2018 IEEE 34th International Conference on Data Engineering

Inf2vec: Latent Representation Model for Social Influence Embedding

Shanshan Feng[1], Gao Cong[2], Arijit Khan[2], Xiucheng Li[2], Yong Liu[1], Yeow Meng Chee[3]
[1] *Institute of High Performance Computing*
*A*STAR, Singapore*
{feng_shan_shan,liuyong}@ihpc.a-star.edu.sg

[2] *School of Computer Science and Engineering,* [3] *School of Physical and Mathematical Sciences*
Nanyang Technological University, Singapore
{gaocong@,arijit.khan@,xli055@e.,ymchee@}ntu.edu.sg

***Abstract*—As a fundamental problem in social influence propagation analysis, learning influence parameters has been extensively investigated. Most of the existing methods are proposed to estimate the propagation probability for each edge in social networks. However, they cannot effectively learn propagation parameters of all edges due to data sparsity, especially for the edges without sufficient observed propagation. Different from the conventional methods, we introduce a novel social influence embedding problem, which is to learn parameters for nodes rather than edges. Nodes are represented as vectors in a low-dimensional space, and thus social influence information can be reflected by these vectors. We develop a new model *Inf2vec*, which combines both the local influence neighborhood and global user similarity to learn the representations. We conduct extensive experiments on two real-world datasets, and the results indicate that Inf2vec significantly outperforms state-of-the-art baseline algorithms.**

I. Introduction

Online social networks, such as Facebook, Twitter, LinkedIn, Flickr, and Digg are platforms that are used for spreading ideas and messages. Users' behaviors and opinions are highly affected by their friends on social networks, which is defined as social influence. Motivated by various applications, e.g., viral marketing [1], social influence studies have attracted extensive research attention. One fundamental problem for social influence study is to learn influence parameters from observations [2], [3], [4], [5], [6], [7], [8]. We can observe a sequence of actions of users on social networks. For example, users like a story on Digg — which is a news sharing website, and then their friends may like the story as well. Based on users' online behaviors, we aim at learning parameters to reflect the social influence. The process of modeling social influence can benefit many tasks, such as predicting who will be influenced over the social networks. Various methods [2], [3], [4], [9], [10] have been proposed to learn the influence parameters, and most of them learn diffusion probability for each edge. However, due to the sparsity of propagation observations, these methods cannot effectively estimate the influence parameters for all the edges, especially for the edges without sufficient observed propagations. Moreover, all these methods only consider the social influence in estimating influence parameters, but do not consider other factors, such as similarity of user interest.

Network embedding [11], [12], [13], [14], [15] has been recently proposed to represent each user in a latent low-dimensional space. The structure of a network is captured by the learned representations of users.

Inspired by the network embedding approaches, we investigate a new approach for modeling social influence. Instead of directly estimating propagation probability of each edge, we attempt to learn representation of each node, such that the social influence is reflected by the representations of nodes in a latent low-dimensional space. This approach has two advantages. First, it can help to effectively identify the hidden influence relationships among users. For instance, given that user u_1 can influence user u_3, and user u_2 can affect both user u_3 and user u_4, then user u_1 probably is also able to influence u_4. However, such relationships cannot be explicitly captured by previous models[2], [3]. Second, it can alleviate the challenge caused by sparse observation data. In particular, existing models cannot effectively learn probabilities for the edges without observed influence propagation. For instance, if no social influence has been observed on a link (u, v), it is hard to estimate the influence probability P_{uv}. In contrast, embedding model can learn the representation of node u and node v respectively, and then estimate the diffusion relationship between u and v.

To the best of our knowledge, none of the existing work on learning influence models jointly captures the influence propagation and network embedding, and none of previous work considers user interest similarity. To fill this gap, we propose a novel research problem: **social influence embedding**. This problem aims to effectively embed the social influence propagation in a low-dimensional latent space. The challenges of this problem are threefold. First, we need to model multiple factors that would influence users' online actions, including social network structure, past influence propagations, and similarity of user interests. Second, how to effectively learn representations of nodes based on the sparse observed propagation data? Third, the learning process should be efficient such that we can handle large-scale social

2375-026X/18/$31.00 ©2018 IEEE
DOI 10.1109/ICDE.2018.00089

networks. To address these challenges, we develop a new representation model called **Inf2vec** to learn social influence embedding.

The key of Inf2vec model is how to generate influence context, which is a set of users that would be influenced by a given user, from the observed propagations. However, it is nontrivial to generate influence context since we can merely observe users' actions, but do not know who are indeed affected by a given user. In this work, we consider two constituents for generating influence context. First, we employ local influence propagation neighborhood. Given a social network and a set of action observations, we extract a propagation network, and utilize a random-walk strategy on the propagation network to produce a set of users. These users act as local influence context for the given user. Second, we consider the similarity of user interest. A user's behavior can be influenced not only by his friends, but also by the user's individual preferences (as to be analyzed in Section III). Intuitively, users with similar interests are more likely to perform the same action. To incorporate the effect of user interest similarity, for a given user, we randomly sample a set of users who perform the same action as the global user similarity context. With the two constituents, we are able to incorporate three factors in learning node embedding: local influence context reflects network structure and influence propagation, while global user similarity context represents the factor of user interests. Based on the generated influence context, Inf2vec leverages the word2vec technique [16], [17] to learn the representations for social influence embedding.

The main contributions are summarized as follows.

- We propose a new framework to learn influence parameters from observations, which is a fundamental problem for social influence analysis. Specifically, we investigate a novel social influence embedding problem, which is to represent the influence propagation information in a low dimensional latent space. Different from most previous studies on learning influence parameters, we learn the latent representation of each node, instead of learning the propagation probability of every edge. To the best of our knowledge, this is the first work that directly utilizes representation of nodes to capture social influence.
- We propose a new algorithm Inf2vec to learn nodes' representations. The novelty of this algorithm is generating influence context, which combines both the local influence context and the global user similarity context. Consequently, our approach is able to incorporate three factors: network structure, influence diffusion, and similarity of user interests. However, none of previous work on learning influence parameters considers user interest.
- We conduct extensive experiments on two real-life datasets. The empirical results demonstrate that Inf2vec significantly outperforms the state-of-the-art baselines.

II. Related Work

With the proliferation of online social networks, a great deal of social influence data has been generated. Such data enable the influence propagation analysis in online social networks. Many research problems are proposed for influence propagation analysis, such as inferring the underlying diffusion network [18], [6], [7], topic-aware influence analysis [5], [9], and temporal dynamics of influence [4], [8]. One of the fundamental research problems in influence propagation analysis is to learn influence parameters [2], [3], [19], [20], [21] from a given social network and action log, which is the focus of our work. Therefore, in this section we review related work on learning influence parameters.

We first introduce two prevalent influence diffusion models, which are widely used in previous work to capture social influence. One is Independent Cascade (IC) Model, in which each newly activated node u affects its neighbors independently. Consider a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where each edge $(u, v) \in \mathcal{E}$ is associated with influence probability P_{uv}. Let A_t be the users that are affected at step t. Starting with an initial set of seeds A_0, other users in that graph would be activated by these seeds. Each node $v \in A_t$ tries to activate its inactive friends at time $t+1$. If one neighbor w is activated, w will switch from $inactive$ to $active$. Irrespective of whether v succeeds, v loses its ability to affect its inactive friends in subsequent rounds. If no more newly activated node exists, the whole process ends. The other is Liner Threshold Model (LT), in which an inactive node becomes active if the sum of the weights of the edges with active neighbors exceeds the threshold. Majority of the previous influence parameter learning algorithms [2], [3], [10] are based on these two prevalent models. In contrast, we propose a new data-driven algorithm to directly capture diffusion information from real-life dataset, without any prior assumption of spread models.

We proceed to review the related work on learning influence parameters based on the aforementioned spread models. Most of existing methods directly estimate the diffusion probability for each edge [2], [3], [10]. Goyal et al. [3] propose to estimate the propagation probabilities using the co-occurrence counting. Specifically, the Maximum Likelihood Estimator is employed to infer the probability: $P_{uv} = \frac{A_{u2v}}{A_u}$, where A_{u2v} is the number of times that u successfully influences v and A_u is the total number of trials. This approach is simple and efficient. Another type of methods utilize Expectation Maximization (EM) technique to learn the diffusion probabilities [2], [9], [10]. Saito et al. [2] propose an EM method to infer the probability P_{uv} for the IC model. Barbieri et al. [9] extend the EM framework to learn the topic-aware propagation probability P^z_{uv}, where z indicates a topic. Goyal et al. [21] further propose a credit distribution model that directly learns the top-k influencers from past propagation traces. All the previous studies do not consider the user interest similarity. More importantly, due to the sparsity of available propagation data, these existing approaches are not able to effectively learn the influence parameters.

Recently, Bourigault et al. [10] aims at learning the probabilities of edges based on an embedded cascade model. The social influence information of two users is captured by the Euclidean distance between their representations. Compared

with our method, this approach has several limitations. First, it does not explicitly utilize the network structure, which is important for social influence analysis. In that work, social links are created. It creates a link (u_1, u_2) if and only if u_1 performs an action before user u_2. This may not be true, since a user can be influenced by other users only if real social relationships exist such that this user can watch the activity of others. Second, the proposed approach is specifically designed for the IC model and fails to incorporate user interest factor to model user's online behaviors. Last but not least, the proposed algorithm is very slow for large-scale networks. It employs the EM technique [2], which is time consuming.

Our work is also related to the work on network embedding [11]. Generally, network embedding problem is to learn the representation of each node in a latent low-dimensional space such that the network structure can be preserved. The recent network embedding methods [11], [12], [13], [22], [15] utilize the word2vec technique [17], [16], which is developed for learning representations of words. Perozzi et al. [11] propose the Deepwalk model, which first generates context with random walks and then update the representations with skip-gram [17], and utilize a hierarchical softmax method [23] to solve it. Tang et al. [12] design the LINE algorithm, which is able to preserve both the local and global network structure by using the first-order and second-order proximity. LINE utilizes negative sampling technique to approximate the softmax. Grover et al. [13] develop the node2vec model for network embedding. To incorporate homophily and structural equivalence, node2vec defines a diverse notion of a node's network neighborhood. A biased random walk procedure is designed to explore flexible neighborhood. Yang et al. [22] develop a semi-supervised learning method for network embedding. They jointly exploit the class label and network structure to learn the embedding. Ribeiro et al. [15] exploit the word2vec technique to preserve structural identity for node representation. Wang et al. [14] develop a deep learning architecture to address network embedding problem. Different from existing network embedding methods, we additionally consider influence propagation and similarity of user interest. Therefore, our task is more challenging.

III. Social Influence Embedding

In this section, we first present the observation and analysis on real-life datasets. Based on the data analysis, we give the definition of social influence propagation.

A social network can be modeled as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of users and $\mathcal{E}$ is the set of edges. An edge (u, v) indicates that user u is a friend of user v. We are also given an action log $\mathcal{A}$, which records users' online behaviors. The action log $\mathcal{A}$ contains a set of tuples in the form of (u, i, t_u^i), which denotes that user u performs action i, e.g., like a story or a photo, at time t_u^i. Each item i corresponds to one diffusion episode $\mathcal{D}_i = \{(u, t_u^i)\}$, which is a set of users who adopt action i in chronological order.

Before introducing the observation and problem statement, we discuss two assumptions.

- If we observe that user u performs an action before user v, and if there is also a directed link (u, v) in social network, then we assume that user u influences user v. This assumption is widely made in previous studies on learning influence probability, such as [2], [3], [9]. The underlying intuition of the assumption is that if user v lists user u as a friend, user v may watch the activity of user u and be affected by user u.
- For the users that perform identical actions, we assume that they share similar user interest. This assumption is widely adopted in user behaviour analysis and recommendation systems [24]. Individual interest plays an important role in users' behaviors, and users have different interests [9]. By exploiting this assumption, we consider the user interest for modeling users' online behaviors.

A. Data Observations

1) Datasets: To study the social influence on social networks, we use two publicly available datasets, which are also used in previous work on learning social influence propagation parameters. One is Digg, which contains information about stories displayed on the front page of Digg (digg.com) in June 2009 [25]. The Digg dataset comprises 68K users connected by 823K edges. The dataset also contains Digg votes, each of which records users' voting on a particular story and the voting time. The other dataset is Flickr, which contains a friendship graph and a list of favorite marking records of the photo sharing social network (www.flickr.com) [26]. There are 162K users connected by 10M edges. The statistics of two datasets are stated in Table I. Each action contains the information of $(user, item, time)$. We observe that the action data is very sparse. It is challenging to effectively learn social influence propagation parameters based on such sparse data.

Dataset	#User	#Edge	#Item	#Action
Digg	68,634	823,656	3553	2,485,976
Flickr	162,663	10,226,532	14,002	2,376,230

TABLE I
STATISTICS OF DIGG AND FLICKR DATASET

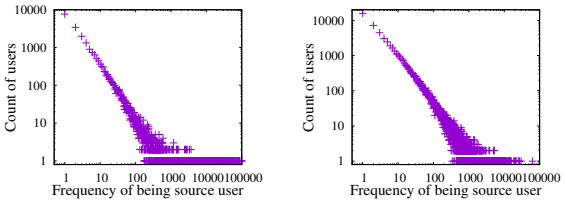
2) Observations: Given a social graph and its action log, we extract the social influence pairs based on the first assumption. We define the social influence pairs as follows.

Definition 1: **(Social Influence Pair)** Given a social network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a diffusion episode $\mathcal{D}_i$, social influence pair $(u_i \rightarrow u_j)$ exists if it satisfies: (1) $u_i \in \mathcal{V}$ and $u_j \in \mathcal{V}$; (2) $(u_i, u_j) \in \mathcal{E}$; (3) $t_{u_i}^i < t_{u_j}^i$.

For a user u_i, if his/her friend u_j performs the same action after u_i, then there exists a social influence pair $(u_i \rightarrow u_j)$ between them. In this way, we get 7.9M social influence pairs for Digg and 5.3M pairs for Flickr. Each social influence pair $(u_i \rightarrow u_j)$ contains a source user u_i and a target user u_j. To examine the characteristics of social influence pairs, we plot distributions of the source user frequency and target user frequency on Digg and Flickr dataset.

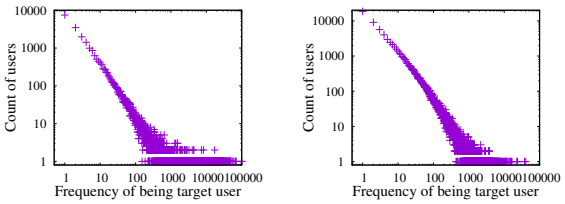
Figure 1 illustrates the distribution of source users on Digg and Flickr. We observe that the source user frequency follows

a power-law distribution. The high frequency of a user being source user indicates that this user can influence many users and thus is influential. Most of the users are not influential, while some users are extremely influential on both social networks. Similarly, as shown in Figure 2, the distribution of target users also follows the power-law distribution. It demonstrates that some users are more likely to be influenced by their friends.



(a) Digg (b) Flickr

Fig. 1. Distributions of users being source users on Digg and Flickr. The X-axis presents the number of times an user acts as a source user and the Y-axis shows the count of such users.



(a) Digg (b) Flickr

Fig. 2. Distributions of users being target users on Digg and Flickr. The X-axis presents the number of times an user acts as a target user and the Y-axis shows the count of such users.

To investigate the effect of social influence on users' online behaviors, we compute the cumulative distribution function (CDF) of the count of friends that have performed the same action before a user. Figure 3 shows the CDF on Digg and Flickr. In Digg (resp. Flickr) dataset, the CDF of $x = 0$ is 0.7 (resp. 0.5), which indicates that 70% (resp.50%) users conduct an activity without any influence from their friends. Meanwhile, 30% (resp. 50%) users perform an action after at least one of his/her friends does that. Since a user may see his/her friends' online activity, we assume that this user would be influenced by his friends. This observation demonstrates that although social influence plays a significant role in the decision of online behaviors for users, but the users' behaviors are also affected by other factors.

B. Problem Statement

Given a social network and its action log, modeling influence propagation aims to infer the influence probabilities between users. As a fundamental problem of social influence analysis in social networks, learning influence parameters has been investigated in several proposals [3], [2], [4], [10]. Figure 4(a) shows the basic idea of these existing social influence learning problems, which learn the propagation probability for each edge. Generally, these problems attempt to estimate the probabilities of $|\mathcal{E}|$ edges.

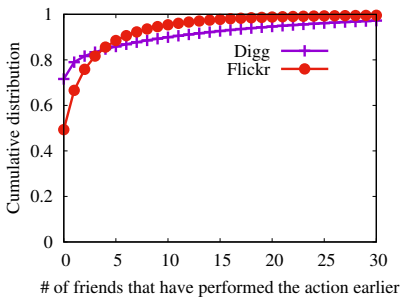
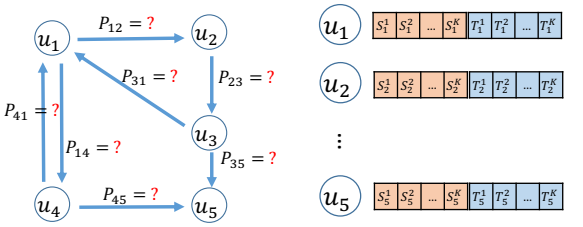


Fig. 3. The CDF of taking an action after x friends have performed the action.

Different from existing studies, we investigate a novel research problem: social influence embedding. We aim to represent the social influence propagation in a low-dimensional latent space. In this problem, we attempt to learn the representations of $|\mathcal{V}|$ nodes in the given network. The basic idea of social influence embedding is shown in Figure 4(b), where we learn the representations for nodes: $\{u_1, u_2, u_3, u_4, u_5\}$.

In social influence embedding, the propagation relationship between two users is modeled by the similarity between their vectors. Note that influence propagation is directed. To reflect the direction of social influence, user u has two vectors in K dimensional space: $S_u \in \mathcal{R}^K$ acts as the source representation, which indicates the capability to influence other users; while $T_u \in \mathcal{R}^K$ acts as the target representation, which represents the tendency of being affected by other users. Here, the number of dimension K is a tunable parameter, and its value is determined empirically.



(a) Conventional problem (b) Social influence embedding

Fig. 4. Social influence learning on social networks.

In social influence analysis, we need to consider the global property for each user. On the one hand, intuitively, some users, such as movie stars and politicians, are more influential than ordinary users in social networks. On the other hand, some users are more inclined to be affected by others. These intuitions can be explained by Figure 1 and Figure 2. However, such global property cannot be reflected by the two latent vectors that we will learn for each user. To better model the social influence, we additionally introduce two terms: influence ability bias b_u reflects the overall ability of user u to affect others, and conformity bias $\tilde{b_u}$ reflects a user's inclination to be influenced by others [27].

We are now ready to define the social influence embedding problem as follows.

Definition 2: **(Social Influence Embedding Problem)** Given a social network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, an action log $\mathcal{A} = \{D_i\}$, where D_i is a diffusion episode, and the number of dimension K, we aim to learn: (1) source embedding $S_u \in \mathcal{R}^K$ and target embedding $T_u \in \mathcal{R}^K$ in K dimensional latent space for each user u, as well as (2) influence ability bias b_u and conformity bias $\tilde{b_u}$ for each user u.

Compared with existing influence learning work that estimates probabilities for edges [2], [3], [10], our solution to the social influence embedding problem aims to better capture the social influence propagation by effectively capturing the influence relations among users and handling the data sparsity. In addition, the existing methods are designed for particular influence spread models, e.g., the IC model and the assumed influence spread models cannot take into consideration user similarity factor. In contrast, we aim to incorporate user similarity into parameter learning.

IV. Inf2vec Representation Model

We proceed to present our proposed Influence-to-vector (Inf2vec) method to address the challenges mentioned in Section I for the social influence embedding problem. We first present how to generate the social influence context. Then we state the procedure to learn representations of nodes based on the generated influence context.

A. Generating Influence Context

Given a user, we need to identify the users who are probably influenced by the user, which is called as *influence context* of the user. However, given a social network and a diffusion episode, we cannot exactly know the influence context. In addition, the social influence would spread in the social network, i.e., a user may influence other persons through the intermediate users. Furthermore, it is very important to incorporate similarity of user interest in the influence model, although it is challenging to incorporate such additional information. We next present our approach to generate the influence context, including local influence context and global similarity context.

1) Local Influence Context: Given a social network $\mathcal{G}$ and an episode $\mathcal{D}$, we can obtain corresponding social influence pairs. However, the extracted social influence pairs only reflect the first-order propagation, i.e., whether a user influences his/her friends. The social influence would spread from one user to other users, who are not confined to the first-order neighbors. For example, given two social influence pairs $(u_1 \rightarrow u_2)$ and $(u_2 \rightarrow u_3)$, we can infer that user u_1 may affect u_3 indirectly. Therefore, we need to consider such high-order influence propagation. Consequently, we can further obtain an influence propagation network by combining all the influence pairs. For each episode $\mathcal{D}_i$, we build a propagation network, which records how the information about i propagates in the social network $\mathcal{G}$.

Definition 3: **(Influence Propagation Network)** Given a social network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a diffusion episode $\mathcal{D}_i$, the propagation network is $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$, which satisfies: (1) $\mathcal{V}_i \subset \mathcal{V}$ and $\mathcal{E}_i \subset \mathcal{E}$; (2) For each $(u, v) \in \mathcal{E}_i$, there is a social influence pair $(u \rightarrow v)$ in diffusion episode $\mathcal{D}_i$.

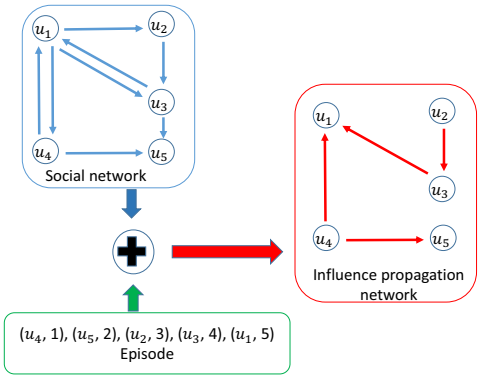


Fig. 5. An example of building influence propagation networks.

Figure 5 illustrates the idea of obtaining influence propagation networks. The example social network contains 5 users: u_1, u_2, u_3, u_4, u_5. Each episode contains the sequential action of users. Based on the social network and episode data, we extract all the social influence pairs. For example, since user u_4 performs the action before user u_5, we can obtain a social influence pair $(u_4 \rightarrow u_5)$. Similarly, we obtain other three social influence pairs: $\{(u_2 \rightarrow u_3), (u_4 \rightarrow u_1), (u_3 \rightarrow u_1)\}$. By combining these influence pairs, we get the influence propagation network.

The propagation network $\mathcal{G}_i$ is a subset of the social network $\mathcal{G}$. The influence propagation network is a directed acyclic graph due to the time constraint. Each node may have multiple children and also may have several parents.

Based on the propagation network, we consider the high-order influence in social networks. We utilize a random walk with restart process to model a user's influence spread in the influence propagation network. This approach has two benefits. First, it can simulate the influence spread sequences, and thus high-order influence can be considered. Second, it can produce more influence pairs by additionally considering high-order influence. Hence we can alleviate the challenge caused by the sparsity of diffusion data. Note that A. Goyal et al. [21] utilize a similar strategy to solve sparsity issue. They propose a credit distribution model to assign influence in propagation network. However, they only exploit first-order and second-order influence propagation. With random walk process, our method can capture higher-order propagation.

The random walk process reflects the local influence neighborhood. Given an influence propagation network $\mathcal{G}_i$ and a user u, we generate the influence context set C_u^i , which contains the users that are probably influenced by user u. We utilize a random walk with restart strategy to generate C_u^i. Starting from user u, it randomly chooses one neighbor to visit. Based on the currently visited user, it randomly samples one neighbor of this user to visit next. At each step, it has some probability to go back to user u (In our work, we set the restart ratio as 0.5 by following default setting of the work [13]). To limit the

size of C_u^i, we use a length threshold L_θ. The random walk process stops when L_θ is reached.

2) Global User Similarity Context: As shown in Figure 3, users' online behaviors are not always attributed to social influence. In this work, we further consider the user preference similarity for generating influence context. User preference has been shown very important to model users' behaviors in recommendation systems [24]. A user's online action reflects his/her personalized interest. The users with similar interest are more likely to have similar behaviors. Given a propagation network $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$, all the nodes in $\mathcal{V}_i$ have performed action i, which means that these users are interested in the same item. However, existing methods do not consider such information. Moreover, it is very hard to integrate user interest similarity in the conventional IC model, which is commonly used in existing studies on influence parameter learning [2], [10]. To address this issue, we consider the user interest similarity when generating influence context C_u^i.

To capture the user interest similarity, we additionally consider the users who perform the same action. Given a user u in propagation network $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$, we randomly sample a set of users, denoted by L'_θ, in $\mathcal{V}_i$. Note that the local influence context only reflects the local neighborhood, while the samples of similar users can reflect the global context. Hence, we consider these sampled users as the global user similarity context.

3) Algorithm for Generating Influence Context: Next, we propose an algorithm for generating the influence context pair (u, C_u^i) in the propagation network by combining the local influence context and global user similarity context. The pseudo code for generating influence neighbors is shown in Algorithm 1. Given a user $u \in \mathcal{V}_i$, we aim to generate the influence context C_u^i. To balance the contribution of these two components, we utilize a component weight α. We generate $(L_\theta = L \cdot \alpha)$ local influence neighbors by using a random walk starting at user u (line 2). Next, we randomly sample $(L'_\theta = L \cdot (1-\alpha))$ users from the nodes in $\mathcal{V}_i$ (line 3). By this way, the influence context set C_u^i consists of two types of nodes (line 4). The time complexity of Algorithm 1 is $O(L)$, since we generate L nodes as the context.

Algorithm 1: Generating Influence Context

input : Propagation network $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$, user $u \in \mathcal{V}_i$, length threshold L, component weight α
output: The user and its influence context (u, C_u^i)

1 $C_1 \leftarrow \emptyset$, $C_2 \leftarrow \emptyset$, $C_u^i \leftarrow \emptyset$;
2 $C_1 \leftarrow (L \cdot \alpha)$ nodes by random walk from $\mathcal{G}_i$ starting at u ;
3 $C_2 \leftarrow L \cdot (1-\alpha)$ nodes by uniformly sampling from $\mathcal{V}_i$;
4 $C_u^i \leftarrow C_1 + C_2$;
5 **return** (u, C_u^i);

B. Learning Representations

The next step of social influence embedding is how to effectively model the relationships between users and their influence contexts.

As shown in Section III, the social influence observations follow power law distribution, which is the same as the word frequency distribution in natural language. This finding motivates us to utilize the word2vec technique [17], [16] to learn the latent representation for each user. Word2vec has been demonstrated to capture the semantic relationships among words very well. Recently word2vec has been adopted for many other applications such as learning network embedding [11], [13] and modeling user's sequential behaviors [28].

We exploit the skip-gram architecture [17], which is to predict the context of a given user. The key of influence embedding is to estimate the probability of observing the influence context $P(C_u^i|u)$ that are generated with corresponding episode $\mathcal{D}_i$. Assuming that the users $\{v \in C_u^i\}$ are independent with each other, and then the probability $\Pr(C_u^i|u)$ is determined by each independent probability $\Pr(v|u)$.

$$\Pr\left(C_u^i|u\right) = \prod_{v \in C_u^i} \Pr(v|u) \tag{1}$$

For each episode $\mathcal{D}_i$, we can generate a list of (u, C_u^i) tuples, which is denoted by $P_{\mathcal{D}_i}$. We consider all the observed episodes $\mathcal{D}_i \in \mathcal{A}$, and attempt to maximize the log probability of them. Therefore, we define the objective function to be maximized as follows.

$$O = \sum_{\mathcal{D}_i \in \mathcal{A}} \sum_{(u, C_u^i) \in P_{\mathcal{D}_i}} \sum_{v \in C_u^i} \log \Pr(v|u) \tag{2}$$

In order to calculate this objective function, we need to compute the propagation probability from u to v. In our approach, the probability $\Pr(v|u)$ is computed by their representations: S_u is the representation of user u as source, T_v is the representation of user v as target, b_u denotes the influence ability bias of user u and $\tilde{b_v}$ indicates the conformity bias of user v. Given a user u, we predict the probability that user v being influenced by user u, which can be formulated as a softmax function. The probability $\Pr(v|u)$ is defined as:

$$\Pr(v|u) = e^{(S_u \cdot T_v + b_u + \tilde{b_v})} \Big/ Z(u) \ , \tag{3}$$

where $S_u \cdot T_v$ denotes the inner product of S_u and T_v, and $Z(u) = \sum_{w \in \mathcal{V}} e^{(S_u \cdot T_w + b_u + \tilde{b_w})}$ is the normalization term.

It is computationally expensive to directly compute Eq. (3), since calculating $Z(u)$ needs to enumerate each item $w \in \mathcal{V}$. To alleviate this issue, we adopt the *negative sampling* [16], which is popularly used to compute softmax functions. The idea of negative sampling is straightforward: instead of enumerating all the nodes, it only considers a small set of sampled nodes. We randomly generate several negative instances for each node $v \in \mathcal{V}$. Then we employ the sampled negative instances to approximate the softmax function:

$$\log \Pr(v|u) \approx \log \sigma(z_v) + \sum_{w \in N} \log \sigma(-z_w), \tag{4}$$

where $z_v = (S_u \cdot T_v + b_u + \tilde{b_v})$ and $z_w = (S_u \cdot T_w + b_u + \tilde{b_w})$, N is the set of randomly sampled negative instances and $\sigma(x) = \frac{1}{1+exp(-x)}$ is sigmoid function. In Eq. 4, the first term reflects the observed instances, and the second term models the sampled negative instances.

We exploit Stochastic Gradient Descent (SGD) method [29] to learn all the parameters. In each step, we update the parameters Θ by calculating the gradient:

$$\Theta \leftarrow \Theta + \gamma \frac{\partial}{\partial \Theta} \left(\log(\Pr(v|u)) \right), \tag{5}$$

where γ is the learning rate and $\frac{\partial}{\partial \Theta}$ indicates the gradient of parameters Θ. Based on Eq. 4, the gradient for corresponding parameters can be computed as follows.

$$\begin{aligned} \frac{\partial}{\partial S_u} &= (1 - \sigma(z_v)) \cdot T_v + \sum_{w \in \mathcal{N}} (-\sigma(z_w)) \cdot T_w \\ \frac{\partial}{\partial T_v} &= (1 - \sigma(z_v)) \cdot S_u, \ \frac{\partial}{\partial T_w} = (-\sigma(z_w)) \cdot S_u \\ \frac{\partial}{\partial b_u} &= (1 - \sigma(z_v)) + \sum_{w \in \mathcal{N}} (-\sigma(z_w)) \\ \frac{\partial}{\partial \tilde{b_v}} &= (1 - \sigma(z_v)), \ \frac{\partial}{\partial \tilde{b_w}} = (-\sigma(z_w)) \end{aligned} \tag{6}$$

The proposed Inf2vec approach is summarized in Algorithm 2. It contains two parts: the first part (lines 3–8) generates the social influence context, and the second part (lines 9–17) learns the parameters based on the generated influence context for each user. The Inf2vec algorithm starts with initializing the parameters (line 1). For each episode $\mathcal{D}_i$ in action log, we obtain the propagation network $(\mathcal{V}_i, \mathcal{E}_i)$ (line 4) and get C_u^i for each $u \in \mathcal{V}_i$ by the randomized procedure described in Algorithm 1 (line 6). We utilize a list $P_{\mathcal{D}_i}$ to store the generated (u, C_u^i) tuples for episode $\mathcal{D}_i$ (line 7). Then $P_{\mathcal{D}_i}$ is inserted to P, which denotes the tuples generated from all episodes (line 8).

After that, we learn the representations based on these generated tuples. In each tuple (u, C_u^i), we consider each node $v \in C_u^i$ and we update the parameters by negative sampling (lines 12–16). We first update the parameters for (u, v) (line 13): $S_u, T_v, b_u, \tilde{b_v}$. Then we randomly sample a set of negative instances N (line 14). Following that, we update parameters for each negative node w (lines 15–16): $S_u, T_w, b_u, \tilde{b_w}$. The parameters are updated by the SGD method in Eq. 5, and gradient of parameters can be computed by Eq. 6.

The time complexity of the learning algorithm is $O(|P| \cdot L) + O(I \cdot |P| \cdot L \cdot |N| \cdot K)$, which contains two components: generating context (lines 3-8) and learning representation (lines 9-17). $|P|$ is the size of generated (u, C_u^i) tuples, I is the number of iterations until converge, $|N|$ is the number of negative sampling (typically, 5-10). Number of dimension K indicates the length of representation vectors for each node, which means that we need to learn $O(K)$ parameters for each node. Compared with the second component, the first component can be ignored. Hence, the time complexity is $O(I \cdot |P| \cdot L \cdot |N| \cdot K)$. The space complexity of Inf2vec is $O(|\mathcal{V}| \cdot K)$, since we need to learn K-dimensional vectors for each node $v \in \mathcal{V}$.

Algorithm 2: Inf2vec

input : Social network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, action log $\mathcal{A} = \{\mathcal{D}_i\}$, learning rate γ, component weight α, number of dimension K

output: Representation for each node $u \in \mathcal{V}$: source embedding $S_u \in \mathcal{R}^K$, target embedding $T_u \in \mathcal{R}^K$, influenceability bias b_u, conformity bias $\tilde{b_u}$

1 Initialize S_u and T_u with uniform distribution $[-\frac{1}{K}, \frac{1}{K}])$, $b_u \leftarrow 0$, $\tilde{b_u} \leftarrow 0$;
2 Initialize $P \leftarrow \emptyset$;
3 **foreach** $\mathcal{D}_i \in \mathcal{A}$ **do**
4 Extract propagation network $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$;
5 **foreach** $u \in \mathcal{V}_i$ **do**
6 Generate influence context C_u^i by Algorithm 1;
7 Insert (u, C_u^i) into $P_{\mathcal{D}_i}$;
8 Insert $P_{\mathcal{D}_i}$ into P ;
9 **repeat**
10 **foreach** $P_{\mathcal{D}_i} \in P$ **do**
11 **foreach** $(u, C_u^i) \in P_{\mathcal{D}_i}$ **do**
12 **foreach** $v \in C_u^i$ **do**
13 Update $S_u, T_v, b_u, \tilde{b_v}$;
14 Sample a set of negative instances N ;
15 **foreach** $w \in N$ **do**
16 Update $S_u, T_w, b_u, \tilde{b_w}$;
17 **until** *convergence*;
18 **return** S_u, T_u, b_u, $\tilde{b_u}$ for each user u

C. Predicting Influence Propagation

Next, we introduce how to utilize learned representations of users to predict the influence propagation. A given user might be influenced by multiple users. Hence we need to take all the possible social influence into account. Given a set of users S_v that may affect user v, the likelihood that user v being influenced by S_v is defined by

$$\mathcal{F}\left(\{x(u, v), u \in S_v\}\right), \tag{7}$$

where $x(u, v) = S_u \cdot T_v + b_u + \tilde{b_v}$ reflects the likelihood that user v is affected by user u, and $\mathcal{F}()$ is an aggregation function to merge the influence from S_v. In our work, we consider four common aggregate functions as follows:

- Ave: take the average of all the elements, $\mathcal{F}(x_1, x_2, ...x_n) = \frac{\sum_{i=1}^n x_i}{n}$.
- Sum: linearly combine all the elements, $\mathcal{F}(x_1, x_2, ...x_n) = \sum_{i=1}^n x_i$.
- Max: choose the most significant factor by taking the maximum value, $\mathcal{F}(x_1, x_2, ...x_n) = Max(x_1, x_2, ...x_n)$.
- $Latest$: only consider the latest element, $\mathcal{F}(x_1, x_2, ...x_n) = x_n$.

The performance of these four aggregation functions will be evaluated in our experiments.

V. Experiments

A. Experimental Setup

1) Dataset: In the experiments, we use the two datasets, Digg and Flickr, introduced in Table I. In addition, we show a case study over a DBLP citation network dataset, which will be introduced in Section V-D.

For the action log $\mathcal{A} = \{\mathcal{D}_i\}$ of Digg and Flickr, we randomly select 80% episodes as training set, 10% as tuning set, and 10% as test set.

2) Parameter Setting: The default number of dimensions is $K = 50$. The length threshold L is set as 50 by default, i.e., the size of influence context is limited to 50. Based on the empirical study on tuning set, we set the default component weight $\alpha = 0.1$. The learning rate γ is set as $\gamma = 0.005$. To predict the influence propagation, the default aggregation function is Ave in Eq. 7.

All methods are implemented in C++ and experiments are conducted on a windows server, with a single core Intel(R) Xeon (R), 2.80 GHz CPU and 60 GB memory.

3) Evaluated Methods: We evaluate the performance of the following approaches.

- **DE**: degree-based method. The probability of each edge is set as $P_{uv} = \frac{1}{Indegree(v)}$, where $Indegree(v)$ is the number of friends of user v. This method is widely used in social influence analysis, such as influence maximization problem [1].
- **ST**: the static model with maximization likelihood estimator method [3]. The probability P_{uv} of edge (u, v) is computed by $P_{uv} = \frac{A_{uv}}{A_u}$, where A_{uv} is the number of actions that user u performs before user v and A_u is the total number of actions that u performs.
- **EM**: the Expectation-Maximization method based on the independent cascade (IC) model [2], which learns the propagation probability of each edge.
- **Emb-IC**: the embedded cascade model [10], which is the state-of-the-art approach to learn representation of influence diffusion. Emb-IC is based on the IC model, and the parameters are inferred by an EM algorithm similar to the algorithm [2].
- **MF**: the user-user matrix factorization method by Bayesian Personalized Ranking [30]. The entry of the matrix is the frequency that two users take the same action. If two users do not share any common action, the entry is set as 0. Note that this method only reflects the global user similarity.
- **Node2vec**: the node2vec model [13], which is the state-of-the-art algorithm for network embedding. Note that node2vec only considers the social network structure.
- **Inf2vec**: the proposed model as described in Algorithm 2, which considers both the local influence context and global user similarity context.

Note that the first 4 methods (DE, ST, EM, and Emb-IC) are based on the IC model. The probability of user v performing an action is calculated by the social influence from his friends:

$$\Pr(v) = 1 - \prod_{u \in S_v} (1 - P_{uv}), \tag{8}$$

where P_{uv} is the transition probability from u to v, and S_v is the set of friends who may influence user v.

The other two baselines (MF and Node2Vec), as well as our method Inf2vec are latent representation models. Given a set of users S_v that may affect user v, the likelihood score is computed by Eq. 7.

4) Evaluation Tasks: For performance evaluation, we consider three tasks as follows.

- **Activation prediction**. Following the evaluation method [3], we include this task, which is to predict whether a user will be influenced by his friends. Note that it only considers first-order influence propagation.
- **Diffusion prediction**. This is to follow the evaluation method in work [10]. Given a set of seed users, we attempt to identify users that are influenced by them. Different from activation prediction, diffusion prediction additionally considers the high-order propagation.
- **Visualization**. To provide better understanding of learned representations, we map them to a 2-dimensional space by using a dimension reduction tool [31].

Next, we will discuss the experimental results of these tasks. In addition, we also report results about sensitivity and efficiency, as well as a case study on a citation network.

B. Comparison with Baselines

1) Activation Prediction: To evaluate the quality of learned influence parameters, we utilize the evaluation strategy in the work [3]. Given a test episode $\mathcal{D}_i = \{(u, t_u^i)\}$, which contains the chronological order of users, we want to predict the users who will perform the action i in the future. We denote the users who have already performed a given action as activated users. The set of activated users is initially empty. We read records in $\mathcal{D}_i$ one by one. Once we read a record (u, t_u^i), we insert u into the set of activated users. These activated users would further affect other users. A user v is a *candidate* user if at least one of his friends has performed the action. Let S_v denotes the set of activated neighbors who may affect user v. In this way, we can obtain candidate users and their activated friends, i.e., $\{(v, S_v)\}$. We attempt to predict whether a candidate user v will be activated by S_v.

Given S_v, we calculate the likelihood score of activating candidate user v: IC-based methods utilize Eq. 8, while latent representation methods employ Eq. 7. Then we rank all the candidate users by their scores. The ground-truth users are those who have actually been activated by their neighbors, i.e, those in $\mathcal{D}_i$. Ideally, the ground-truth users can be ranked higher than the other users. Since we evaluate the performance based on the ranking, it is fair and reasonable to compare IC-based methods and representation-based methods.

We consider three evaluation metrics.

- AUC. Following the work [3], we utilize the area under curve (AUC) value of Receiver Operating Characteristic (ROC). Different from previous work [3] that sets a threshold value to make predictions, which is difficult to set, we utilize the ranking scheme [32] to calculate the AUC value.
- MAP. Since only a very small fraction of test cases are positive (be activated by its neighbors) and most of them are negative (not be activated by its neighbors), mean average precision (MAP) is informative for such imbalanced situation[33].
- P@N. We are interested in the positive cases that users are affected by their friends, and thus we investigate the

precision for top-N predictions. In our experiments, we set N as 10, 50 and 100 respectively.

Dataset	Method	AUC	MAP	$P@10$	$P@50$	$P@100$
Digg	DE	0.4144	0.0170	0.0098	0.0101	0.0099
	ST	0.8619	0.1790	0.5221	0.2917	0.2344
	EM	0.8623	0.2071	0.4994	0.3482	0.2959
	Emb-IC	0.8072	0.1503	0.4721	0.2621	0.2203
	MF	0.8568	0.1691	0.2783	0.2813	0.2614
	Node2vec	0.6437	0.0322	0.0323	0.0513	0.0434
	Inf2vec	**0.8893**	**0.2744**	**0.6401**	**0.4389**	**0.3729**
	(stdve σ)	(0.0003)	(0.0033)	(0.0128)	(0.0100)	(0.0028)
Flickr	DE	0.4782	0.0166	0.0071	0.0055	0.0050
	ST	0.7738	0.0566	0.0953	0.0710	0.0623
	EM	0.7479	0.0582	0.1056	0.0787	0.0655
	Emb-IC	0.7213	0.0242	0.0433	0.0421	0.0374
	MF	0.7785	0.0353	0.0656	0.0517	0.0451
	Node2vec	0.5693	0.0065	0.0017	0.0044	0.0043
	Inf2vec	**0.8068**	**0.0606**	**0.1564**	**0.1043**	**0.0824**
	(stdve σ)	(0.0022)	(0.0029)	(0.0035)	(0.0008)	(0.0007)

TABLE II
THE RESULTS OF ACTIVATION PREDICTION ON DIGG AND FLICKR

The experimental results of various methods on Digg and Flickr are presented in Table II. In each table, we show the results of the five evaluation metrics. We observe that the proposed method Inf2Vec consistently outperforms all the other methods. In this paper, the reported results of latent representation models are the average value of 10 runs, and we provide the standard deviation score (stdev σ) of the Inf2vec algorithm. Note that all reported improvements over baseline methods are statistically significant with p-value < 0.05.

We find that DE gets the lowest scores on both Digg and Flickr. The poor performance of DE method indicates that this naive approach is not feasible for learning social influence parameters, although it is widely used. ST estimates the diffusion probabilities between edges based on observations, and can achieve reasonable performance. Compared to ST, EM utilizes a complex Expectation-Maximization technique to learn influence parameters. Although the technique is more complex and time consuming, the improvement is not significant. The performance of Emb-IC is slightly worse than ST and EM, which indicates that the Emb-IC is not suitable for the activation prediction task. The Inf2vec model outperforms all the IC-based methods, including ST, EM and Emb-IC. The possible reasons are twofold. First, these IC based methods attempt to learn the transition probability of each edge, but it is hard for them due to the data sparsity. There is no sufficient historical online behavior data available to exactly estimate probabilities of edges. Second, these IC based methods cannot incorporate the similarity of user interest, which is an important factor in modeling people's behaviors.

For the two latent representation based baselines, MF performs reasonable well while node2vec performs worse. MF makes use of only user interest similarity, and the reasonable results achieved by MF demonstrate the usefulness of global user interest similarity in learning influence parameters. But its performance is not as good as Inf2vec, since it does not exploit the local influence context. This comparison indicates that local influence context is helpful. The reason for the poor performance of node2vec is perhaps that it considers only the social network structure, but not the other two factors. This result indicates that directly utilizing the existing algorithms of network embedding does not work for the social influence embedding problem.

To further investigate the impact of user similarity, we consider a special case of the proposed Inf2vec by setting the component weight $\alpha = 1.0$ in Algorithm 2. In this setting, Inf2vec only considers the local influence propagation context, which is denoted as Infvec-L. We report the results of Inf2vec-L in Table IV. As indicated by the results of activation prediction task, Inf2vec-L consistently performs worse than Inf2vec. Inf2vec-L only utilizes local influence context, but not global user similarity context, to learn representation of nodes. This indicates that the global user similarity context is helpful to model the social influence embedding.

2) Diffusion Prediction: Next, we investigate the performance of various methods for diffusion prediction task [10]. Given a set of activated users, we attempt to identify the set of users who will be influenced by them. For each test episode, we exploits the first 5% users as the seed users, and the rest 95% users as ground truth. Seed users are initially activated users, and they may affect the other users. In this task, we need to consider both the first-order (1-hop) and high-order (multiple-hop) influence propagation of seed users. Similar to activation prediction, we calculate the score of each node given the seed users. For representation models (FMC, MF, Node2vec, and Inf2Vec), we directly employ Eq. 7 to compute the score. For IC-based models (DT, ST, EM, and Emb-IC), we exploit the Monte-Carlo simulation [1] to estimate the probability score for each user, which is widely used for computing influence spread. Starting from a set of seed users, it simulates the diffusion process 5,000 times. Each newly activated user has a single chance to activate his neighbors independently. If no more new activated node exist, the simulation process ends. To assess the performance of diffusion prediction, we also utilize AUC, MAP, and P@N as evaluation measures.

The experimental results of various methods on Digg and Flickr are presented in Table III. We observe that the proposed Inf2vec always achieves the best performance on both datasets. The experimental results indicate that representation models is able to effectively capture the influence propagation information. We also present the results of In2vec-L for diffusion prediction task in Table IV. The performance of Inf2vec-L is worse than Infvec, which indicates the importance of similarity of user interest. It is also worth noting that Monco-Carlo simulation with IC model is extremely time consuming [1], while representation models can complete the prediction task in much shorter time. For example, Inf2vec uses 41 seconds and Emb-IC uses 9,246 seconds to finish the diffusion prediction task on Digg.

3) Visualization: One interesting application is to produce visualizations of the learned representations. Each node u has two vectors S_u and T_u, and hence we concatenate them as one representation vector for user u. Since the representations

Dataset	Method	AUC	MAP	$P@10$	$P@50$	$P@100$
Digg	DE	0.6183	0.0173	0.0121	0.0145	0.0132
	ST	0.6874	0.1064	0.6735	0.3841	0.3091
	EM	0.7095	0.1241	0.6261	0.4364	0.3572
	Emb-IC	0.6649	0.1047	0.5458	0.3912	0.3286
	MF	0.8677	0.1347	0.5087	0.4059	0.3389
	Node2vec	0.6606	0.0219	0.0810	0.0718	0.0556
	Inf2vec	**0.8904**	**0.1793**	**0.7386**	**0.4750**	**0.3932**
	(stdev σ)	(0.0002)	(0.0015)	(0.0214)	(0.0107)	(0.0049)
Flickr	DE	0.6177	0.0026	0.0025	0.0048	0.0041
	ST	0.6840	0.0242	0.1215	0.0871	0.0685
	EM	0.7479	0.0260	0.1115	0.0773	0.0636
	Emb-IC	0.7582	0.0199	0.0955	0.0754	0.0622
	MF	0.8699	0.0280	0.1044	0.0832	0.0703
	Node2vec	0.6233	0.0023	0.0010	0.0053	0.0048
	Inf2vec	**0.8778**	**0.0301**	**0.1254**	**0.0943**	**0.0759**
	(stdev σ)	(0.0011)	(0.0004)	(0.0054)	(0.0009)	(0.0007)

TABLE III
THE RESULTS OF DIFFUSION PREDICTION ON DIGG AND FLICKR

Results of Inf2vec-L for Activation Prediction Task					
Dataset	AUC	MAP	$P@10$	$P@50$	$P@100$
Digg	0.8649	0.1837	0.4880	0.2996	0.2576
Flickr	0.7974	0.0437	0.1058	0.0783	0.0665

Results of Inf2vec-L for Diffusion Prediction Task					
Dataset	AUC	MAP	$P@10$	$P@50$	$P@100$
Digg	0.7326	0.0928	0.5886	0.3165	0.2622
Flickr	0.8148	0.0219	0.0845	0.0677	0.0594

TABLE IV
THE RESULTS OF INF2VEC-L ON DIGG AND FLICKR.

are proposed to capture influence propagation, the nodes in frequent social influence pairs should be near in the latent space. For example, if pair $(u \rightarrow v)$ is frequently observed in episodes, then the representation of u should be close to representation of node v. We investigate three latent representation models (MF, Node2vec, and Inf2vec) and the Emb-IC. Although Emb-IC is an IC-based method, it also learns representations of nodes.

We show the visualization result of four methods on Digg dataset in Figure 6. Based on the extracted social influence pairs in Digg, we select the 10,000 most frequent pairs, which cover 524 nodes. We map the 524 nodes into a 2-dimensional space by the t-SNE algorithm [31]. Particularly, we highlight the top-5 most frequent social influence pairs, where the two nodes in a pair are marked with same symbol. For instance, the 2 circles represent two nodes in an influence pair. As shown in Figure 6(d), each pair of symbols are always close to each other, which indicates that their representations can effectively capture their propagation relationships. However for other models (Emb-IC, MF, and Node2vec), some pairs of nodes are far away. The visualization result shows that Inf2vec is able to learn meaningful representations for social influence.

C. Sensitivity and Efficiency

1) Effect of Parameters: To investigate the effect of aggregation function $\mathcal{F}()$ in Eq. 7, we report the experimental results for activation prediction task on Digg and Flickr in Table V. We consider 5 evaluation metrics: AUC, MAP, $P@10$, $P@50$, $P@100$. For all evaluation metrics, the larger value means better performance. For Digg dataset, the Ave aggregation achieves the best performance except $P@100$. The highest score of $P@100$ belongs to $Latest$. Overall, the Ave performs better than $Latest$. For Flickr, the Ave obtains the highest values for all evaluation metrics. Therefore, in our experiments, we use Ave as the default aggregation function.

Dataset	$\mathcal{F}()$	AUC	MAP	$P@10$	$P@50$	$P@100$
Digg	Ave	**0.8898**	**0.2732**	**0.6255**	**0.4412**	0.3687
	Sum	0.8631	0.1320	0.2002	0.1164	0.1125
	Max	0.8856	0.2429	0.5567	0.3895	0.3397
	$Latest$	0.8890	0.2669	0.6014	0.4297	**0.3705**
Flickr	Ave	**0.8127**	**0.0614**	**0.1491**	**0.1043**	**0.0837**
	Sum	0.7943	0.0276	0.0251	0.0335	0.0340
	Max	0.8136	0.0486	0.1067	0.0865	0.0727
	$Latest$	0.8080	0.0545	0.1286	0.0946	0.0766

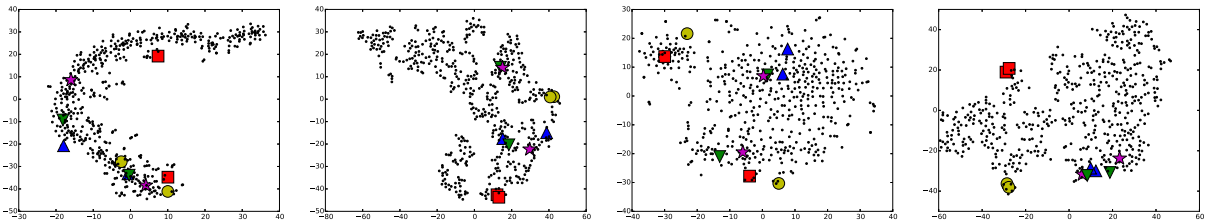
TABLE V
THE RESULTS OF AGGREGATION FUNCTIONS ON DIGG AND FLICKR

To investigate the effect of number of dimension K, we show the MAP results by varying K in Figure 7. Generally, the MAP increases with the increase of K because high dimensions can better embody the influence relationships. The performance drops when K becomes too large, which may be caused by learning too many parameters based on relatively sparse observations. The result implies that the highest MAP may be obtained between $K = 50$ and $K = 100$. In addition, as analyzed in Section IV-B, the computational cost increases linearly with the increase of K, and thus the larger value of K needs more running time. Therefore, considering the trade-off between running time and accuracy, we set $K = 50$ by default in our experiments.

To study the influence of context length threshold L, we show the MAP results with various L in Figure 8. Overall, the performance increases with the increase of L because more training instances can be exploited to learn node embedding. The MAP with $L = 100$ on Flickr dataset is slightly worse than $L = 50$, which may be caused by the over-fitting of learning process. With a larger L, more influence context nodes are generated by Algorithm 1, which leads to higher computational cost. Empirically, we set $L = 50$ for a trade-off between effectiveness and running time.

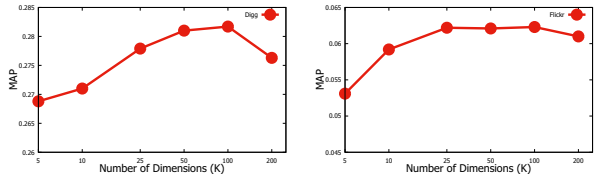
2) Efficiency: Next, we investigate the efficiency for evaluated algorithms. We compare the running time of Inf2vec model and Emb-IC, which is the state of the art algorithm [10]. Based on the IC model, Emb-IC [10] employs the EM framework [2] to learn representations. Empirically, these two methods would converge after 10-20 iterations. Therefore, we report the running time of one iteration with different K in Figure 9. For both methods, the running time increases with number of dimension K. We can find that the running time of Inf2vec is much less than Emb-IC model. For example, Inf2vec is 6 times (12 times) faster than Emb-IC method on Digg (Flickr), when K is set as 50.

Note that Inf2vec generates more nodes in the influence context by Algorithm 1. If we exploit the same setting as Emb-IC, i.e., only exploit extracted social influence pairs (without Algorithm 1), running time of one iteration of our method is reduced to 32 (120) times less than Emb-IC on Digg (Flickr).



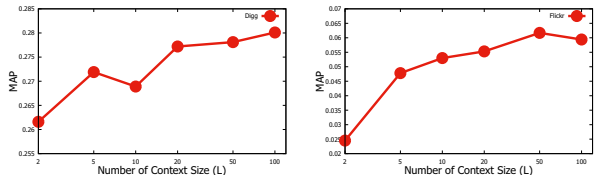
(a) Emb-IC (b) MF (c) Node2vec (d) Inf2vec

Fig. 6. The visualization of learned representations for Digg dataset.



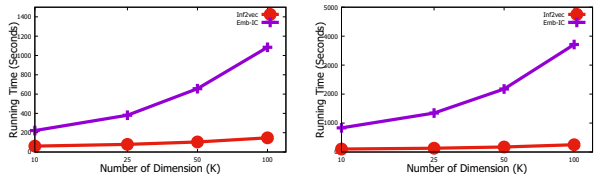
(a) MAP on Digg (b) MAP on Flickr

Fig. 7. Effect of number of dimension K on Digg and Flickr.



(a) MAP on Digg (b) MAP on Flickr

Fig. 8. Effect of context size L on Digg and Flickr.



(a) Running time on Digg (b) Running time on Flickr

Fig. 9. Running time of one iteration on Digg and Flickr.

Overall, the results suggest that Inf2vec runs much faster than Emb-IC and can be adopted for large-scale datasets.

D. Case Study

To provide intuitive understanding of our embedding model, we additionally investigate a case study on citation networks. The purpose of this case study is to compare the embedding model with conventional influence learning model. We utilize the "DBLP-Citation-network-V9" dataset (https://aminer.org/citation) [34], which collects the authors and references of 3.6M papers. We choose the papers related to data engineering including ICDE, SIGMOD, VLDB, KDD, ICDM, CIKM, TKDE, TODS, and TOIS. Finally, we get 4,345 papers with 4,259 authors. If a paper cites a reference, then the authors of the reference would influence the authors of the paper. In this way, we obtain 138,046 author influence relationships. We randomly select 80% as training set, and 20% as test set.

To capture social influence, embedding model learns parameters of nodes while conventional model learns parameters of edges. To make fair comparison, we only exploit first-order social influence pairs in embedding model. Given the influence pairs, we learn authors' representations by Eq. 4. For conventional model, we learn the probabilities by the ST model [3]. Given a test author, we attempt to predict top-10 researchers that cite the publications of test author. For embedding model, we utilize Eq. 7 to compute the likelihood score of being influenced. While for conventional influence learning model, we run 5,000 Monto-Carlo simulations to calculate the score.

Table VI shows the predicted top-10 researchers that would be influenced by 3 test authors. Here we examine three authors with most papers: Michael Stonebraker, Hector Garcia-Molina and Rakesh Agrawal. By using embedding model and conventional model respectively, we predict 10 followers that would cite their papers. Sign "+" indicates that this person indeed cites test author's papers, i.e., there is an influence relationship in test set. Sign "-" means that we do not observe such influence relationship in test set.

As summarized in the last row of Table VI, embedding model can identify more true followers of each test author. In addition, we conduct quantitative evaluation of the top-10 prediction for all the test authors in test set. The average precision of embedding model is 0.1863, which is much better than the average precision of conventional model (0.0616). This would be explained by two reasons. First, the citation relationships are very sparse. The conventional model fails to estimate accurate influence probabilities from the sparse observation data. The embedding model is able to learn representations of nodes from the limited number of citation relationships. Second, the embedding model directly predicts followers by the learned parameters without relying on any underlying diffusion model. However the conventional model relies on the IC model, which may not be accurate. Overall, experimental results demonstrate that the embedding model can effectively capture the academic citation relationships among researchers, which validates the idea of using embedding model for learning social influence.

VI. Conclusion

In this paper, we study the social influence embedding problem, which is to represent each user with latent vectors. We propose a new algorithm Inf2vec, which incorporates three factors: network structure, influence propagation, and similarity of user interest. The key technical contribution

Author	Michael Stonebraker		Hector Garcia-Molina		Rakesh Agrawal	
Method	Embedding Model	Conventional Model	Embedding Model	Conventional Model	Embedding Model	Conventional Model
Top-10 predicted followers	Hans-Jrg Schek (-)	Stephen Todd (-)	Dennis R. McCarthy (-)	Stephen Todd (-)	Raymond A. Lorie (+)	Raymond A. Lorie (+)
	W. Kevin Wilkinson (-)	Mosh M. Zloof (+)	Marek Rusinkiewicz (+)	Mosh M. Zloof (-)	Morton M. Astrahan (+)	Morton M. Astrahan (+)
	Mosh M. Zloof (+)	Gerhard Jaeschke (-)	JC Freytag (+)	Raymond F. Boyce (-)	Peter Klahold (+)	Carlo Zaniolo (-)
	Avraham Leff (+)	Jeffrey F. Naughton (-)	Jeffrey Goh (+)	Franois Bancilhon (-)	Rajeev Rastogi (-)	Patricia G. Selinger (+)
	Marie-Anne Neimat (-)	Catriel Beeri (-)	Waqar Hasan (-)	Jeffrey F. Naughton (-)	Calton Pu (-)	Raymond F. Boyce (-)
	Kyuseok Shim (-)	Hans-Jrg Schek (-)	Gabriel M. Kuper (-)	Carlo Zaniolo (-)	R. Erbe (+)	Stephen Todd (-)
	Hans-Peter Kriegel (+)	S. Bing Yao (+)	Franois Bancilhon (-)	David Maier (-)	Tobin J. Lehman (-)	Mosh M. Zloof (-)
	George Samaras (+)	Yehoshua Sagiv (-)	King-Ip Lin (-)	Lawrence A. Rowe (-)	Andreas Reuter (+)	Gerhard Jaeschke (-)
	Harry K. T. Wong (-)	Arie Shoshani (-)	Dennis Shasha (-)	Michael Hammer (-)	Raymond T. Ng (+)	C. Mohan (+)
	Roberta Cochrane (-)	Serge Abiteboul (-)	Gio Wiederhold (-)	Gerhard Jaeschke (-)	Alexander Tuzhilin (+)	Vincent Y. Lum (-)
Accuracy	4/10	2/10	3/10	0/10	7/10	4/10

TABLE VI
PREDICTION OF TOP-10 FOLLOWERS ON CITATION NETWORK

lies in the approach of generating influence context, which combines the local social influence context and global user similarity context. We conduct extensive experiments on two real datasets. The empirical results demonstrate that the proposed Inf2vec model significantly outperforms the baselines.

Several interesting research problems exist for future exploration. First, users' social behaviors are influenced by other factors, such as topical features. It is interesting to develop some methods to model the topic-aware influence propagation. Second, the proposed Inf2vec is not limited to using random walks to generate context. We can investigate other approaches for context generation to incorporate more factors related to social influence.

ACKNOWLEDGMENT

This work was supported by MOE Tier-1 RG83/16, MOE Tier-1 RG31/17, and MOE Tier-2 MOE2016-T2-1-137 awarded by Ministry of Education Singapore, and a grant awarded by Microsoft. Any opinions, findings, and conclusions in this publication are those of the authors, and do not necessarily reflect the views of the funding agencies.

REFERENCES

[1] D. Kempe, J. M. Kleinberg, and E. Tardos, "Maximizing the spread of influence through a social network," in *SIGKDD*, 2003, pp. 137–146.
[2] K. Saito, R. Nakano, and M. Kimura, "Prediction of information diffusion probabilities for independent cascade model," in *KES*, 2008, pp. 67–75.
[3] A. Goyal, F. Bonchi, and L. V. S. Lakshmanan, "Learning influence probabilities in social networks," in *WSDM*, 2010, pp. 241–250.
[4] M. Gomez-Rodriguez, D. Balduzzi, and B. Schlkopf, "Uncovering the temporal dynamics of diffusion networks," in *ICML*, 2011, pp. 561–568.
[5] J. Tang, J. Sun, C. Wang, and Z. Yang, "Social influence analysis in large-scale networks," in *SIGKDD*, 2009, pp. 807–816.
[6] B. Zong, Y. Wu, A. K. Singh, and X. Yan, "Inferring the underlying structure of information cascades," in *ICDM*, 2012, pp. 1218–1223.
[7] M. Gomez Rodriguez, J. Leskovec, and A. Krause, "Inferring networks of diffusion and influence," in *SIGKDD*, 2010, pp. 1019–1028.
[8] S. Lamprier, S. Bourigault, and P. Gallinari, "Extracting diffusion channels from real-world social data: a delay-agnostic learning of transmission probabilities," in *ASONAM*, 2015, pp. 178–185.
[9] N. Barbieri, F. Bonchi, and G. Manco, "Topic-aware social influence propagation models," in *ICDM*, 2012, pp. 81–90.
[10] S. Bourigault, S. Lamprier, and P. Gallinari, "Representation learning for information diffusion through social networks: an embedded cascade model," in *WSDM*, 2016, pp. 573–582.
[11] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *SIGKDD*, 2014, pp. 701–710.
[12] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: large-scale information network embedding," in *WWW*, 2015, pp. 1067–1077.
[13] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *SIGKDD*, 2016, pp. 855–864.
[14] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *SIGKDD*, 2016, pp. 1225–1234.
[15] L. F. Ribeiro, P. H. Saverese, and D. R. Figueiredo, "struc2vec: Learning node representations from structural identity," in *SIGKDD*, 2017, pp. 385–394.
[16] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *ICLR*, 2013.
[17] T. Mikolov and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013, pp. 3111–3119.
[18] M. Eslami, H. R. Rabiee, and M. Salehi, "Dne: A method for extracting cascaded diffusion networks from social networks," in *SocialCom*, 2011, pp. 41–48.
[19] F. Bonchi, "Influence propagation in social networks: a data mining perspective," *IEEE Intelligent Informatics Bulletin*, vol. 12, no. 1, pp. 8–16, 2011.
[20] W. Chen, L. V. Lakshmanan, and C. Castillo, "Information and influence propagation in social networks," *Synthesis Lectures on Data Management*, vol. 5, no. 4, pp. 1–177, 2013.
[21] A. Goyal, F. Bonchi, and L. V. S. Lakshmanan, "A data-based approach to social influence maximization," *PVLDB*, vol. 5, no. 1, pp. 73–84, 2011.
[22] Z. Yang, W. W. Cohen, and R. Salakhutdinov, "Revisiting semi-supervised learning with graph embeddings," in *ICML*, 2016, pp. 40–48.
[23] F. Morin and Y. Bengio, "Hierarchical probabilistic neural network language model," in *AISTATS*, 2005, pp. 246–252.
[24] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Adv. Artificial Intellegence*, pp. 421 425:1–421 425:19, 2009.
[25] K. Lerman and R. Ghosh, "Information contagion: an empirical study of the spread of news on digg and twitter social networks," in *ICWSM*, 2010, pp. 90–97.
[26] M. Cha, A. Mislove, and K. P. Gummadi, "A measurement-driven analysis of information propagation in the flickr social network," in *WWW*, 2009, pp. 721–730.
[27] H. Li, S. S. Bhowmick, and A. Sun, "Casino: towards conformity-aware social influence analysis in online social networks," in *CIKM*, 2011, pp. 1007–1012.
[28] S. Feng, G. Cong, B. An, and Y. M. Chee, "Poi2vec: Geographical latent representation for predicting future visitors," in *AAAI*, 2017.
[29] X. Rong, "Word2vec parameter learning explained," in *arXiv preprint arXiv:1411.2738*, 2014.
[30] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," in *UAI*, 2009, pp. 452–461.
[31] L. van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, pp. 2579–2605, 2008.
[32] A. P. Bradley, "The use of the area under the roc curve in the evaluation of machine learning algorithms," *Pattern recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.
[33] T. Saito and M. Rehmsmeier, "The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets," *PloS one*, vol. 10, no. 3, 2015.
[34] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, "Arnetminer: extraction and mining of academic social networks," in *SIGKDD*, 2008, pp. 990–998.