Formatting Instructions for Authors Using LaTeX

AAAI Press
Association for the Advancement of Artificial Intelligence
2275 East Bayshore Road, Suite 160
Palo Alto, California 94303

Abstract

AAAI creates proceedings, working notes, and technical reports directly from electronic source furnished by the authors. To ensure that all papers in the publication have a uniform appearance, authors must adhere to the following instructions.

Introduction

Understanding the processes and dynamics of *information diffusion* through networks plays a fundamental role in a variety of domains, such as evaluating the effects of networks in marketing (Domingos and Richardson 2001; Kempe, Kleinberg, and Tardos 2003; Leskovec, Adamic, and Huberman 2007), monitoring the spread of news, opinions, and scientific ideas via citation networks (Adar et al. 2004; Gruhl et al. 2004; Leskovec, Kleinberg, and Faloutsos 2005), and detecting the spread of erroneous information (Dong, Berti-Equille, and Srivastava 2009).

In practical applications, the underlying diffusion network (e.g. networks on who influenced whom) is often hidden. Therefore, many interesting models have been developed to automatically infer diffusion networks from cascade observations, i.e., timestamps when users post a blog on certain topics or purchase products (Gomez-Rodriguez, Leskovec, and Krause 2012; Gomez-Rodriguez, Balduzzi, and Schölkopf 2011; Yang and Zha 2013; Zhou, Zha, and Song 2013; Wang et al. 2014; Daneshmand et al. 2014; Du et al. 2012; 2013).

To be more specific, the diffusion network is usually modeled as a matrix $A = \{\alpha_{u,v}\}$, where $\alpha_{u,v}$ is a parameter as the strength of influence user u has on v. Depending on the diffusion model, $\alpha_{u,v}$ can stand for the parameter of the delay distribution for information to propagate from u to v or the probability that v publish a related post given that u has published a post previously. The *Network Inference* problem focuses on discovering the all parameters $\alpha_{u,v}$ from the observed cascades.

Most previous work on network inference assume that the strength of influence $\alpha_{u,v}$ between u and v remains the same during the whole life of the diffusion processes. We can easily see that this assumption is unrealistic. Consider the propagation of a story in microblog as an example. It is more likely for a user to be influenced by friends to talk about the story when it is popular due to its freshness in the beginning of the propagation. On the contrary, when the story becomes stale and less popular towards the end of its life, it is less likely that one can influence her friends to have interest in it.

Ignoring the heterogeneity of influence strength in different stages can lead to unsatisfactory solution of the Network Inference problem. Assume that we observe user v posts the same contents immediately after u. Existing Network Inference algorithm will treat it as a strong evidence that user u has strong influence on v. However, it is also possible that v reposes fast simply because the story is very popular and any of her friends' post will lead to similar rapid response. On the other hand, if the above scenario occurs when the story is not no longer popular, v's rapid response to u's post can be safely treated as strong evidence that u is very influential on v. The confusion between strong influence between friends and the popularity of the story leads to the failure in this scenario.

In this work, we propose to incorporate the heterogeneity of influence strength in different life stage of diffusion process to improve the accuracy of network inference. We design a heuristic referred to as *Life Stage Heuristics* (LSH) that can be easily applied to any existing network inference algorithm. In our LSH, we model the strength of influence as a function of the diffusion life stage $\alpha_{u,v}(t)$ instead of a constant $_{u,v}$ in previous network inference algorithms. first approximate the popularity in different life stage by the number of activation. Then, the diffusion rate associated with each edge is modeled as the multiplication of the popularity of the information in the current stage and the strength of influence between the pair of individuals.

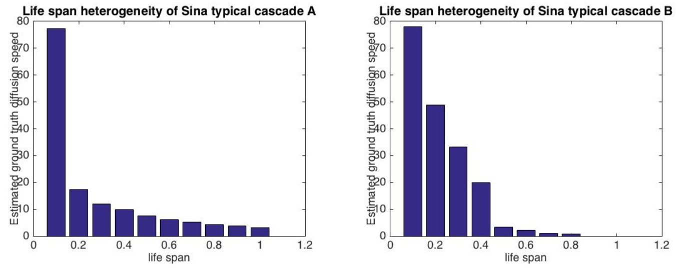
Recent work has shown that the network inference algorithms are more effective when considering the *heterogeneous* influence such as topic-dependent transmission rates (Du et al. 2013), heterogenous delay distribution (Du et al. 2012) and multi-aspects multi-pattern cascades (Wang et al. 2014). However, most previous work ignore the heterogeneity of diffusion speed with respect to the life stage of the cascade. Instead, they assume

Most previous work on network inference assumes that all propagations share the *same* diffusion network. We can easily see that this assumption is unrealistic as the influence between users depends on the topic or the entity under propagation. For example, IT analysts have more influence on others regarding the choice of smart phones, while users tend to trust personal friends as a doctor on medical advices. Moreover, recent work has shown that the network inference algorithms are more effective when considering the *heterogeneous* influence such as topic-dependent transmission rates (Du et al. 2013), heterogenous delay distribution (Du et al. 2012) and multi-aspects multi-pattern cascades (Wang et al. 2014). Inferring aspect-specific diffusion networks is a challenging problem. Naively applying existing network inference algorithms to each aspect-specific cascades independently usually results in unsatisfactory solution. For example, for a network with a hundred nodes, they usually require at least a thousand cascades to achieve accurate inference (Gomez-Rodriguez, Leskovec, and Krause 2012), while in practice it is unlikely that we can gather this many number of cascades for each topic.

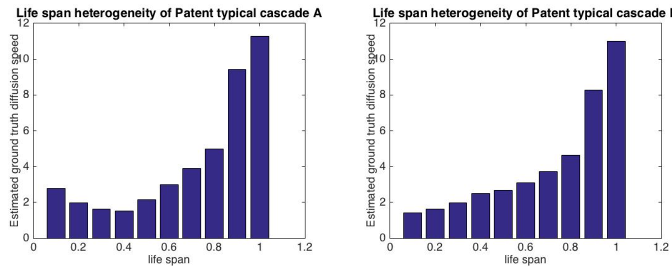
Empirical analysis

Existence of diffusion speed heterogeneity

We make an interesting analysis of message's diffusion speed on 2 real world datasets: Sina microblog and U.S. patent. We normalize the activation times with Min-Max method and divide the time length of cascade into several periods. As diffusion speed is unmeasurable, we compute the $\frac{1}{\Delta t}$ in each life span that is proportion to the ground truth diffusion speed. As in Figure 1, the maximum of $\frac{1}{\Delta t}$ reaches 80 in Sina and about 12 in Patent dataset, while the minimum approaches 0. There is distinct difference of diffusion speed in different life span. However, former algorithms have not taken this into consideration. They assume that the coefficient of Δt's distribution which controls diffusion speed does not change with time. We believe the life span heterogeneity is a key point to re-understand network inference problem.



(a) Cascades of Sina Microblog dataset



(b) Cascades of U.S. Patent dataset

Figure 1: The diffusion speed of 2 real world dataset differs a lot in different life span.

Validity of diffusion speed approximation approach

We aim to make use of diffusion speed information in network inference algorithm. But the estimation of diffusion speed requires the parent relationship in cascades which is seldom available in real world data, we would like to use the number of activated nodes as surrogate for the diffusion speed, which is:

$$S_p(t_j) = (I^{t_j+lag} - I^{t_j-lag})/I^T \tag{1}$$

t_j is the target time stamp. lag is half the length of sliding window around t_j, T is the end of every cascade. I^t refers to the number of infected nodes by time t. The value of $S_p(t_j)$ ranges from 0 to 1.

We show the validity of this approach by empirically calculating the correlation. We select 80 long cascades in Patent dataset and 50 long cascades in Sina dataset. As in Table 1, we use Evans's (1996) interpretation for r coefficient to describe the strength of correlation. 65% of Patent cascades and 90% of Sina cascades present strong or very stronger correlation. Our approach of calculating activated nodes number can reflect the level of diffusion speed to a large extent.

Table 1: Correlation coefficients between ground truth diffusion speed and the number of activated nodes in time unit.

coefficient range	strength	pct. Patent	pct. Sina
.00-.19	very weak	0.05	0.02
.20-.39	weak	0.125	0.02
.40-.59	moderate	0.175	0.06
.60-.79	strong	0.3	0.08
.80-1.0	very strong	0.35	0.82

Proposed algorithm

In this section, we'll introduce (C)LSH method that aims to improve inferring performance by taking advantage of information about messages' diffusion speed. Note that our methods don't ask for any extra input data other than traditional cascades set. We'll show that our method can be easily incorporated into existing network inference algorithms without adding any computational overhead. Some procedures for optimizing our method will be discussed at the end of this section.

Data and definitions

Incorporation with existing algorithms

We incorporate our method into 3 existing algorithms: NetRate, ConNie and NetInf. We make several changes to the original algorithms without breaking their main structure.

- LSH-NetRate
  The transmission likelihood of an activated node j infecting i depends not only on infection time delay $(t_i - t_j)$ and edge strength α_{ji}, but also on the diffusion speed of message c at time t_j. Transmission is more likely to occur with high diffusion speed, short infection interval and strong connection of edge.
  We redefine the transmission rate function by combining edge strength and diffusion speed together, denoted as:
  $$g_{ji}(t_j, \alpha_{ji}) = \alpha_{ji} * S_p(t_j); \tag{2}$$
  Subsequently, survival analysis functions are:
  - Exponential model:
  $$f(t_i|t_j, g_{ji}) = \begin{cases} g_{ji} * e^{-g_{ji}*(t_i-t_j)}, & \text{if } t_j < t_i \\ 0, & \text{otherwise} \end{cases}$$
  $$\log S(t_i|t_j, g_{ji}) = -g_{ji} * (t_i - t_j)$$
  $$H(t_i|t_j, g_{ji}) = g_{ji}$$
  - Power law model:
  $$f(t_i|t_j, g_{ji}) = \begin{cases} \frac{g_{ji}}{\delta}(\frac{t_i-t_j}{\delta})^{-1-g_{ji}}, & \text{if } t_j < t_i \\ 0, & \text{otherwise} \end{cases}$$
  $$\log S(t_i|t_j, g_{ji}) = -g_{ji} * log(\frac{t_i - t_j}{\delta})$$
  $$H(t_i|t_j, g_{ji}) = g_{ji} * \frac{1}{t_i - t_j}$$
  - Rayleigh model:
  $$f(t_i|t_j, g_{ji}) = \begin{cases} g_{ji}(t_i-t_j)e^{-\frac{1}{2}g_{ji}(t_i-t_j)^2}, & \text{if } t_j < t_i \\ 0, & \text{otherwise} \end{cases}$$
  $$\log S(t_i|t_j, g_{ji}) = -g_{ji} * \frac{(t_i - t_j)^2}{2}$$
  $$H(t_i|t_j, g_{ji}) = g_{ji} * (t_i - t_j)$$
  Note that the computation of $S_p(t_j)$ is finished in preprocessing. $S_p(t_j)$ are constants in inference process which will not affect the survival analysis process. The range of $S_p(t_j)$ is [0,1]. Constraints of the former problem have no change.
- LSH-ConNie
  ConNie declares a weighted adjacency matrix A each entry of which controls the conditional probability of edge transmission. In our method, transmission probability on an edge depends on diffusion speed of parent node's time($S_p(t_j)$) besides edge weight A_{ji}. We define the transmission rate function by combining $S_p(t_j)$ and edge weight, denoted as:
  $$g_{ji}(t_j, A_{ji}) = A_{ji} * S_p(t_j); \tag{3}$$
  When inferring the incoming edges of node i, the modified likelihood function of i is:
  $$L_i(A_{:,i}; C) = \prod_{c\in C; t_i^c<\infty} \left[1 - \prod_{j:t_j\le t_i} (1 - w_{ji} g_{ji}(t_j^c, A_{ji}))\right] * \prod_{c\in C; t_i^c=\infty} \prod_{j:t_j^c<\infty} (1 - g_{ji}(t_j^c, A_{ji})) \tag{4}$$
  We have to transform this into a convex optimization problem as the way ConNie did. Similarly, we redefine the variables B_{ji}^c and γ_c. Note that as we incorporate $S_p(t_j)$ into B_{ji}^c, the new defined variable B^C is more than just a $N * N$ matrix (N is the number of nodes in ground truth network). There are $|C|$ of $N * N$ matrix to be estimated in B^C, the size of which will change with the cascades amount.
  $$B_{ji}^c = 1 - g_{ji}(t_i^c, A_{ji}) \tag{5}$$
  $$\gamma_c = 1 - \textstyle\prod_{j:t_j\le t_i}(1 - w_{ji} * g_{ji}(t_j^c, A_{ji})) \tag{6}$$
  Our method makes no change to the constraints of B_{ji}^c and γ_c as the range of $S_p(t_j)$ is [0, 1], too. The objective problem becomes:
  $$\max_{\gamma_c, B^C(:,i)} \prod_{c\in C; t_i^c<\infty} \gamma_c * \prod_{c\in C; t_i^c=\infty} \prod_{j:t_j^c<\infty} B_{ji}^c \tag{7}$$
  $$subject \quad to:$$
  $$0 \leqslant B_{ji}^c \leqslant 1 \quad \forall j$$
  $$0 \leqslant \gamma_c \leqslant 1 \quad \forall c$$
  $$\gamma_c + \textstyle\prod_{j:t_j\le t_i} \left(1 - w_{ji} * \left(1 - B_{ji}^c\right)\right) \leqslant 1 \quad \forall c.$$
  Negative logarithm of the objective problem is shown in Eq.8, in which $\hat{\gamma_c} = \log \gamma_c$ and $\hat{B_{ji}^c} = \log B_{ji}^c$:
  $$\min_{\gamma_c, B^C(:,i)} - \sum_{c\in C; t_i^c<\infty} \hat{\gamma_c} - \sum_{c\in C; t_i^c=\infty} \sum_{j:t_j^c<\infty} \hat{B_{ji}^c} \tag{8}$$
  $$subject \quad to:$$
  $$\hat{B_{ji}^c} \leqslant 0 \quad \forall j$$
  $$\hat{\gamma_c} \leqslant 0 \quad \forall c$$
  $$\log\left[e^{\hat{\gamma_c}} + \textstyle\prod_{j:t_j\le t_i} \left(1 - w_{ji} * \left(1 - e^{\hat{B_{ji}^c}}\right)\right)\right] \leqslant 0 \quad \forall c.$$

The problem defined in Eq.8 is convex on its variables γ_c and $B^C(:,i)$. In order to transform Eq.9 into convex problem, we have got more variables which might add to the computation burden. We'd like to propose a compromise to reduce variables amount when cascades number is large. We skip the S_p property of the second part of Eq.9, which is:

$$L_i(A_{:,i};C) = \prod_{c;t_i^c<\infty} \left[1 - \prod_{j:t_j \le t_i} (1 - w_{ji} * g_{ji}(t_j^c, A_{ji}))\right] * \prod_{c;t_i^c=\infty} \prod_{j:t_j^c<\infty} (1 - \underline{A_{ji}}) \tag{9}$$

By this way, we can declare $B'_{ji} = 1 - A_{ji}$ without considering its variance in different cascades, while γ_c stays as it is in Eq.6. The number of variables reduces to $N * N + |C|$ as we demand.

- LSH-NetInf
  NetInf algorithm considers only the infection time interval in transmission probability function. We incorporate LSH method into NetInf by combining diffusion speed with the original transmission probability. Basically, the redefined $P_c(u,v)$ of exponential model and power-law model is:

$$\hat{P}_c(u,v) = \alpha S_p(t_u) * e^{-\frac{t_v - t_u}{\alpha S_p(t_u)}}$$

$$\hat{P}_c(u,v) = (\alpha S_p(t_u) - 1)\frac{1}{(t_v - t_u)^{\alpha S_p(t_u)}}$$

  NetInf does not consider all possible propagation trees but only the most likely one. The modified likelihood in LSH-NetInf of all cascades with the maximum weighted propagation tree is:

$$\begin{aligned} P(C|G) &= \prod_{c\in C} \sum_{T\in\mathcal{T}_c(G)} P(c|T) \\ &\approx \prod_{c\in C} \max_{T\in\mathcal{T}_c(G)} P(c|T) \\ &\approx \beta^q \varepsilon^{q'} (1-\varepsilon)^{s+s'} \prod_{(u,v)\in E_T} \hat{P}_c(v,u) \end{aligned}$$

  As noted in NetInf paper, q is network edges number and q' is ε-edges number in T; s and s' refer to those that did not transmit respectively. Note that our method just affects the pairwise transmission probability of the original NetInf algorithm without breaking its main structure or imposing extra computation burden.

Optimization methods

Clustering cascades

Dealing with root node

Experiments

Experiments for LSH and CLSH are performed to evaluate their abilities in improving network inference algorithms' performance. We mainly focus on 3 existing algorithms: NetRate, ConNie , NetInf, on which we apply LSH and CLSH method respectively.

Experiments on synthetic dataset

Synthetic networks of 256 nodes are generated from Kronecker model with parameter matrix [0.5 0.5; 0.5 0.5]. At the beginning of each cascade, root node is selected randomly and recorded as $t = 0$. Pairwise edge weights are sampled from $U(0,1)$. The larger the value of edge weight, the higher the probability of transmitting a message between this pair of nodes. Once a node is infected, it begins to infect the adjacent nodes. Each activated node infects its neighbors with probabilities generated from transmission likelihood function(Exponential model). The diffusion proceeds iteratively until time window T, which is set at 10.

Measures in this paper are AUC(Area under Precision-Recall curve) and maximum F_1 score. F_1 score is the harmonic mean of precision and recall while we consider the maximum value in evaluation.

When generating synthetic cascade data, we use the piecewise function $S_p(t_j)$ to control life span heterogeneity. Value of $S_p(t_j)$ implies the diffusion speed of message in t_j. Large deviation in $S_p(t_j)$ causes strong heterogeneity. In this part, $S_p(t_j)$ is set as [0.03 0.9 0.07 0 0]. (time window $T = 10$ as noted) When $T_j = 3.2$, $S_p(t_j) = 0.9$; when $T_j = 5.9$, $S_p(t_j) = 0.07$, etc.

Performance vs. cascade number. We make experiments of LSH-NetRate/LSH-ConNie/LSH-NetInf. By applying LSH method, the performance of network inference improves comparing with their original algorithms, as shown in Table 2 and 3. Figure 2 plots the Precision-Recall curve of LSH-NetRate/LSH-ConNie/LSH-NetInf method. Our methods make distinct improvement.

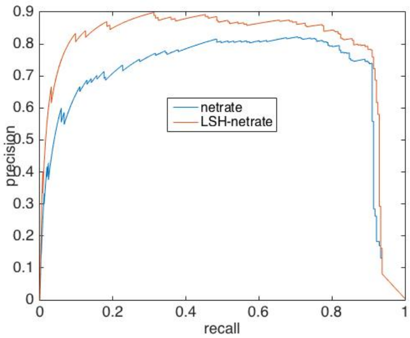
Table 2: Max F_1 score of LSH-NetRate/LSH-ConNie/LSH-NetInf vs. casNum

	casNum 300	casNum 600	casNum 900
NetRate	0.6742	0.7913	0.8161
LSH-NetRate	0.7251	0.8210	0.8425
ConNie	0.7500	0.7526	0.7905
LSH-ConNie	0.7653	0.7734	0.8138
NetInf	0.5927	0.6799	0.7360
LSH-NetInf	0.6018	0.6997	0.7446

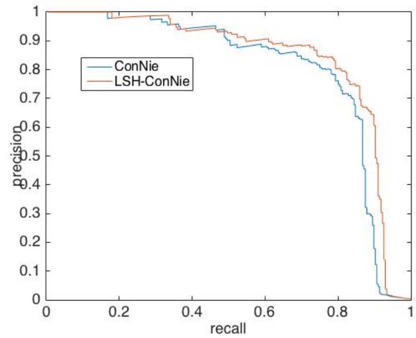
Table 3: Area under PR curve of LSH-NetRate/LSH-ConNie/LSH-NetInf vs. casNum

	casNum 300	casNum 600	casNum 900
NetRate	0.4794	0.6299	0.6858
LSH-NetRate	0.5573	0.6892	0.7805
ConNie	0.6899	0.7424	0.7985
LSH-ConNie	0.6916	0.7657	0.8334
NetInf	0.3801	0.4638	0.5754
LSH-NetInf	0.4186	0.5421	0.5764

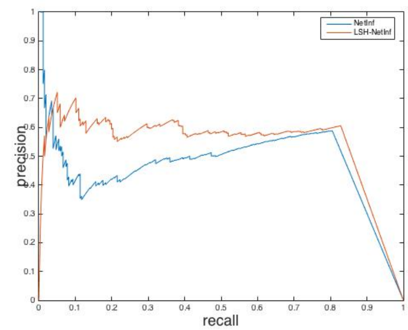
Performance vs. heterogeneity strength. We generate synthetic cascades with various heterogeneity strength, as shown in Table 4. The level of standard deviation depicts the strength of heterogeneity. Experiments in Figure 3 show



(a) LSH-NetRate



(b) LSH-ConNie



(c) LSH-NetInf

Figure 2: Precision-Recall curve of LSH-NetRate/LSH-ConNie/LSH-NetInf method

that our method gains larger advantage to original NetRate as standard deviation of S_p increases.

Table 4: 4 types of Sp function with different standard deviation.

	Sp function	standard deviation
1	[0.2 0.2 0.2 0.2 0.2]	0
2	[0.2 0.4 0.3 0.05 0.05]	0.1541
3	[0.1 0.6 0.3 0 0]	0.2549
4	[0.03 0.9 0.07 0 0]	0.3924

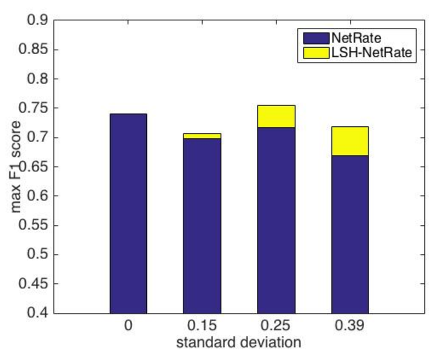


Figure 3: Experiments on different heterogeneity strength.

Setting of lag**.** lag is half the length of sliding window around target time stamp that can affect the estimation of S_p. We run our methods with 4 different lag settings separately, results are shown in Figure 4. We can see that their differences are slight as we change lag's value. But LSH method performs better with lag set around 10% of T.

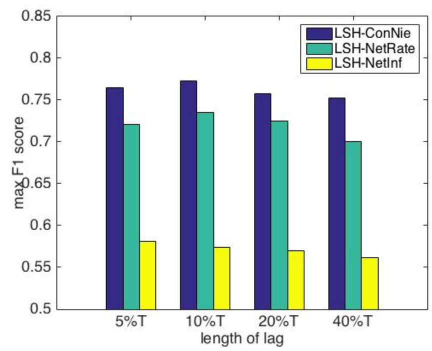


Figure 4: Experiments of different lag values.

Experiments of CLSH method. CLSH method attempts to cluster cascades so that cascades in the same group have similar $S_p(t_j)$. In this part, we use 3 types of $S_p(t_j)$ functions: [0.1 0.8 0.1 0 0] / [0.8 0.2 0 0 0] / [0 0 0 0.2 0.8]. Before simulating a cascade, we randomly choose one of this three to be the specific S_p function of this cascade.

Performance of CLSH vs. cascade number. We make experiments of CLSH-NetRate/CLSH-ConNie/CLSH-NetInf methods on synthetic cascades with multiply heterogeneity functions. As shown in Table 5 and 6, our methods outperform the original algorithms on max F_1 score and AUC.

Table 5: Max F_1 score of CLSH-NetRate/CLSH-ConNie/CLSH-NetInf vs. casNum

	casNum 300	casNum 600	casNum 900
NetRate	0.7124	0.7222	0.7254
CLSH-NetRate	0.7256	0.7425	0.7414
ConNie	0.7296	0.7531	0.7619
CLSH-ConNie	0.7474	0.7520	0.7692
NetInf	0.6727	0.7133	0.7316
CLSH-NetInf	0.6886	0.7314	0.7489

Table 6: Area under PR curve of CLSH-NetRate/CLSH-ConNie/CLSH-NetInf vs. casNum

	casNum 300	casNum 600	casNum 900
NetRate	0.5243	0.5445	0.5894
CLSH-NetRate	0.5771	0.6527	0.6582
ConNie	0.7006	0.7449	0.7612
CLSH-ConNie	0.7194	0.7600	0.7602
NetInf	0.4079	0.4248	0.5138
CLSH-NetInf	0.4319	0.4402	0.5268

Setting of k. As shown in Figure 5, clustering cascades into $3 \sim 4$ groups produces higher accuracy.

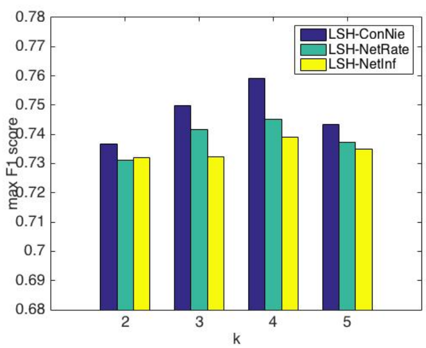


Figure 5: Experiments of different k values.

Experiments on real world dataset

We use *U.S. patent dataset* by *the National Bureau of Economic Research* to test (C)LSH method. The dataset spans 25 years from 1975 to 1999, containing 16,522,438 citations. Inventors of patents cite their sources. Therefore, we take inventors as the entities of network and follow the citations to reveal information flow. Note that we remove all self-loops in patent data, which happen when inventors cite their own previous works. By ranking inventors according to their activity, we have 147 most active inventors to form the ground truth network. We extract 150/250/350/450 transmission trees for cascade information. Experiments of (C)LSH-NetRate, (C)LSH-ConNie, (C)LSH-NetInf are made on this dataset.

For real world dataset, the lengths of cascades differ a lot. We apply Min-Max Normalization to activation times in each cascade separately. Cascades' length is 1 after transformation.

- (C)LSH-NetRate
  Performance vs. cascade number. From Table 7 we can see improvement by applying LSH method to NetRate on various data scale. Increasing k in some occasions can help improve accuracy.

Table 7: Real data: Max F_1 score of NetRate/ LSH-NetRate/ CLSH-NetRate vs. casNum

	cas 150	cas 250	cas 350	cas 450
NetRate	0.5915	0.6110	0.6418	0.6667
LSH-NetRate	0.6324	0.6364	0.6817	0.6975
CLSH-k 3	0.6328	0.6431	0.6731	0.6997
CLSH-k 5	0.6324	0.6360	0.6752	0.7121

- (C)LSH-ConNie
  Performance vs. cascade number. LSH method outperforms ConNie notably as shown in Table 8. Furthermore, applying CLSH to cluster similar cascades extends the performance advantage over original ConNie method.

Table 8: Real data: Max F_1 score of ConNie/ LSH-ConNie/ CLSH-ConNie vs. casNum

	cas 150	cas 250	cas 350	cas 450
ConNie	0.5702	0.6127	0.6412	0.6788
LSH-ConNie	0.8125	0.8222	0.8532	0.8673
CLSH-k 3	0.8249	0.8298	0.8502	0.8733
CLSH-k 5	0.8140	0.8218	0.8561	0.8581

- (C)LSH-NetInf
  Performance vs. cascade number. Table 9 and 10 present the AUC and maximum F_1 score results of our methods and NetInf algorithm. In all cases, our methods perform better than NetInf. When including more cascades, the complexity increases. Proper increment of k may lead to higher performance.

Table 9: Real data: Max F_1 score of NetInf/ LSH-NetInf/ CLSH-NetInf vs. casNum

	cas 150	cas 250	cas 350	cas 450
NetInf	0.5284	0.5382	0.5779	0.5981
LSH-NetInf	0.7604	0.7762	0.7993	0.7932
CLSH-k 3	0.7757	0.7902	0.8000	0.7938
CLSH-k 5	0.7833	0.7762	0.8129	0.8062

Table 10: Real data: AUC of NetInf/ LSH-NetInf/ CLSH-NetInf vs. casNum

	cas 150	cas 250	cas 350	cas 450
NetInf	0.7860	0.7919	0.8160	0.8292
LSH-NetInf	0.9413	0.9393	0.9410	0.9429
CLSH-k 3	0.9502	0.9473	0.9418	0.9436
CLSH-k 5	0.9547	0.9393	0.9490	0.9506

Conclusions

References

Adar, E.; Zhang, L.; Adamic, L.; and Lukose, R. 2004. Implicit structure and the dynamics of blogspace. In *Workshop on the Weblogging Ecosystem*, volume 13.

Daneshmand, H.; Gomez-Rodriguez, M.; Song, L.; and Schölkopf, B. 2014. Estimating diffusion network structures: Recovery conditions, sample complexity & soft-thresholding algorithm. In *Proc. 31st Intl. Conf. on Machine Learning*.

Domingos, P., and Richardson, M. 2001. Mining the network value of customers. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, 57–66.

Dong, X. L.; Berti-Equille, L.; and Srivastava, D. 2009. Truth discovery and copying detection in a dynamic world. *Proc. VLDB Endow.* 2(1).

Du, N.; Song, L.; Yuan, S.; and Smola, A. J. 2012. Learning networks of heterogeneous influence. In *Proc. 24th Advances in Neural Information Processing Systems*. 2780–2788.

Du, N.; Song, L.; Woo, H.; and Zha, H. 2013. Uncover topic-sensitive information diffusion networks. In *Proc. 16th Intl. Conf. on Artificial Intelligence and Statistics*.

Gomez-Rodriguez, M.; Balduzzi, D.; and Schölkopf, B. 2011. Uncovering the temporal dynamics of diffusion networks. In *Proc. 28th Intl. Conf. on Machine Learning*, 561–568.

Gomez-Rodriguez, M.; Leskovec, J.; and Krause, A. 2012. Inferring networks of diffusion and influence. *ACM Transactions on Knowledge Discovery from Data* 5(4).

Gruhl, D.; Guha, R.; Liben-Nowell, D.; and Tomkins, A. 2004. Information diffusion through blogspace. In *Proceedings of the 13th International Conference on World Wide Web*, WWW '04, 491–501.

Kempe, D.; Kleinberg, J.; and Tardos, E. 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*, KDD '03, 137–146.

Leskovec, J.; Adamic, L. A.; and Huberman, B. A. 2007. The dynamics of viral marketing. *ACM Trans. Web* 1(1).

Leskovec, J.; Kleinberg, J.; and Faloutsos, C. 2005. Graphs over time: Densification laws, shrinking diameters and possible explanations. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, KDD '05, 177–187.

Wang, S.; Hu, X.; Yu, P. S.; and Li, Z. 2014. Mmrate: Inferring multi-aspect diffusion networks with multi-pattern cascades. In *Proc. 20th Intl. Conf. on Knowledge Discovery and Data Mining*, 1246–1255.

Yang, S.-H., and Zha, H. 2013. Mixture of mutually exciting processes for viral diffusion. In *Proc. 30th Intl. Conf. on Machine Learning*.

Zhou, K.; Zha, H.; and Song, L. 2013. Learning social infectivity in sparse low-rank network using multi-dimensional hawkes processes. In *Proc. 30th Intl. Conf. on Machine Learning*.