Data-driven Influence Learning in Social Networks

Feng Wang[§], Wenjun Jiang[†], Guojun Wang[‡,*] Dongqing Xie[‡]
[§]*School of Information Science and Engineering, Central South University, Changsha 410083, China*
[†]*College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China*
[‡]*School of Computer Science and Educational Software, Guangzhou University, Guangzhou 510006, China*
**Correspondence to: csgjwang@gmail.com*

***Abstract*—How to model the influence diffusion process accurately is an open issue that it has attracted a lot of researchers in the field of social network analysis. The existing researches assume they have already owned the social graphs with edges labeled with the influence probability. However, the question of how to obtain these probability from social networks has been largely ignored. Thus, it is interesting to address the problem of how to model the influence diffusion based on the data of social graphs and action logs. This is the main problem we addressed in this paper, and our purpose is to solve the problem of seeds detection via the data-driven influence probability calculation. We consider the influence probability can be viewed as two parts of the influence strength and the influence threshold. For learning the influence probability, we propose a novel Data-driven Influence Learning (DIL) algorithm including three stages. The experimental results illustrate our algorithm performs better than other baselines in various datasets. In addition, our algorithm enables us to detect the seed sets in large social networks.**

***Keywords*-Influence learning; Online social networks; Data driven; Influence maximization**

I. Introduction

With the availability of tremendous social network data, researchers in various domains are attracted to the field of social network analysis. In this field, many problems like information propagation [4, 22], viral marketing [2, 7] and influence maximization [1, 3, 12] require an accurate and finer grained model of influence diffusion.

The problem of learning influence diffusion probability from the real data has been initiated and since that several variants have been proposed [7, 11]. They supposed the infection was binary, and the diffusion network was unknown. In addition, influence relationships did not depend on the propagated content, and infection probabilities between users did not vary in time. Thus, in this paper we suppose that the process of influence diffusion is also binary, which contains the probability of launcher initiates the influence and the probability of receiver affected by the influence.

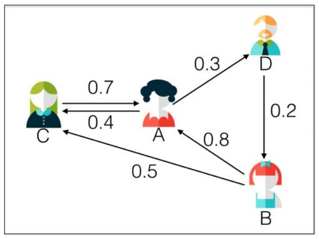
Recently, there are several studies [4-10] addressed the problem of influence probability learning. Saito et al. [4] focused on the IC model and presented a method for predicting diffusion probabilities from a log of past propagations by using the EM algorithm. Goyal et al. [5] proposed models and algorithms for learning the model parameters and for testing the learned models to make predictions. Goyal et al. [8] estimated influence spread by exploiting historical data, thus avoiding the need for learning influence probabilities, and avoiding the costly Monte Carlo simulations, which is the standard way to estimate influence spread.

However, how to model the influence diffusion based on the real data is still an open issue. There are several disadvantages in the existing approaches as follows: (1) While the existing approaches supposed the social graphs with edges labeled with influence probability, we consider the influence probability is not exactly enough to model the influence diffusion process. In this paper, we divide the process of influence diffusion into two parts: the launcher (influence strength) and the receiver (influence threshold), which can generate an accurate and finer grained influence diffusion model. (2) The question of how to calculate these influence strength and threshold from social networks' data has been largely ignored; (3) The influence relationships are hard to detect and characterize, especially the dynamic property. In addition, the action log is huge, any algorithm developed for learning and testing the influence models should make the minimal number of scans over the data.

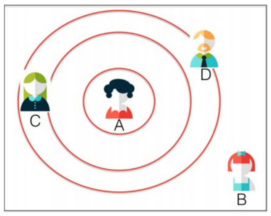
To address the challenges above, we propose a novel approach to learn the influence probability. The main steps are presented as follows: first we calculate the influence strength between each pairwise users via the action logs and social graphs. Then based on the data of diffusion episode, we project users' feature to the latent space vector through represent learning approach. We transform the classical influence diffusion paradigm to an influence continuous space as Figure 1, and we use the obtained distance between users as the threshold of users' influence probability. Finally, we determine the influence diffusion process through the comparison of influence strength and threshold. To summarize, our main contributions are as follows:

- We propose a novel data-driven influence learning approach to simulate the influence diffusion process. The idea of considering the influence diffusion process into two parts may be the first attempt in the field of influence diffusion.
- Based on the social network data sets, we calculate the influence strength between each pairwise users via the

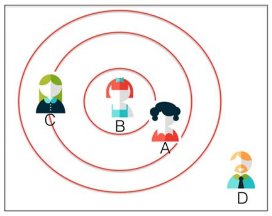
0-7695-6329-5/17/31.00 ©2017 IEEE
DOI 10.1109/ISPA/IUCC.2017.00177



(a) Classical influence diffusion graph



(b) Relative position for user A



(c) Relative position for user B

Figure 1. (a) presents a classical influence graph. The values associated to edges represent the influence probability between users. (b) shows a relative positions for user A through representation learning approach. (c) provides another relative positions for user B in the diffusion center. The difference is the person in the center who launched the diffusion.

action logs and social graphs. We adopt the discrete time model based on Bernoulli distribution.

- Based on the data of diffusion episode, we transform the classical influence diffusion paradigm to an influence continuous space via the representation learning approach, and we transform the distance between users into the threshold of influence probability.
- We conduct the experiments on the social network data sets. We calculate the range of influence spread on various data sets, and compare the running time with other existing algorithms. The experimental results demonstrate the effectiveness and efficiency of our data-driven influence learning algorithm.

The rest of this paper is organized as follows: we review the related works in section 2. We present the details of data-driven influence learning algorithm in section 3. We present the experimental results in section 4. Finally, we conclude our work and provide the future direction in section 5.

II. Related works

A. Influence probability learning

The problem of influence diffusion in social network has been widely studied through various algorithms [4-10]. The previous approaches typically assume that we have the social graphs with edges labeled with influence probability. Recently, a few works focus on estimating the influence probabilities based on the real world data sets. Goyal et al. [5] proposed models and algorithms for learning the model parameters and for testing the learned models to make predictions. Cao et al. [6] studied the influence maximization problem under the linear threshold model with unknown model parameters. Wang et al. [9] modeled the information diffusion through analyzing the multipolar influence.

Saito et al. [4] focused on the IC model and defined the likelihood for multiple episodes. They presented a method for predicting diffusion probabilities from a log of past propagations. Goyal et al. [8] estimated influence spread by exploiting historical data, thus avoiding the need for learning influence probabilities through costly Monte Carlo simulations. Saito et al. [10] estimated parameters (diffusion probability and time-delay parameter) of the probabilistic model as a function of the node attributes from the observed diffusion data. However, none of these researches try to model the finer grained influence diffusion model. In this paper, we consider the influence diffusion process into two parts of receiver and launcher, which correspond to the influence threshold and influence strength.

B. Embedding cascade model

Embedding cascade model has been a growing topic in the field of online social networks [14]. Representation learning has been used in a series of applications involving diffusion episode or more generally dynamic data. In [17], the authors presented a new method for learning node representations into a latent space, common to all the different node types. A typical representation learning method of Max-Margin DeepWalk learned low-dimensional representations for vertices in the social networks [18]. In [19], node2vec was proposed, an algorithmic framework for learning continuous feature representations for nodes in networks. Recently, a Structural Deep Network Embedding method was proposed, namely SDNE [20]. More specifically, they first proposed a semi-supervised deep model, which has multiple layers of non-linear functions, thereby being able to capture the highly non-linear network structure.

While our problem differs from the existing works that diffusion cascades spread as tree structures, not straight sequences, with possibly simultaneous infections. Perozzi et al. [15] proposed a novel approach named DeepWalk, learning latent representations of vertices in a network. These latent representations encode social relations in a continuous vector space. Cao et al. [16] proposed a novel model for learning vertex representations of weighted graphs. This model learned low dimensional vectors to represent vertices appearing in a graph. However our goal is different with these existing works. Our algorithm focus on modeling the accurate and finer grained influence diffusion process.

III. Data-driven Influence Learning algorithm

In this section, we provide the details of our data-driven influence learning algorithm. It includes three stages: first we calculate the influence strength between each pairwise of users via the action logs and social graphs. Based on the data of diffusion episode, we project users' feature to the latent space vector through the represent learning approach. We transform the classical influence diffusion paradigm into an influence continuous space, and we use the calculated distance between users as the threshold of influence probability. Finally, we determine the influence diffusion through the comparison of influence strength and threshold between each pairwise of users.

A. *Influence strength learning*

The social networks data provide social graphs and action logs. We define the social graphs $G = (V, E, T)$, where V denotes users, E denotes the edge $(u, v) \in E$ between user u and v, and T denotes the timestamp on which time the edge was created. The action logs provide us the relation actions as $(user, action, time)$. We use a tuple (u, a, t_u) to indicate user u perform action a at time t_u. Obviously, the link between social graphs and action logs is the user in the action tuple, which corresponds to the node in the graph. We use the notation $\mathcal{A}$ to denote all actions. We use A_u to denote the number of actions performed by user u, and $A_{u|v}$ denotes the number of actions performed by user u or v. We use A_{u2v} to denote the actions propagated from user u to v.

Based on the action logs, we define the **action propagation** as an action $a \in \mathcal{A}$ can propagate from user u to v if: (1) $(u, v) \in E$; (2) the action time t_u and t_v in the action tuple satisfies: $t_u < t_v$; (3) the timestamp on the edge $T(u, v) < t_u$. In other words, user u performs the action after user v did, and meanwhile after the time of the edge (u, v) was created. When these three conditions are satisfied, we deem the action propagation $prop(a, u, v, \Delta t)$ has happened, where $\Delta t = t_u - t_v$. It is need to be emphasized that the action a should performed after the moment the edge between user u to v was created. Through the action propagation, we can obtain the **propagation graph** as $PG(a) = (v(a), e(a))$. $v(a)$ denotes the action $a \in \mathcal{A}$, and $e(a)$ is an edge between user u to v whenever $prop(a, u, v, \Delta t)$.

The action logs (u, a, t_u) can be seen as a collection of propagations. When a user performs an action, we say that it is activated. Once a user activates, it becomes contagious and cannot de-activate. In addition, it may influence its inactive neighbor users. In this paper, we model the influence probability as the influence strength and threshold. The problem we tackle in this section is how to learn the influence strength and threshold among the users through mining the available set of past propagation logs.

1) Influence strength learning framework: We define S as the set of its active neighbors. Suppose an inactive user u can be activated by each neighbor $v \in S$ after user u and v became neighbors. We need to calculate the probability $p_u(S)$ to predict whether user u will be activated. We suppose the joint influence probability of S on user u as the general threshold model. If the influence probability is higher or equal to the threshold $p_u(S) \geq \theta_u$, we can conclude that user u is activated. We assume that the probability of neighbors influencing user u are independent. Thus the probability $p_u(S)$ can be calculated as follows:

$$p_u(S) = 1 - \prod_{v \in S}(1 - p_{v,u}) \tag{1}$$

We should notice that the individual influence probabilities $p_{v,u}$ are static. Hence, it can ensure the joint influence probability is monotone ($p_u(S) \leq p_u(T)$, whenever $S \subseteq T$) and submodular ($p_u(S \cup \{w\}) - p_u(S) \geq p_u(T \cup \{w\}) - p_u(S)$, whenever $S \subseteq T$).

To calculate the joint influence probability, we should computer the individual influence probability first. For the purpose of incrementally and efficiently calculation, we adopt the discrete time model based on Bernoulli distribution to capture the influence probability $p_{v,u}$.

The activated user v has a fixed probability to influence its inactive neighbor u. If the neighbor u becomes activated, it is a success attempt and user u can continue to influence his neighbors. Each attempt of action can be viewed as a Bernoulli trial. The influence probability of user v on u can be calculated as the ratio of number of successful attempts over the total number of trials. Thus the influence probability can be calculated as follows:

$$p_{v,u} = \frac{A_{v2u}}{A_v} \tag{2}$$

To simulate the finer grained and accurate influence diffusion process, we suppose that in the discrete time model, after user v performs an action, the influence probability remains constant at $p_{v,u}$ for a time window of $t_{v,u}$. After that it becomes 0, in other words, user v can influence u in the time interval $[t_v, t_v + t_{v,u}]$. This can allow us to calculate the influence probability incrementally. So we need to modify the definition of S so that it affects only contagious neighbors of user u. Thus, when a contagious neighbor w becomes non-contagious, we need to update $p_u(S)$ as follows:

$$p_u(S \backslash w) = \frac{p_u(S) - p_{w,u}}{1 - p_{w,u}} \tag{3}$$

The notation $t_{v,u}$ denotes the average time delay that defined as follows

$$t_{v,u} = \frac{\sum_{a \in A}(t_u(a) - t_v(a))}{A_{v2u}} \tag{4}$$

where $t_u(a)$ denotes the time of user u performs action a, and A denotes the set of action in the training data.

2) Influence strength learning algorithm: We input the social graphs and action logs to execute the influence strength learning algorithm, and the action logs are sorted on the ID of action and tuples on an action are chronologically ordered. This can allow algorithm to process an action at a time. Thus our algorithm can learn the influence probability in no more than two scans of the action logs. The details of first scan is presented in Algorithm 1.

Algorithm 1 The first scan of action log.

```
Input:
    social graphs $G$, action log $\mathcal{A}$.
Output:
    $A_{v2u}$, $t_{v,u}$, the updated $current\_table$
 1: for each action $a$ in action log do
 2:   $current\_table = \varnothing$.
 3: for each tuple $(u, a, t_u)$ in chronological order do
 4:   increment $A_u$, $parents = \varnothing$
 5: for each user $v : (v, a, t_v) \in current\_table \&\& (v, u) \in E^{t_v}$ do
 6:   if $t_u > t_v$ then
 7:     increment $A_{v2u}$
 8:     update $t_{v,u}$
 9:     insert $v$ in $parents$
10:     add $(u, a, t_u)$ to $current\_table$
```

The Algorithm 1 maintains the tuples for the current action a in a $current\_table$ indexed by user ID. When it reads a new tuple (u, a, t_u), the algorithm checks the $current\_table$ for the neighbors of user u, whether the link between user u and v has been established before either of them perform action a. The notation E^{t_v} denotes the set of edges at time t_v. We use the condition $t_u > t_v$ in line 6 to ensure user u performed the action a after user v did. Then through the first scan of action log, we can obtain the required parameters and updated $current\_table$.

Algorithm 2 The second scan of action log.

```
Input:
    social graphs $G$, action log $\mathcal{A}$, $t_{v,u}$.
Output:
    $p_{v,u}$, the updated $current\_table$
 1: for each action $a$ in action log do
 2:   $current\_table = \varnothing$.
 3: for each tuple $(u, a, t_u)$ in chronological order do
 4:   increment $A_u$, $parents = \varnothing$
 5: for each user $v : (v, a, t_v) \in current\_table \&\& (v, u) \in E^{t_v}$ do
 6:   if $0 < t_u - t_v < t_{v,u}$ then
 7:     increment $A_{v2u}$
 8:     insert $v$ in $parents$
 9:     if $parents \neq \varnothing$ then
10:     update $p_{v,u}$
11:     add $(u, a, t_u)$ to $current\_table$
```

In order to learn the influence probability $p_{v,u}$, we need to do the first scan to obtain the required parameters. Algorithm 2 presents the second scan of action log, if $0 < t_u - t_v < t_{v,u}$, the influence probability $p_{v,u}$ can be updated whenever at least one neighbor of user u is influenced.

B. Embeddings for influence diffusion

Based on the notations and parameters in section 3.1, through the action logs, we can obtain the influence diffusion episodes $\mathcal{D} = (D_1, D_2, \cdots, D_n)$. The diffusion episode denotes the set of timestamped user actions: $D = \{(u, t^D(u)) | u \in V \wedge t^D(u) < \infty\}$. $t^D(u)$ is the infection timestamps for users first infected by the influence diffusion, and ∞ denotes the users are not infected.

1) Diffusion model: We propose to embed the general threshold model in a continuous space to capture the relationship between users. Based on the relative positions, the model allows one to simulate diffusion processes that comply to the observed dynamics. The influence strength for a user infects his neighbors is calculated as Equation (1). The notation $p(u, v)$ denotes the influence strength of transmission from user u to v. Then we propose a function of users latent representation $f : R^d \times R^d \to [0, 1]$ as follows:

$$p_{u,v} = f(z_u, \omega_v) \tag{5}$$

where d is the size of the latent vector space, $z_u \in R^d$ represents user u as the sender of influence, $\omega_v \in R^d$ denotes user v as the receiver of influence.

Recall the individual influence probabilities are independent, we need the function f provides a value in the range $[0, 1]$, so that it can be viewed as a probability, the calculation method is presented as follows:

$$f(z_u, \omega_v) = \frac{1}{1 + exp(z_u^{(0)} + \omega_v^{(0)} + \sum_{i=1}^{d-1} (z_u^{(i)} - \omega_v^{(i)})^2)} \tag{6}$$

where $z_u^{(i)}$ and $\omega_v^{(i)}$ denote the i-th component of latent vector. This method transmits the influence threshold as the distance between user u and v. The function f returns a real value in [0, 1] which acts as a consistent probability that can directly be viewed as the influence probability threshold. Figure 1 depicts our proposal in this section, transforming the classical influence diffusion paradigm to an influence continuous space. Moreover, it is the foundation part of seed detection in our approach. Note that we use $z_u^{(0)}$ and $\omega_v^{(0)}$ as the priors on the general tendency to infect other users. In addition, we notice that the same distance between two users does not means the same influence probability. It should be determined by who are the users in the concern.

2) Learning algorithm: The learning algorithm follows the method of maximization likelihood algorithm. The probability threshold of observing an influence diffusion episode is calculated as follows:

$$p(D) = \prod_{v \in D(\infty)} p_v^D \prod_{v \in \overline{D}(\infty)} (1 - p_v^D) \tag{7}$$

where $p^D = p(t^D)$ is the probability of being infected in diffusion episode D, user v denotes the neighbors. Thus, we consider the following log-likelihood for all diffusion episodes as follows:

$$\mathcal{L}(p;\mathcal{D}) = \sum_{D\in\mathcal{D}} (\sum_{v\in D(\infty)} log(p_v^D) + \sum_{v\in\overline{D}(\infty)} log(1-p_v^D)) \tag{8}$$

We use the Expectation-Maximization (EM) algorithm to solve this optimization problem, and obtain the distances between users in the vector space.

3) Influence diffusion process: Through the calculation process above, we use the distances between users as the threshold of users' influence probability. And from section 3.1, we can obtain each user's influence strength based on the real data of act logs. In our paper, we propose the simplest way to decide whether the influence diffusion can happen or not. If the influence strength is high or equal to the threshold of users' influence probability, the influence diffusion can happen between users. Hence, the details of data-driven influence learning algorithm is presented in Algorithm 3.

Algorithm 3 Data-driven influence learning.

```
Input:
    social graphs $G$, action log $\mathcal{A}$, diffusion episode $D$.
Output:
    $p_{v,u}$, $f(z_u, \omega_v)$
1: for each action $a$ in action log do
2:     updating time delay $t_{v,u}$ and $current\_table$ in Algorithm 1.
3:     calculating the influence strength $p_{v,u}$ in Algorithm 2.
4: for each diffusion episode $D$ in social network do
5:     calculating the influence probability threshold (be viewed as distance) $f(z_u, \omega_v)$ via Eq. (6).
6: If $p_{v,u} \geq f(z_u, \omega_v)$, then the influence diffusion can happen between users.
7: Else the influence diffusion stops.
```

Complexity analysis. We use the notation R to denote the number of influence spread simulation, n and m denote the number of users and edges in the social network, respectively. The step of calculating the influence strength in Algorithm 1 and 2 is $O(n)$. The step of calculating the influence threshold is $O(m)$. Thus, the time complexity of DIL algorithm is $O(R(n + m))$.

IV. Experiments

We conduct our experiments on three online social networks. All algorithms are implemented in Python, and run on a 3.20 GHz quad core machine with 4.00 Gb memory.

A. Datasets

We conduct our experiments on three online social networks (Twitter, Digg and Sina Weibo). In these datasets, diffusion episodes with less than 3 users are removed. These datasets essentially consist of triplets of the form: $(user, item, timestamp)$. We use the contents that sent or reposted by the users to denote the items. Each triplet indicates that a given user interacted with other users at a known time by interacting with the content item. Each item corresponds to a diffused content, which infects every user who interacted with it. Table 1 gives some statistics about the datasets. In this table, we reports the number of users, links, episodes (training and test), and the average length of diffusion episodes.

Table I
Statistics of the online social networks dataset

Dataset	Twitter	Digg	Sina Weibo
Nodes	2976	3298	63641
Edges	891252	689324	1391718
Training Episodes	10000	16000	18000
Test Episodes	1000	1000	1000
Avg Episodes size	20.7	3.42	23.1

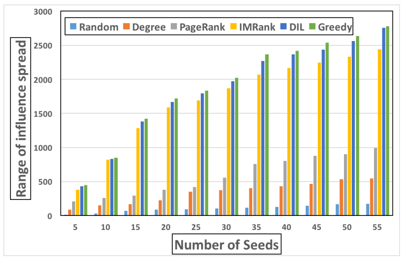
B. Baselines

The influence spread classic model of Independent Cascade (IC) mode and Linear Threshold (LT) model are both considered in our experiments [1]. We compare the existing algorithms with our DIL algorithm based on the IC and LT model, respectively. The comparison algorithms are presented as follows: (1) **Random** algorithm selects top-k users as seeds randomly. (2) **Degree** algorithm selects influential users as seeds based on the degree centrality, and it is a heuristic algorithm [21]. (3) **Greedy** algorithm selects seeds through maximizing the margin influence spread iteratively [1]. (4) **PageRank** algorithm chooses the seeds via the network structure, selecting the k highest-ranked users as seed set [22]. (5) **IMRank** algorithm accelerates the process of seeds detection through the ranking strategy [23].

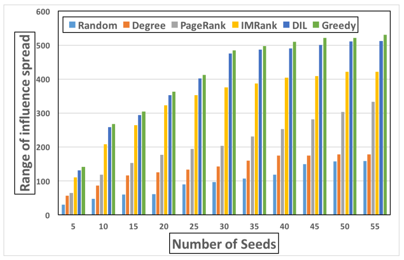
C. Influential users detection

In the existing algorithms, the influence spread estimation is based on the influence probability p_{uv} associated with each edge (u, v), and we set the threshold to be 0.35 in LT model. The other parameters are calculated through these algorithms, respectively.

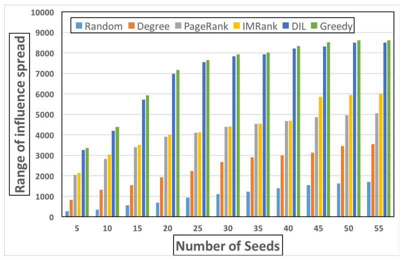
Our DIL algorithm computers the influence strength through the social graphs and action logs. Then in the part of influence threshold calculation, training is performed on the training data set by maximizing the likelihood. This allows us to learn the model parameters of representations in our algorithm. The DIL algorithm is generative so that once the model has been trained, it can be used for influence prediction tasks on the test data set. And in the influential users detection process, we need to avoid the common neighbors between users. Thus, we compare the common neighbors in each pairwise users so that the seed set can have the least common neighbors.



(a) Twitter

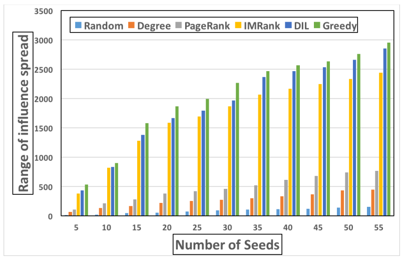


(b) Digg

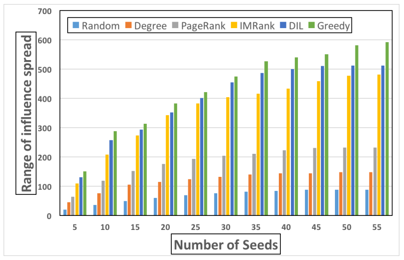


(c) Sina weibo

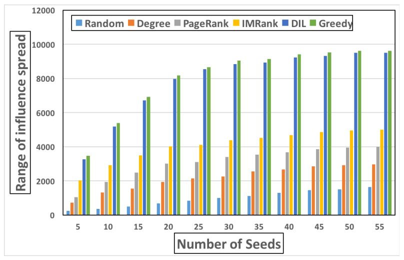
Figure 2. Comparison of influence spread range under IC model



(a) Twitter



(b) Digg



(c) Sina weibo

Figure 3. Comparison of influence spread range under LT model

D. Experimental results

Influence spread. We set 2000 simulation after each round of adding a new influential user to the seeds, and take the newly influence spread as the spread of updated seeds. We conduct the comparison experiments to measure how many users can be affected by the obtained seeds. We set the size of seeds $k = (5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55)$. We use the IC and LT model to simulate the influence spread process, respectively.

Figure 2 and Figure 3 present the influence spread range of different algorithms on the three datasets, and the comparison experiments are conducted on different seed size k and diffusion model. The performance of algorithms on our datasets are similar. Our DIL algorithm performs better than other four algorithms (Random, Degree, PageRank and IMRank). These algorithms are the representation of heuristic, network structure and hybrid algorithm. The experimental results illustrate that it is better to simulate the influence diffusion process based on the real data. These experimental results prove our algorithm is more effective than the existing heuristic and hybrid algorithms. Although our method performs a little worse than the greedy algorithm, but we should make the trade-off between performance and scalability. Due to the costly Monte Carlo simulations, the greedy algorithm suffers serious scalability problem, so our algorithm has higher availability in real applications.

Running time. We compare the running time of the DIL algorithm with other five algorithms. In this part, we select the size of seed set $k = 55$ under the IC model and LT model, respectively. As shown in Table 2 and 3, we know that there exists huge difference in the aspect of running time. The Random algorithm can finish in just a few seconds, and even almost instantly in small-size dataset, but the results of influence spread is terrible. Although the greedy algorithm has the best influence spread performance, but it suffers serious scalability problem in large size dataset (almost 2 days). Since the Degree algorithms is the heuristic algorithm, the running time is faster than ours. But comparing to other three algorithm (Greedy, PageRank, and IMRank), our running time is better. In addition, in the large size dataset of Sina weibo, our algorithm can complete the calculation in about 10 mins. Thus, our algorithm can scalable to the large size of dataset.

Table II
RUNNING TIME COMPARISON IN IC MODEL

Dataset	Random	Greedy	Degree	PageRank	IMRank	DIL
Twitter	3s	N/A	105s	296s	187s	104s
Digg	<1s	N/A	32s	40s	54s	36s
Sina Weibo	5s	N/A	457s	643s	708s	599s

Table III
RUNNING TIME COMPARISON IN LT MODEL

Dataset	Random	Greedy	Degree	PageRank	IMRank	DIL
Twitter	5s	N/A	130s	302s	193s	124s
Digg	2s	N/A	44s	84s	67s	55s
Sina Weibo	9s	N/A	453s	655s	723s	603s

From the experimental results above, we conclude that our DIL algorithm is more scalable to large size of social networks and with the guaranteed accuracy. These results also prove the essential of considering the influence probability as the parts of influence strength and threshold.

V. Conclusion

We propose a data-driven influence learning algorithm to enable influence maximization in large size of social networks in this paper. We consider the influence probability into two parts of influence strength and threshold, and we use the real-world data of social graphs, action logs, and diffusion episodes to obtain these two parameters. Then we use the calculation results to solve the problem of influence maximization. Experimental results demonstrate our DIL algorithm is efficient and scalable.

In the future research, we will try to model the process of influence spread more close to the real-life situation. Besides, we only simple use the approach of representation learning, we will study the relationship between node's features and influence spread via network representation learning algorithm.

Acknowledgment

This work is supported in part by the National Natural Science Foundation of China under Grants 61632009, 61472451 and 61502161, in part by the Guangdong Provincial Natural Science Foundation under Grant 2017A030308006 and High-Level Talents Program of Higher Education in Guangdong Province under Grant 2016ZJ01.

References

1. Kempe, D., Kleinberg, J., Tardos, E. Maximizing the spread of influence through a social network. In: KDD, pp. 57-66, August 2003.
2. Chen, W., Wang, C., Wang, Y. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In: KDD, pp. 1029-1038, July 2010.
3. Chen, W., Wang, Y., Yang, S.: Efficient influence maximization in social networks. In: KDD, pp. 199-208, June 2009.
4. Saito, K., Nakano, R., Kimura, M. Prediction of information diffusion probabilities for independent cascade model. In: KES 2008, pp. 67-75.
5. Goyal A, Bonchi F, Lakshmanan L V S. Learning influence probabilities in social networks. WSDM 2010, New York, Ny, USA, February. DBLP, 2010:241-250.
6. Cao T, Wu X, Hu T X, et al. Active Learning of Model Parameters for Influence Maximization. Machine Learning and Knowledge Discovery in Databases. Springer Berlin Heidelberg, 2011:280-295.
7. K. Saito, R. Nakano, and M. Kimura. Prediction of information diffusion probabilities for independent cascade model. In KES 08, pages 67-75. Springer-Verlag, 2008.
8. Goyal A, Bonchi F, Lakshmanan L V S. A Data-Based Approach to Social Influence Maximization. Proceedings of the VLDB Endowment, 2011, 5(1):2011: 73-84.
9. Wang W, Peng Z, Liu Z, et al. Learning the Influence Probabilities Based on Multipolar Factors in Social Network. International Conference on Knowledge Science, Engineering and Management. Springer, Cham, 2015: 512-524.
10. Saito K, Ohara K, Yamagishi Y, et al. Learning Diffusion Probability Based on Node Attributes in Social Networks. Foundations of Intelligent Systems International Symposium, Ismis 2011. DBLP, 2011:153-162.
11. D. Gruhl, R. Guha, D. Liben-Nowell, and A. Tomkins. Information diffusion through blogspace. In WWW 04, pages 491-501, New York, NY, USA, 2004. ACM.
12. Wang F, Jiang W, Li X, et al. Maximizing positive influence spread in online social networks via fluid dynamics. Future Generation Computer Systems, 2017. DOI: http://dx.doi.org/10.1016/j.future.2017.05.050
13. Wang F, Li J, Jiang W, et al. Temporal Topic-Based Multi-Dimensional Social Influence Evaluation in Online Social Networks. Wireless Personal Communications, 2017:1-2995(3), 2143-2171.
14. Y. Bengio, A. C. Courville, and P. Vincent. Representation learning: A review and new perspectives. IEEE Trans. Pattern Anal. Mach. Intell., 35(8):1798-1828, 2013.
15. Perozzi B, Al-Rfou R, Skiena S. Deepwalk: Online learning of social representations. In: KDD. ACM, 2014: 701-710.
16. Cao S, Lu W, Xu Q. Grarep: Learning graph representations with global structural information. In CIKM. ACM, 2015: 891-900.
17. Jacob Y, Denoyer L, Gallinari P. Learning latent representations of nodes for classifying in heterogeneous social networks. In: WSDM. ACM, 2014: 373-382.
18. Tu C, Zhang W, Liu Z, et al. Max-Margin DeepWalk: Discriminative Learning of Network Representation. IJCAI. 2016: 3889-3895.
19. Grover A, Leskovec J. node2vec: Scalable feature learning for networks. In: KDD. ACM, 2016: 855-864.
20. Wang D, Cui P, Zhu W. Structural deep network embedding. In: KDD. ACM, 2016: 1225-1234.
21. Qin Y, Ma J, Gao S. Efficient influence maximization under TSCM: a suitable diffusion model in online social networks. Soft Computing, 2017, 21(4):827-838.
22. Yang Y, Yang Y, Yang Y, et al. An Influence Propagation View of PageRank. ACM Transactions on Knowledge Discovery from Data, 2017, 11(3):30.
23. Cheng S, Shen H, Huang J, et al. IMRank:influence maximization via finding self-consistent ranking. In: SIGIR. ACM, 2014:475-484.