Who Spread to Whom? Inferring Online Social Networks with User Features

Derek Wang
School of Information Technology
Deakin University
VIC, 3125, Australia
deruiw@deakin.edu.au

Sheng Wen
Centre of Cyber Security Research
Deakin University
VIC, 3125, Australia
wesheng@deakin.edu.au

Jun Zhang
Data61, CSIRO
Melbourne, VIC
Australia
zhjun@deakin.edu.au

Surya Nepal
Data61, CSIRO
Melbourne, VIC
Australia
surya.nepal@data61.csiro.au

Yang Xiang
Centre of Cyber Security Research
Deakin University
VIC, 3125, Australia
yang@deakin.edu.au

Wanlei Zhou
Centre of Cyber Security Research
Deakin University
VIC, 3125, Australia
wanlei@deakin.edu.au

ABSTRACT

Network inference has been extensively studied to better understand the information diffusion in online social networks. In this field, a widely used priori knowledge is about users' infection time stamps. Researchers also assume that the smaller the time difference between two nodes, the higher the likelihood of an edge between the pair of users. Based on the priori knowledge and assumption, current research has achieved around 60% accuracy on real media site network datasets. However, the nature of online social network is different from which of media network. In order to address the weakness of existing methods in online social network inference, we carried out a series of technical analysis and empirical studies, and demonstrated two critical problems 1) alternative spreading paths 2) users' delivery delay, which lead to the inaccuracy of previous methods. In this paper, we developed an innovative method to address the inference inaccuracy caused by the exposed two problems. This method determined the existence of an edge between a pair of users according to part of the users' features. To evaluate the performance of our proposed method, we conducted a series of experiments based on real social networks and simulations. The experiment results suggested that our method can achieve around 70% accuracy in inferring the online social network structures while existing methods fail to infer the structure.

CCS Concepts

•**Security and privacy** → **Network security;**

WOODSTOCK '97 El Paso, Texas USA

DOI: 10.475/123_4

Keywords

Network inference, machine learning, network security.

1. INTRODUCTION

1.1 Background

Networks represent a fundamental medium for spreading and diffusion of various types of information (*e.g.* news, opinions and rumours) as well as epidemics (computer virus and human being disease) [12]. For the convenience of readers, we borrow the notations from epidemiology to describe the diffusion processes: an infection (*i.e.* information or disease) appears at some node of a network and then spreads like an epidemic from neighbouring nodes to neighbouring nodes over time via the edges of the underlying network.

In the context of network diffusion, network structures are being heavily used to model and analyse the properties of network users (i.e. node) and their spreading behaviours [20, 6]. **However, there is a critical problem in the study of network diffusion.** In many cases, the underlying network of relations between nodes is unknown and observation of the network is either impossible, impractical or not cost-effective. For example, many social media users like Facebook and Twitter would not share their own lists of online contacts to the others for privacy preserving purpose[1]. It is also infeasible to build the full picture of node relationships in a network when the network dynamic change and shift over time[18, 8]. **In fact, there is very limited knowledge that we can obtain from networks about the diffusion.** We often observe the temporal traces of diffusion while the pathways over which infection spreads remains hidden. In other words. We observe the times when each node gets 'infected' by the information or disease, but the edges of the network that give rise to the diffusion remain unobservable. The real-world examples include 1) online social media platforms normally show the timestamps of posts while do not explicitely display the names of users who are in the spreading paths of these posts; 2) doctors can obtain the patients' infection time of disease according to their description but may not expose the orgin from which

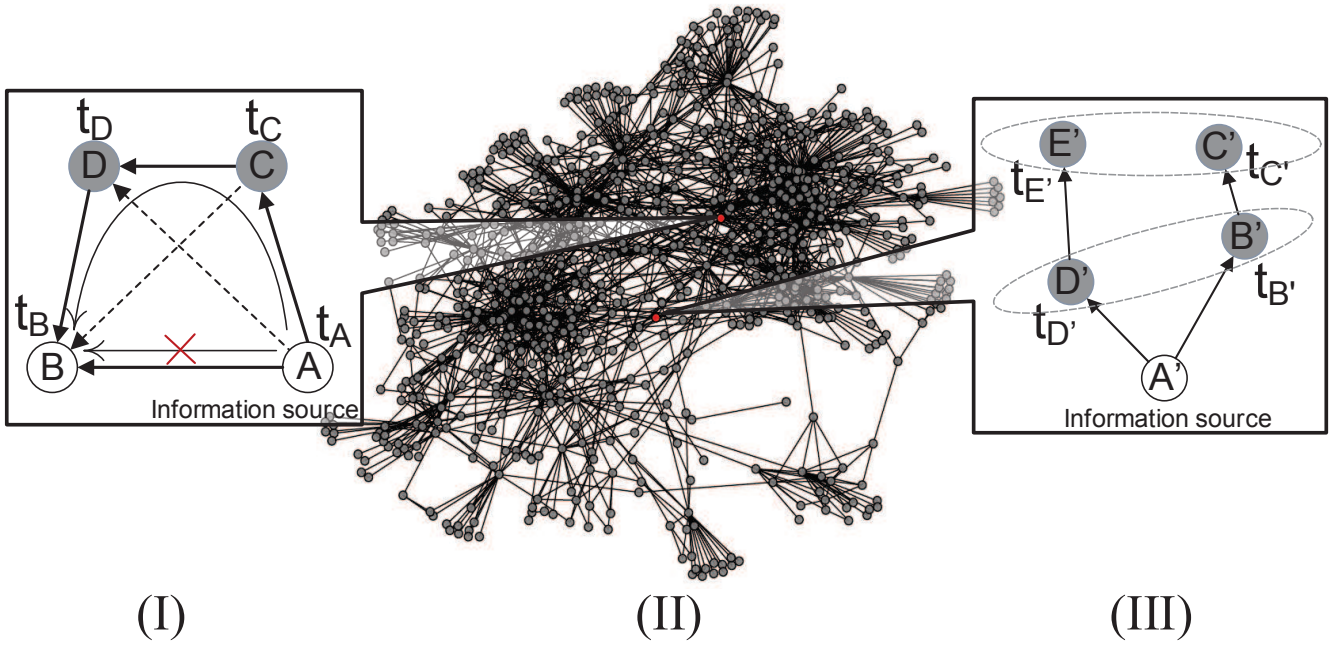

Figure 1: (II) An example of online social networks and illustrations of the problems caused by (I) alternative spreading paths and (III) users' delivery delay. The example was sketched from real Facebook topology [18].

the disease starts to spread (e.g. SARS, 2003, China [6]). In order to better understand network diffusion, it has become a fundamental issue in network diffusion field to use limited knowledge like infection timestamps to infer the network structure.

Many researchers have proposed a series of methods to refer the structures of diffusion networks. They can be roughly divided into two categories which are static network inference[3, 19] and dynamic network inference [4, 17]. Both of these methods employed continuous time information from the diffusion to construct information cascades. The likelihood of each edge's existence was then estimated by time difference between each pair of nodes. Based on estimated likelihoods, the network inference problem became an optimisation problem, which was to search a set of edges that maximise the likelihood against the sampled cascades. Besides, there is also a method proposed lately adopting the changing user status to infer network [15].

1.2 Motivation and our work

First, according to the published papers of these methods, the accuracies of previous methods are around 50 percent to 60 percent on inference of online media networks. However, can they perform as good as they were in OSN inference? We have compared the performances of diffusion cascade based methods in the experiment introduced in Section 4.2. The performances of the methods are limited on inferring OSN with representative structures. Second, current network data that can be obtained are diverse in types and are tremendous in terms of volume. Hence, it is a rational consideration to make use of complementary data in network inference aiming to enhance the performance. In order to improve the inference accuracy, we first need to investigate the reasons behind the unsatisfactory inference results. We identified two possible reasons based on technical analysis and conducted preliminary experiments to further reveal that the two problems are ubiquitous in online social network (introduced in problem statement section).

We then prosed a deep neural network based method incorporating incomplete features of OSN users in network inference. Our method is feasible to infer structure of OSN. Moreover, the method achieves better performance than almost all previous method on OSN.

The paper are organised as follows: Section 2 discusses the critical problems in previous network inference methods when put them into OSN inference. Section 3 presents our model in detail. Section 4 evaluate the performance of the proposed model. Related works are introduced in Section 5. Finally, conclusion and discussion are included in Section 6.

2. PROBLEM STATEMENT

The time stamp differences among nodes have been widely used to determine the likelihood of edge existence. For example, in the work [3, 4, 19, 17], researchers assume the distribution of time stamp differences complies with typical probabilistic distributions like power-law, exponential or Rayleigh distributions. The edges among the nodes will be recognised by a simple and 'intuitively correct' law: the smaller the time difference between two nodes, the higher the likelihood of an edge between the pair of nodes. However, are the assumption and the 'intuitively correct' law always true in the real world? If not, will the errors largely affect the network inference performance? In this section, we organised a series of empirical studies based on real dataset in order to answer these two questions.

2.1 Problem from technical perspective

There are two scenarios that will lead to the problems in network inference methods that merely use time stamps. Suppose there is a social network G for inference. The topol-

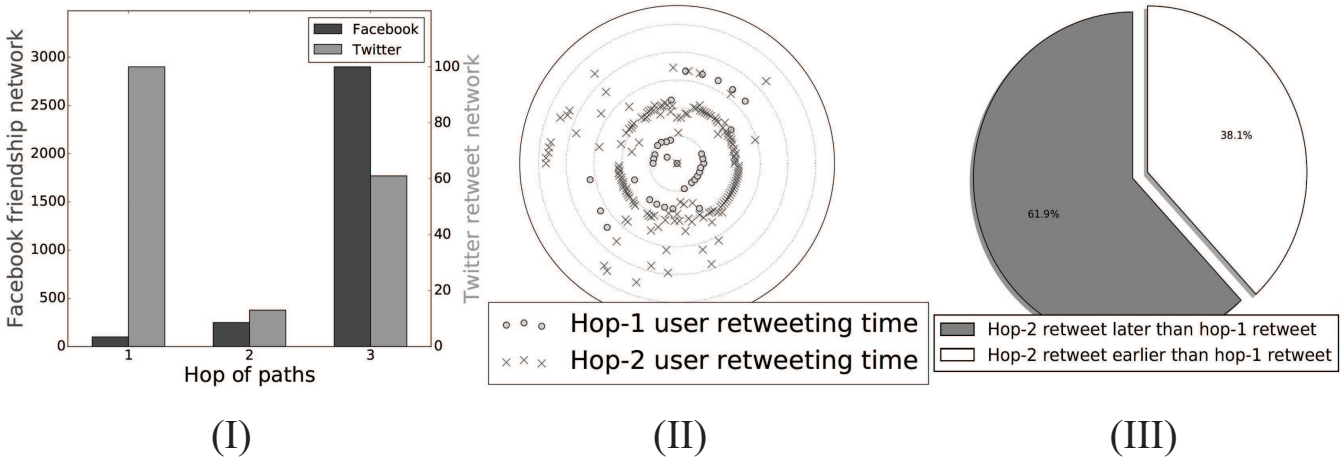

Figure 2: Empirical studies of the proposed problems in previous works (real data collected from Facebook [18] and Twitter [8]); (I) to show the number of alternative paths for an abitrary pair of neighbouring nodes; (II) to show the time differences caused by alternative paths; (III) to show the statistics of Twitter users' posts that through alternative paths

.

ogy of network G is shown in Figure1(II). A node denotes a user and an edge indicates connection between users. Firstly, a message can travel through a long path to reach the neighbouring node. For example, message posted by node A may travel through C and D to reach B, even if B is the direct neighbour of A (refer to Fig.1(I)). Hence, the time stamp recorded from C and D will be earlier than the one from B. In return, previous methods recognise (A, C) and/or (A, D) as two edges in the network G, which does not match the real structure of G. Second, message may also be stored by users due to users' behaviour and/or environment. Therefore, the time stamp may not reflect the exact receiving and posting time of messages (refer to Fig.1(III)). For example, B' may store the message sent by A' while D' and E' instantly forward the message. Accordingly, E' will post the message before B' does. Therefore, current methods will identify edge (A', D') and/or (A', E') instead of the real edge (A', B') in network G. From technical perspective, these two critical scenarios disable the network inference with an inaccurate likelihood caused by misleading time stamps.

2.2 Problem from empirical perspective

We also carried out a series of experiments based on real datasets in order to demonstrate the existence of the above problems. First, we investigated the number of alternative paths between an arbitrary pair of nodes in networks (Fig.1c). The real datasets for this part of analysis are collected from snapshots of Facebook ([18]) and Twitter [8] topologies. Without loss of generality, we randomly selected 100 pairs of nodes which have direct connections, and then analysed the number of alternative paths between each pair of nodes. Assuming the transmission probability is p (i.e. edge), probabilities of transmission through 2-hop path (p^2) and 3-hop path (p^3) are indispensable compared with p. Thus we mainly focus on the 2-hop and 3-hop alternative paths. It can be observed from Fig.2(I) that the number of 2-hop and 3-hop paths takes an indispensable proportion in propagation paths between pairs of nodes. Based on the analysis showing OSN contains large proportion of long path between adjacent users, the potential path for spreading information could be paths other than shortest path.

Second, we analysed the time stamp of information diffusion by using the twitter user activity data during the discovery of Higgs boson downloaded from SNAP [8]. We uniformly sampled 500 users as retweet sources. For each source user, the time stamps of retweeting activities of its direct neighbours (hop-1) and indirect neighbours (hop-2) were recorded. We compared the time stamps of hop-1 time stamps and hop-2 time stamps. The comparison of an example user has been presented in Fig.2(II). The closer the time stamp to the centre, the earlier the time is. It can be observed that actually large amount of hop-2 retweet time stamps are earlier than hop-1 time stamps. In real world, this phenomenon may be seen as some of your closest friends may not always be the first ones to forward your posts. Furthermore, we show the average proportion of hop-2 time stamps that are earlier than corresponding hop-1 time stamps over 500 source users in Fig.2(III). As the figure indicates, almost 4 out of 10 hop-2 users can retweet before hop-1 users. Thus, difference between time stamps does not necessarily represent the likelihood of edges.

3. METHODS

We introduce our Feature based Network Inference model (FNI) in this section. Our objective is to infer a hidden network G based on observed sub-network of G and user features. Practically, the known network G^* used for training DNN can be obtained by monitoring users who give consent to sharing their information or surveying user circles on OSN. Given the node pairs in G^* and their partial features, we are going to infer the hidden network G. We employ multilayer perceptron deep neural network (DNN) to infer network structure and transmission rates in the network. The pipeline of our method contains feature construction, node pair classification and transmission rate estimate from the DNN output. The definitions of used symbols are provided in Table 1.

Table 1: Notation definition

Symbol	Definition
G	The hidden network to be inferred.
G^*	The observed partial network, $G^* \in G$.
a_{ji}	Transmission rate between i and j.
X	User feature vector.
E	Feature vector of node pair.
$Weight_{e_m}$	Weight of the m^{th} feature in the node pair feature vector.
w_{x_u}	Weight of the u^{th} feature in the user feature vector
$w^l_{m,n}$	The weight from neuron m in the $(l-1)^th$ layer to neuron n in the l^{th} layer.
b^l_n	the bias of neuron n in the l^{th} layer.
L^{l-1}_m	Output of neuron m in the l^{th} layer.

3.1 User feature extraction

We introduced the construction of features used for edge inference in this section. We first introduce the feature preprocessing techniques. Then the construction of node pair features is presented.

Hierarchically embedding user feature. As the first step, we need to compete the missing features for some users. In real world situation, some users may hide their information from public for privacy, Thus the feature data collected would be incomplete. There are several feature embedding methods to complete missing features [2][5][14]. In the social network scenario, collected user features are usually hierarchical. A user may hide private information once the information is detailed enough for compromising the privacy of the user. Since the features in the dataset are hierarchical, we complete the missing feature by comparing the feature tree of the user with the complete feature tree. First, we define the tree graph by including all features appearing at least once in collected dataset as the nodes in the tree. We then gradually generate missing features for a user in hierarchical manner. First, the missing features at the broken branches (i.e. the missing feature that is the nearest to the feature tree root) are random generated. We consider the case of binary feature, if the value at a broken branch equals to 1, the value of the descendent feature of the broken branch will be generated randomly from $\{0, 1\}$. Otherwise, all the descendent features under the branch will be 0 (i.e. the user does not have the feature). Iteratively, we generate all features for users in the hidden network.

Node pair feature construction. We then construct the node pair features based on a mapping function $\psi(X_i, X_j \rightarrow E_{ij})$. Given two user feature vectors X_i and X_j generated from previous step, if the pair of nodes i and j has the same feature $x_\omega \in X_i, X_j$, the feature $x'_\omega \in E_{ij}$ of node pair (i, j) will be assigned as 1. Thus the user features are mapped to node pair features. Intuitively, if two user have some same features (e.g. same education provider, home town or language etc.), it is considered that the two users have higher likelihood to connect to each other. The type of feature matters in determining whether there are connections between users in OSN. To resolve the problem of weighing features, we employed deep neural network to learn the correlation between feature and connection (i.e. edge) between people on OSN.

3.2 Network inference based on DNN

In this section, we infer edges in the hidden network G based on multilayer perceptron deep neural network (DNN). Practically, the known network G^* and user features of nodes in G^* can be obtained by monitoring users who give consents to sharing their information or surveying users in circles on OSN. Given part of the node pairs in G and their partial features, we classify the rest node pairs in a pairwise manner to determine possible edges of G.

Node pair classification based on DNN. We use a fully connected DNN to infer associations between node pair features and edge existence. As the input. Features of node pairs are put into the input layer $l = 0$ in which each input neuron corresponds to a binary feature. For each perceptron in the hidden layers, the output takes the following form:

$$L^l_n = \Theta_{Activation}(\sum_{m \in l-1} w^l_{m,n} \cdot L^{l-1}_m + b^l_n)$$

We use Rectified linear unit (ReLU) as the activation function for each perceptron. Given the output values from previous layer, the rectified function will generate output as follows:

$$L^l_n = max(0, (\sum_{m \in l-1} w^l_{m,n} \cdot L^{l-1}_m + b^l_n))$$

We use cross-entropy cost to evaluation loss in DNN training. The cross-entropy cost takes form:

$$C(W, B, S^s, E^s) = -\sum_j [E^s_j ln L^l_n + (1 - E^s_j) ln(1 - L^l_n)]$$

in which W is the weights in the DNN, B is the biases in the DNN. Initial W and B are uniformly generated. Provided some input sample s from the training samples S and the desired output for s, E^s, we want to minimize the cost function. Stochastic gradient-based optimisation (SGD) is used for parameter optimisation in the DNN. Instead of traditional SGD methods, we employ Adam Optimiser [7] in our method for the efficiency in computing. Given the gradient of previous cost function:

$$\bigtriangledown_L C(\theta) = \frac{L^l_n - E^s}{L^l_n(1 - L^l_n)}$$

where θ is the parameters, Adam optimiser iteratively updates the parameters according to the following rules:

$$\begin{cases} m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot \bigtriangledown_L C(\theta_{t-1}) \\ \upsilon_t = \beta_2 \cdot m_{t-1} + (1 - \beta_2) \cdot \bigtriangledown_L C^2(\theta_{t-1}) \\ \hat{m}_t = \frac{m_t}{(1-\beta^t_1)} \\ \hat{\upsilon_t} = \frac{\upsilon_t}{(1-\beta^t_2)} \\ \theta_t = \theta_{t-1} - \frac{\gamma \cdot \hat{m}_t}{\sqrt{\hat{\upsilon_t}} + \epsilon} \end{cases}$$

wherein initial step $t_0 = 0$. $m_0 = 0, \upsilon_0 = 0$ are two moment vectors. $\beta_1 = 0.9, \beta_2 = 0.999$ are two exponential

decay rates for moment vectors. $\gamma = 0.001$ is the step size for optimisation (i.e. learning rate). $\epsilon = 10^{-8}$ adjusts the updating values of parameters in DNN. During the DNN training session, for each batch of input sample, the weight and bias stop updating when they converge. Each batch includes 100 samples in our setting. We use the trained DNN to classify node pairs. The output layer of the DNN contains two neurons to output binary classification results of the node pairs in G.

Inferring transmission rates in the network. In this part, we estimate the information transmission rates between users in the OSN. The transmission rate depict the speed of information spreading between user nodes. As the assumption, two users with higher similarity in certain features (or explicitly, personality, social status and habits) are more likely to spread information to each others. We compute the transmission rate of edge (j, i) as follows: 1) Evaluate the weight of each feature based on the output weights of the DNN input layer; 2) Map the node pair feature weights back to user nodes as the weights of user features; 3) Estimate transmission rates between users from feature weights. Given the weights W^{l_0} in the input layer, form each weight $w^{l_0}_{m,n}, m \in l_0, n \in l_1$, we compute the node pair feature weight as follows:

$$Weight_{e_m} = \frac{\sum_{n \in l_1} w^{l_1}_{m,n}}{\sum_{m \in l_0} \sum_{n \in l_1} w^{l_1}_{m,n}}$$

The node pair features weights are then convert to user feature weights W_x based on the reverse mapping function $\psi^*(E_{ij} \rightarrow X_i, X_j)$. It should be noticed that the difference in features between users is as informative as the coexistence of features between users. The weight $w_{x_u} \in W_X$ of a feature x_u in the user feature vector X thus becomes $w_{x_u} = Weight_{e_u}$. Then the transmission rate between user j and user i will be estimated by the following formula:

$$a_{ji} = \frac{\sum_{x_u : x_u \in X} w_{x_u} \sqrt{|x^j_u|^2 - |x^i_u|^2}}{\sum_{x_u : x_u \in X} w_{x_u}}$$

in which the $|*|$ means getting the value of the feature $*$ of a user. The transmission rates can be estimated by a re-training process using previously classified data.

4. EXPERIMENT

We evaluated FNI on both real world and synthetic datasets. We evaluated the inference accuracy and precision-recall of FNI on multiple datasets. We compared the accuracy of FNI with representative network inference methods based on diffusion cascade. The results are presented in the following sections.

Implementation and platform. We implemented our method using Tensorflow. The source code written in python is provided at: *URL*. The platform for running FNI is a desktop PC with Intel Core i7-3537U 2.00GHz×4, 8GB RAM and 64bit-Ubuntu 14.04 LTS system.

4.1 Real world data

We downloaded Facebook ego network dataset from Stanford University SNAP database [10]. The Facebook ego dataset was collected by a Facebook survey app which collect the social circle information of participants. The collection of the dataset implies a practical way to collect data from OSN users in the future inference tasks. The dataset contains 88,234 friendships among 4,039 users, and the personal information of users were also collected in the dataset. The personal information are parsed into features by using a tree structure[9]. There are totally 1,023 features in the feature space. The dataset has missing features which are not presented by some users. We completed the features based on method introduced in Section 3.1.

Table 2: Datasets statistics

Test dataset No.	Number of node pairs
1	50,000
2	30,000
3	10,000
4	5,000
5	2,500

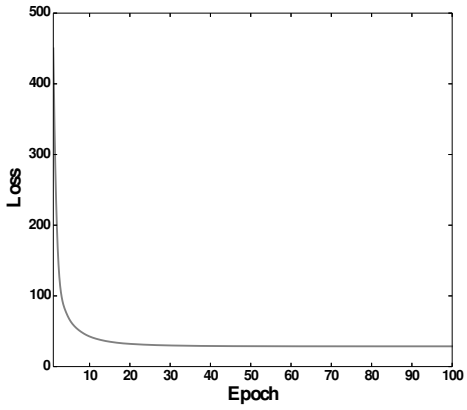

Figure 3: The relationship between loss and epoch number, It can be observed that FNI normally reaches optimal setting within 10 epochs.

Training and testing dataset. We first extracted a training dataset that approximately contains 5,000 node pairs and corresponding user features from Facebook circle network. The proportion of positive samples (i.e. node pairs that are actually edges in Facebook circles network) in the training dataset is 50%. We then tested the trained classifier on 5 different sized datasets extracted from Facebook ego network. The statistics of datasets are listed in Table 2.

Learning result optimisation. We iteratively optimised the DNN by recording the latest epoch with the highest classification accuracy. During training session. We split 10% of samples in the training dataset as test samples to calculate the accuracy of FNI after each training epoch. If the accuracy has no improvement after certain epoch window, the latest parameters will be kept as the parameters for the DNN classifier. We chose the epoch window to be 20 since most training sessions reach optimal parameter setting within 20 epochs. The correlations between loss and epoch number is plotted in Figure3. Also, we analysed the relationship between classification accuracy and training sample

number from 10 training runs. The relationship is plotted in Figure4.

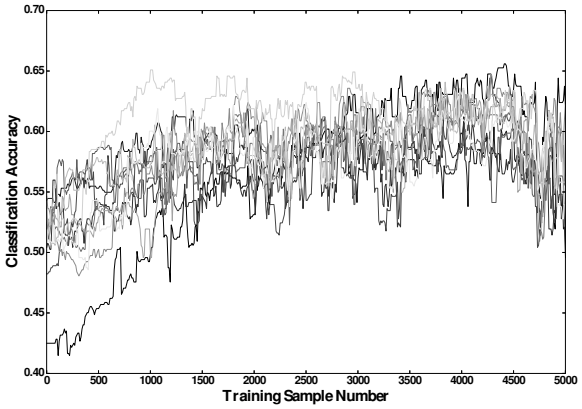

Figure 4: The relation between accuracy and number of samples used for training from 10 independent training sessions. Each curve in the figure came from a training session. In most cases, it can be observed that the accuracy can be optimised within 1,000∼2,000 samples, which only contribute a tiny proportion in the data volume used in experiments. Therefore, small volume of training samples can optimise the accuracy of the classifier.

It can be observed from Figure3 and Figure4 that FNI requires small amount of samples and epochs for completing training procedure. In the actual inference tasks, researchers can obtain small sub-network dataset and features for training and collect partial features form the hidden network for inference tasks.

Accuracy of edge inference. We evaluated the accuracy of inference by comparing the inferred network against the true Facebook circle network. The accuracy definition used in this paper is:

$$Accuracy = \frac{N_{Correct}}{N}$$

wherein $N_{Correct}$ is the number of correctly classified node pairs and N is the total number of node pairs in test sample. The results are presented in Figure5. FNI can achieve 0.6-0.7 on accuracy.

Precision and recall. We analysed the precision and recall of FNI by tuning the number of hidden layers and neurons. The precision-recall curves of FNI on the 5 datasets are presented in Figure6. The results show that FNI achieved the best performance with 1 hidden layers and 800 neurons in the layer.

The precisions and recalls of FNI with 1 hidden layer and 800 neurons are presented in Figure7. As shown in the figure, FNI achieved similar performance in the 5 test dataset. FNI has precisions around 0.6 and high recalls for edges inference task. The precision and the recall are robust against changing size of test dataset.

4.2 Synthetic data

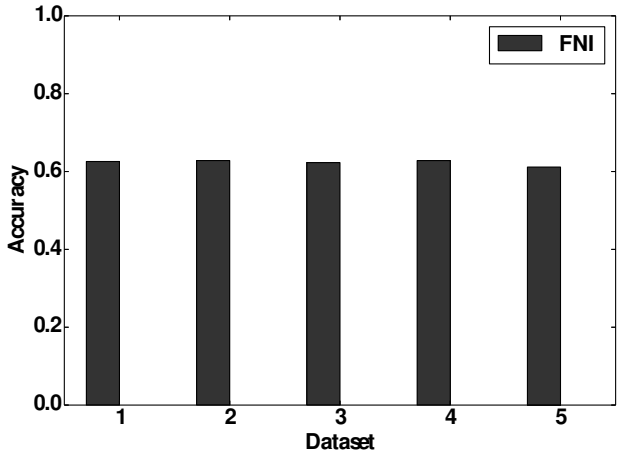

Figure 5: The accuracy of network inference on 5 datasets with different sizes. FNI achieves accuracy of inference around $0.6 \sim 0.7$ based on small set of training samples. Furthermore, it can be observed that FNI is robust against varying testing dataset size.

We construct synthetic datasets to demonstrate the problems faced by previous network inference methods. We scythed a static network by using the structure of the SNAP Facebook ego network. The structure of the network is ubiquitous in OSNs. We then simulated 100 cascades on the network following the method introduced in [13]. The purposes of using this network structure are: 1) This structure differs from media site network structure which has been extensively researched in related works [3],[4],[13],[16]; 2) This structure is a typical OSN topology that ubiquitously exists on OSN. The structure of the network is shown in Figure8. We compared the performance of FNI with NetInf and NetRate, which are two state-of-art network inference methods. The following sections will first introduce the synthesis process and comparison rules. Second, the comparison on accuracy is presented.

Synthetic data generation and rules for comparison. We generated 2 sets of cascades by simulating the spreading of information in OSN. There are 100 cascades in each sets. Most information on OSNs dies out very soon. Only few information about phenomenal topics can spread virally in OSNs. To simulate this phenomenon, we assigned each cascade a popularity value p to mimic real information. Cascade with lower p will be forwarded to/by less neighbours. Being different to the independent cascade model (ICM) used in generating synthetic data for NeInf and NetRate, our generative model does not generate diffusions in a broadcasting manner. Through parameter p, the neighbours who will receive the diffusing information are sampled from total neighbours, and the alternative diffusion paths between users (as indicated in the first problem) can be reflected in the synthetic data. Nodes in cascades record the earliest time stamp when they receive the same information for several times. We set $p = 0.02$ in the generative model. This setting also avoids generating complex

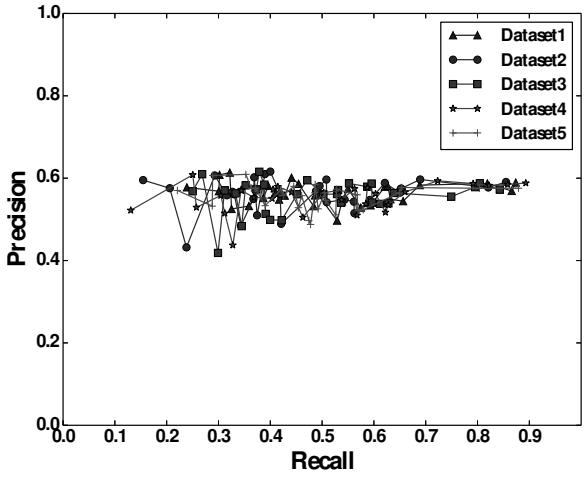

Figure 6: The P-R curves of FNI on the 5 datasets. We swept layer number from 1 to 3, and neuron number from 100 to 1,000 with 100 increase in each interval.

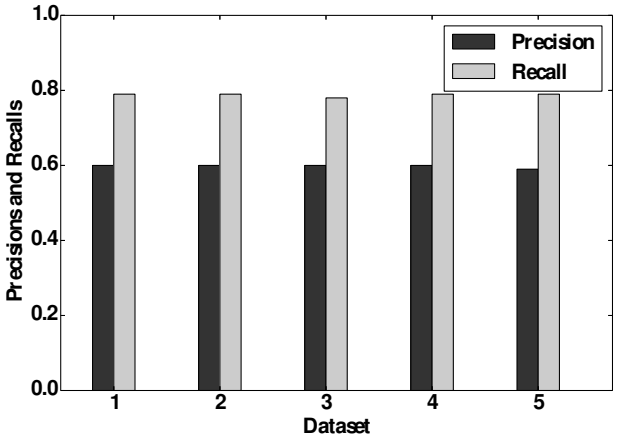

Figure 7: The precision and recall values of FNI when there is 1 hidden layer and 800 neurons in the hidden layer.

cascades that affect the computational cost of NetInf and NetRate. Once the neighbours whom the information will be forward to/by are determined by the p. The probability for forwarding will be measured by transmission likelihood. We generated the transmission likelihood between each pair of neighbours by sampling from uniform distribution $U \sim (0.01, 0.5)$. The transmission rates between nodes are drawn from $U \sim (0.01, 1)$.

We then generated the receiving time stamp once the information is forwarded out to a neighbour. For the first cascade set, the transmission model used for generating timestamps is exponential model. We expected this cascade set to reflect the effect of OSN topology structure on performance of previous methods. In the second cascade set, we added random noise time $t_\delta \sim (0.1t_\Delta, 0.9t_\Delta)$ on each forwarding action to simulate the delay mentioned in Section 2, wherein t_Δ is the time difference calculated based on exponential model. We set the cascade observation time window $T = 10$.

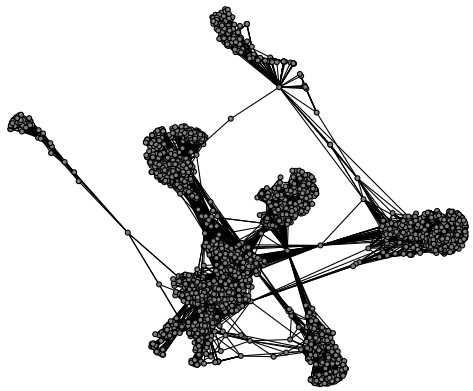

Figure 8: The network structure used to build synthetic data. The structure reflects typical OSN community topology.

We used the exact node pairs presented in the generated cascades as the test samples for the FNI. Features of these users are imported from the Facebook ego network dataset as the input for the FNI. We compared the inference accuracy of FNI against NetInf and NetRate on these same users. There are totally 240 users in Cascade set 1, 274 users in Cascade set 2.

Accuracy of edge inference. We compared the accuracy of our method with the accuracies of NetInf and NetRate on generated synthetic data as introduced. The accuracy comparison is presented in Figure9. As shown in Figure9, NetInf and NetRate had very limited accuracies. The cascades with time delay further reduced the performance of the two methods. The results support our problem statements in Section 2.

5. RELATED WORK

Network inference emerged as a research topic lately. The aim of network inference is to disclose the structure of hidden networks. The applications of inference are mainly epidemic control [12] and analysis of information difussion in online media.[19].

Existing methods can be broadly categorised into 3 classes based on observed/input data for inference. First, there are methods based on diffusion cascade[3],[4],[13],[11],[17],[19]. Diffusion cascade contains time information of the diffusion. Based on the observed time difference between node pairs, the transmission likelihoods between nodes are estimated. From then, the inference task is transformed to finding a network structure that maximises the likelihood of generating cascades as the observed data. The network to be inferred can be static[11],[3] or dynamic. Recently, method using diffusion trace emerges[15]. Diffusion trace differs from cascade on that exact timestamps are unnecessary. However,

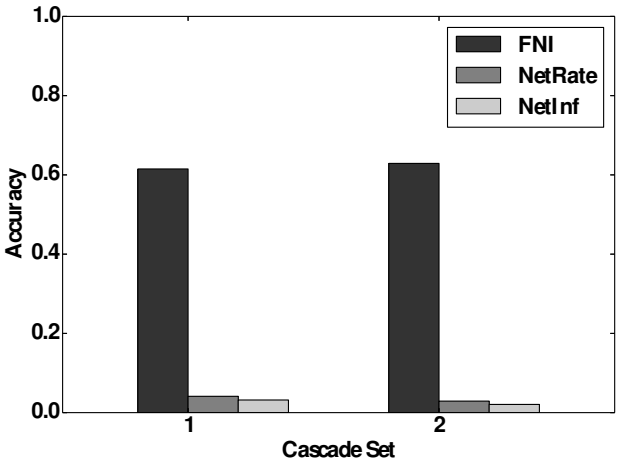

Figure 9: The accuracy comparison of NetInf, NetRate and FNI. Both of NetInf and NetRate give very low accuracy.

the sequence of nodes on the diffusion path is required. Similarly, diffusion trace based method infer hidden network by estimating a network that maximise the likelihood of generating observed traces. Additionally, a method relying on text reuse was proposed[16]. The causality of text reuse is used to infer latent network structure.

6. CONCLUSION

We propose a method which uses limited knowledge of sub-network topology and small amount of user features to infer the large hidden OSN network. The proposed method can achieve viable accuracy, precision and recall on real and synthetic social network dataset. The method outperforms diffusion cascade based methods on OSN network with typical topology. Furthermore, since our method only requires small amount of training samples, it is feasible to collect data by surveying small group of OSN users to acquired data. Incomplete user features can also be used in our model. Moreover, The method exhibits robustness, satisfactory complexity and scalability against varying size of hidden network.

As our future work, firstly, a better strategy for completing feature is going to be proposed. A main concern for user feature based network inference is that user features might be difficult to obtain. As a matter of fact, some OSN users (active users or celebrities) prefer to display more information about themselves to public while others are more willing to preserve their information as privacy. Based on observing the dissemination of posts or other content sourced from active users/celebrities in OSN, the hidden features of other users can be inferred. In return, the inferred features can strengthen the evidences for network inference using classifiers. Next, other data source may be applied on the proposed model in order to make it more suitable for real world tasks. Information within text, image and sound can both be valuable for inferring topology of latent network.

7. REFERENCES

[1] A. D. Avgerou and Y. C. Stamatiou. Privacy awareness diffusion in social networks, Nov 2015.

[2] U. Dick, P. Haider, and T. Scheffer. Learning from incomplete data with infinite imputations. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 232–239, New York, NY, USA, 2008. ACM.

[3] M. Gomez Rodriguez, J. Leskovec, and A. Krause. Inferring networks of diffusion and influence. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, pages 1019–1028, New York, NY, USA, 2010. ACM.

[4] M. Gomez Rodriguez, J. Leskovec, and B. Schölkopf. Structure and dynamics of information pathways in online media. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, WSDM '13, pages 23–32, New York, NY, USA, 2013. ACM.

[5] D. Grangier and I. Melvin. Feature set embedding for incomplete data. In *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada.*, pages 793–801, 2010.

[6] J. Jiang, S. Wen, S. Yu, Y. Xiang, W. Zhou, and E. Hossain. Identifying propagation sources in networks: State-of-the-art and comparative studies. *IEEE Communications Surveys and Tutorials, accepted*, 17(9), 2014.

[7] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[8] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

[9] J. Leskovec and J. J. Mcauley. Learning to discover social circles in ego networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 539–547. Curran Associates, Inc., 2012.

[10] J. Mcauley and J. Leskovec. Discovering social circles in ego networks. *ACM Trans. Knowl. Discov. Data*, 8(1):4:1–4:28, Feb. 2014.

[11] S. A. Myers and J. Leskovec. On the convexity of latent social network inference. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems*, NIPS'10, pages 1741–1749, USA, 2010. Curran Associates Inc.

[12] M. E. J. Newman. *Networks: An Introduction*, chapter 17 Epidemics on networks, pages 700–750. Oxford University Press, 2010.

[13] M. G. RODRIGUEZ, J. LESKOVEC, D. BALDUZZI, and B. SCHÖLKOPF. Uncovering the structure and temporal dynamics of information propagation. *Network Science*, 2(01):26–65, apr 2014.

[14] J. Schafer. *Analysis of Incomplete Multivariate Data*. Chapman and Hall, London, 1997.

[15] E. Sefer and C. Kingsford. Convex Risk Minimization To Infer Networks From Probabilistic Diffusion Data At Multiple Scales. In *Proceedings of IEEE International Conference on Data Engineering*, 2015.

[16] T. M. Snowsill, N. Fyson, T. De Bie, and N. Cristianini. Refining causality: Who copied from

whom? In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 466–474, New York, NY, USA, 2011. ACM.

[17] M. Tahani, A. M. A. Hemmatyar, H. R. Rabiee, and M. Ramezani. Inferring dynamic diffusion networks in online media. *ACM Trans. Knowl. Discov. Data*, 10(4):44:1–44:22, June 2016.

[18] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi. On the evolution of user interaction in facebook. In *Proceedings of the 2Nd ACM Workshop on Online Social Networks*, WOSN '09, pages 37–42, New York, NY, USA, 2009. ACM.

[19] S. Wang, X. Hu, P. S. Yu, and Z. Li. Mmrate: Inferring multi-aspect diffusion networks with multi-pattern cascades. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 1246–1255, New York, NY, USA, 2014. ACM.

[20] Y. Wang, S. Wen, Y. Xiang, and W. Zhou. Modeling the propagation of worms in networks: A survey. *Communications Surveys Tutorials, IEEE*, 16(2):942–960, Second 2014.