiv preprint arXiv:1205.1682
20. Tibshirani R (1996) Regression shrinkage and selection via the lasso. J R Stat Soc Ser B 58(1):267–288. http://www.jstor.org/stable/2346178
21. Yu L, Cui P, Wang F, Song C, Yang S (2015) From micro to macro: uncovering and predicting information cascading process with behavioral dynamics. In: Data mining (ICDM), 2015 IEEE international conference on, pp 559–568. IEEE

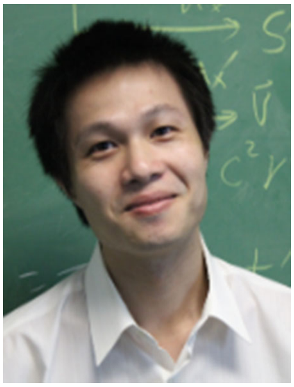


Linyun Yu received the bachelor's degree from Special Pilot CS Class (also known as Yao class) at Tsinghua University in 2012. He is pursuing the Ph.D. degree in the Department of Computer Science and Technology of Tsinghua University. His main research interests include data mining, social computing, and analysis of human dynamics and interactions. Currently, he is sponsored by Tsinghua Scholarship for Overseas Graduate Studies to visit University of Miami, USA, from March 2016 to August 2016.



Peng Cui is an Assistant Professor at Tsinghua University. He received his Ph.D. Degree from Tsinghua University in 2010. He has vast research interests in social network analysis and multimedia analysis. He has published more than 60 papers in conferences such as KDD, ACM Multimedia, SIGIR, and journals such as IEEE TKDE and ACM TKDD. He won 5 best paper awards in recent 4 years, including ICDM2015 Best Student Paper Award, ICME 2014 Best Paper Award, etc. In 2015, he was awarded as ACM China Rising Star.

Fei Wang got his Ph.D. from Tsinghua University in 2008. His research interests are mainly data mining, machine learning algorithms and their applications in social and health informatics.



Chaoming Song is an Assistant Professor in physics at University of Miami. He received his Ph.D. degree in physics from the City University of New York (CUNY) in 2008. He published over 50 papers that received more than 5000 citations and has been awarded with Erds-Renyi Prize in 2015 for his network science research. Currently he is serving as an editorial board member of Scientific Reports. Songs research interests lie in the intersection of statistical physics, network science, biological science, and computational social science, broadly exploring patterns behind petabytes of data. One of his recent activities in this area is aiming to understand the fundamental properties of human mobility and interactions at various scales. He is actively contributing to the network science area an interdisciplinary field studying complex interactions that aims to connect phenomena emerged in different fields into a universal description.

Shiqiang Yang received the B.E. and M.E. degrees from the Department of Computer Science and Technology of Tsinghua University in 1977 and 1983, respectively. He is now a Professor at Tsinghua University and a senior member of IEEE. His research interests include multimedia technology and systems, video compression and streaming, content-based retrieval for multimedia information, social network, and social media analysis. He has published more than 200 papers and got several best paper award including IEEE T-CSVT, ACM MM, ICDM, ICME, and MMM.