Int J Multimed Info Retr
DOI 10.1007/s13735-016-0102-y

REGULAR PAPER

Learning content–social influential features for influence analysis

Na Zhao[1] · **Hanwang Zhang**[1] · **Meng Wang**[2] · **Richang Hong**[2] · **Tat-Seng Chua**[1]

Received: 21 April 2016 / Revised: 1 June 2016 / Accepted: 6 June 2016

Abstract We address how to measure the information propagation probability between users given certain contents. In sharp contrast to existing works that oversimplify the propagation model as predefined distributions, our approach fundamentally attempts to answer why users are influenced (e.g., by content or relations) and whether the corresponding influential features (e.g., hidden factors) can be inferred from the propagation in the entire network. In particular, we propose a novel method to deeply learn the unified feature representations for both user pair and content, where the homogeneous feature similarity can be used to estimate the propagation probability between users with given content. The features are dubbed content–social influential feature since we consider not only the content of the propagation information but also how it propagates over the social network. We design a fast asynchronous parallel algorithm for the feature learning. Through extensive experiments on a real-world social network with 53 million users and 838 million tweets, we show significantly improved performance as compared to other state-of-the-art methods on various social influence analysis tasks.

Keywords Social network · Social influence · Influence prediction · Representation learning

✉ Na Zhao
zhaona@nus.edu.sg

Hanwang Zhang
hanwangzhang@gmail.com

Meng Wang
eric.mengwang@gmail.com

Richang Hong
hongrc@hfut.edu.cn

Tat-Seng Chua
chuats@comp.nus.edu.sg

[1] School of Computing, National University of Singapore, 13 Computing Drive, 117417 Singapore, Singapore

[2] School of Computer and Information, Hefei University of Technology, 198 Tunxi Road, Hefei 230009, Anhui, China

1 Introduction

In online social networks, where word-of-mouth diffusion [1,11] has affected how we collect and disseminate information, social influence plays a crucial role in various applications such as viral marketing [17,27] and recommender systems [33]. Take viral marketing as an example, first user influence is measured and then a small number of key influential users are found. They are expected to trigger a cascade of influence in the social network, achieving a large-scale chain-reaction of influence at a small marketing cost. Thus, social influence analysis offers a huge opportunity for marketers, advertisers and politicians to design economic and effective campaign strategies in social networks.

Sociologists generally define social influence as chan-ges in an individual's thoughts, feelings, attitudes, or behaviors that result from interactions with another individual or group [26]. However, computer scientists are often more interested in how to quantitatively measure the "interactions" and the "changes". Current research efforts generally fall into the following three directions. One is based on heuristic statistics. For example, the number of followers of a user can be recognized as an indication of the user's potential to influence others [6]. However, such statistics are too naive to meet with the dynamics of users' social behavior and interests. The second is based on network structure learning [9,12,28,32], which infer diffusion network or even estimate diffusion probabilities. However, these methods infer influence only

Published online: 27 June 2016

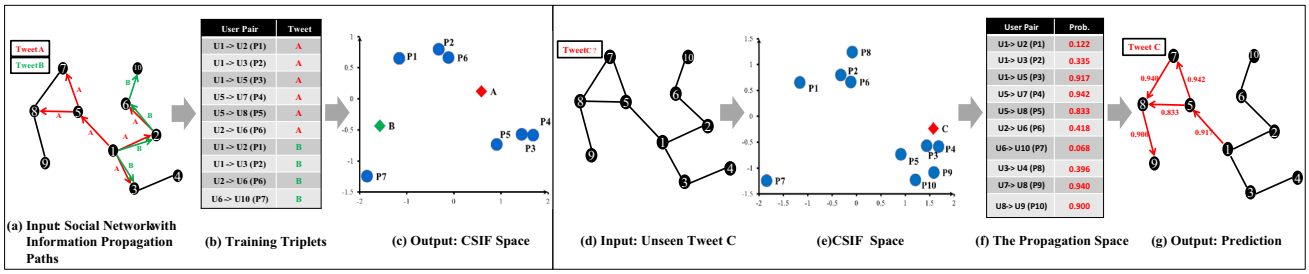


Fig. 1 Our proposed approach learns the latent representations for directed user pair and tweet in information propagation. The learned representations encode propagating relations and content. Here, given a social network labeled with propagating paths of two tweets, our method is used to generate the corresponding latent representations in R^d (in this example, $d = 2$)

from the network structure and influenced cascades, while ignoring the valuable diffusion content. The third one is topic-aware social influence analysis [15,21,30,33], which exploit influence under a topic-aware perspective. But these methods usually fail to jointly analyse the content and network structure, since they require topic assignment as a precondition.

In this paper, we address to model the probability of whether user i is influenced by user j given a content (e.g., tweet). We argue that this task is fundamental since if one could accurately predict such probability, subsequent social network analysis such as propagation can be improved. In particular, we propose to learn social influence by jointly encoding the propagation relations (which can be roughly seen as social relations) and propagation content into a deep and homogeneous space, where the resultant representations are named content–social influential features (CSIF). The CSIF are vector representations of arbitrary user pair or content in this latent space, and any subset of the feature dimensions may correspond to latent factors that result in a certain social influence. With the latent space, the similarity between user pair CSIF and the content CSIF can be used to predict the influence probability. Figure 1 shows the overview of the proposed CSIF learning framework for predicting social influence. In the training stage, we take a social network and its history propagation paths as input. First, the propagation paths are collected as triplets consisting of directed user pairs and the corresponding content. Each triplet encodes a user–user influence record (Fig. 1b). Then, these triplets which together encode the social influence of the network are mapped into a continuous vector space-CSIF space, which preserve the user–user influence, i.e., the distance between the user pair vector and the tweet vector is closer if they can be found in the training triplets; and vice versa (Fig. 1c). In the predicting phase, given any unseen content or user pair, we predict their influence probability (Fig. 1f) by calculating the distance in the learned CSIF space (Fig. 1e). Since the CSIF space learned in the training stage preserves the social influence, the new incoming user pair and content are expected to reproduce the probability of whether the influence is happening. We design a fast asynchronous parallel algorithm for the training stage of our proposed approach.

To evaluate the effectiveness of the proposed approach in real scenarios, we conduct large-scale experiments on a real-world social network (i.e., Tencent microblog platform). Our data set is a 20-day snapshot of the whole network, which contains more than 53 million users and more than 838 millions tweets. The promising results demonstrate that (1) the proposed learning algorithm is efficient and scalable; and (2) the learned features are efficient and informative, which outperform traditional methods in fundamental tasks such as propagation user prediction and domain experts identification.

Our contributions are summarized as follows:

- We propose social influence features named content–social influential features for user–user interaction probability given a content information. This pairwise probability is fundamental for various social influence analysis tasks.
- We design a highly parallel and scalable algorithm, which can be applied in real-world large networks, for content–social influential features learning.
- Through extensive experiments on a real-world social network snapshot with 53 million users and 838 million tweets, the proposed approach shows significantly improvement in two fundamental social influence tasks as compared to other state-of-the-art methods.

The rest of our paper is arranged as follows. Related works are briefly summarized and discussed in Sect. 2. In Sect. 3, we give a formal definition of the problem addressed in this paper and technically introduce our proposed method in detail in Sect. 4. Section 5 outlines our experimental setting and demonstrates the evaluate results of our experiments. Finally we draw a conclusion in the last section.

2 Related work

2.1 Social influence analysis

Over past decades, sustained efforts have been contributed to influence analysis on social networks.

Information maximization Domingos and Richardson [7] defined the seed influencer selection problem as influence maximization. Kempe et al. [17] formulated influence maximization into the two well-known models: independent cascade (IC) and the linear threshold (LT), and proposed an efficient greedy algorithm based on the monotonicity and submodularity properties of the maximization function. Liu et al. [20] proposed an Influence Spread Path (ISP) technique to compute influence spread, which provides a structural representation that can speed up the calculation of influence spread. In addition, several advanced greedy algorithms [14,37] have been proposed to speed up seed influencer selection. A major limitation of the above research is that they are all based on the assumption that the edges of the network graph are represented by influence propagation probabilities. However, the question of where these probabilities come from and how they can be computed have been largely ignored until now. In our work, we address the question by seeking to measure the information propagation probabilities with respect to the propagation users and corresponding content. Moreover, unlike influence maximization problem whose target is to find a set of seed influential users [20,37], our work aims to analyse social influence no matter the influence is from an ordinary user or an influential user.

Statistics based influence analysis This line of research investigates which statistics is a proper measure for social influence. Cha et al. [6] measured user influence by calculating the number of actions of each user. They preformed comparisons between two action influences in Twitter: retweet and mention, and the indegree connections of users. They discovered that the indegree measure does not correlate well with the two action influences. Baskshy et al. [1] estimated influence of ordinary influential users based on the number of extended retweets. Li et al. [19] considered the publications of user as influence, which is then measured by using three node centrality metrics: degree, closeness and betweenness, which were formalized by Freeman [10]. These studies are only based on heuristic statistics, and hence these descriptions are somewhat lopsided and insufficient to analyze the complex social influence. Therefore, many research efforts are done by applying advanced statistical learning approaches on social networks rather than simple statistics. Zhang et al. [36] built an action-based user influence model extended from the PageRank algorithm. They exploited network dynamics by jointly taking influence involvement into considerations. In view of users' connectivity and communication activity in social network, Heidemann et al. [16] proposed an adapted PageRank approach to identify influential users in an undirected and weighted graph based on activity links. The above PageRank-based works aim to obtain a rank of users according to their influential power.

Network structure learning-based influence analysis This line of research investigates the structure of the social network to infer diffusion network or even estimate diffusion probabilities. For example, NetInfer, a scalable algorithm for inferring networks of diffusion and influence, was presented in [12]. It formulates the network structure learning problem as a submodular function maximization problem. Against the fixed transmission rate between all nodes in NetInfer, a representative approach—NETRATE is designed in [28]. It can estimate the transmission rates of infections between two connected nodes in addition to inferring the connectivity of the network. Wang et al. [32] further explored the multi-aspected connection of the network and multi-pattern cascades of the diffusion, and proposed a MMRate model to incorporate aspect-level user interactions and various diffusion patterns. However, these methods infer influence only from the network structure and influenced cascades, while ignoring the valuable diffusion content. In contrast, we conduct social influence analysis by exploiting propagation paths including user relations and corresponding information content, so our work is content–social aware.

Topic-aware social influence analysis Social influence are evidently influenced by the topics of propagation information, that is, it is topic-dependent. However, previous work rarely consider social influence from both social and content aspects. Only current several work turn to topic-specific influence analysis. For example, Weng et al. [33] integrated topic sensitivity into PageRank and propose a topic-sensitive influence measure named TwitterRank to quantify influence in Twitter. But they also neglect the question of where these probabilities come from and how to compute. Tang et al. [30] considered a social network and a prior topic distribution as input, then learned user-pair topic-special influence strength, which does not consider simultaneous learning of topics and influence, nor content propagation process. Liu et al. [21] presented a probabilistic model to jointly infer the topic distribution of labelled citation data in a citation network and influence strength of cited documents. But this work is designed for citation networks, where the generation of links purely depends on the content, hence it is not suitable for social networks. Barbieri et al. [2] proposed topic-aware extensions of LT and IC models in influence maximization, and jointly considered user interests, authoritativeness and topic relevance to model influence propagation. But they overlook influence user–user probability.

2.2 Social feature learning

To predict influence probabilities, our work presents to deeply learn the unified feature representations for propagation users and content. The idea is inspired by some novel feature learning works, which convert symbolic concepts such as words [22] and relations [5] into distributed representations (e.g., continuous vectors). For instance, Tang et al. [31] transforms the latent membership of social community of users into vectors. Their formulation reduces a graph matrix of the size of the whole network into a eigen-decomposition problem, and thus will fail in real-world larg-scale social networks. As an extension to large-scale problem, Perozzi et al. [24] consider a random walk on a social network as a context of a natural language sentence and apply the language feature learning method to map users into vectors. However, these two works are only capable of handling a static social network and are not applicable to the prediction of influence, which is dynamic in nature. To the best of our knowledge, our work is the first feature learning study on social influence. Technically, besides computational scalability, our formulation and algorithm resolves the problem of mapping out-of-sample users and tweets, and hence is predictive. In particular, we not only learn vectors for the user pairs and tweets, but also learn the transformation matrices which can map users and tweets from the original representations to the target feature vectors.

3 Problem definition

We denote the social network as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ denotes the users of the network and $\mathcal{E} \subseteq (\mathcal{V} \times \mathcal{V})$ represents the social relationships between users. For example, if a user u_j follows u_i, there is a edge $(u_i, u_j) \in \mathcal{E}$. Given any tweet t_k, we wish to model the influence that u_j will retweet t_k from u_i as the following conditional probability:

$$p\left(u_i \rightarrow u_j | t_k\right). \tag{1}$$

Note that $(u_i \rightarrow u_j) \neq (u_j \rightarrow u_i)$, indicating the directional property of social influence.

The input to our problem is the past information propagations (e.g., retweets) $\mathcal{S}$, which records the past propagation paths of information together with the corresponding propagation users:

$$\mathcal{S} = \{s_1, \ldots, s_m, \ldots, s_N | s_m = (u_i \rightarrow u_j, t_k)\}, \tag{2}$$

where $(u_i \rightarrow u_j, t_k)$ indicates that user u_i spreads a tweet t_k to u_j.

We consider the directed user pair $(u_i \rightarrow u_j)$ which indicates the propagation relation from one user to another, and its referenced tweet t_k as two entities (i.e., symbols) in social influence. We seek to encode $(u_i \rightarrow u_j)$ and t_k into a low-dimensional vector space, where the similarity between the resultant features reflects the probability of u_j retweet t_k from u_i. We name the low-dimensional representations that are distributed and continuous as content–social influential features (CSIF), and assume that any subset of the CSIF dimensions corresponds to some hidden factors of social influence (such as the number of followers/followees, the popularity of the topics, and even the interests of users). We argue that the CSIF can only be learned by considering the social network topology together with the topics of the tweets.

In particular, by denoting $\mathbf{u}_{ij}$ as user–user feature for $(u_i \rightarrow u_j)$ and $\mathbf{t}_k$ as tweet feature for t_k (cf. Sect. 5.1.1 for details), we learn two feature transformation matrices $\mathbf{U}$ and $\mathbf{T}$ that can map users and tweets into the desired CSIF space. By doing this, given any triplet, we can predict the influence probability in terms of CSIF. Our learning objective should be consistent with the fact that the desirable matrices should map $\mathbf{u}_{ij}$ and $\mathbf{t}_k$ closer if $(u_i \rightarrow u_j, t_k)$ exists in the training set and at the same time preserves topology of all the other triplets in the social network.

4 Learning content–social influential features

In this section, we detail our proposed algorithm for learning the content–social influential features. We first introduce the formulation of our model and then illustrate an efficient on-line learning algorithm. Finally, we show how to use the learned feature to compute the user–user influence probability and its induced global-level influence along the network.

4.1 Formulation

Our training objective is to maximize the likelihood (or minimize the negative log-likelihood) of any influence observation $(u_i \rightarrow u_j, t_k)$ in the training set $\mathcal{S}$,

$$J = -\sum_{i,j,k} \log p\left(u_i \rightarrow u_j | t_k\right). \tag{3}$$

In particular, we define $p\left(u_i \rightarrow u_j | t_k\right)$ using the softmax function:

$$p\left(u_i \rightarrow u_j | t_k\right) = \frac{\exp\left(\mathbf{u}_{ij}^T \mathbf{U}^T \mathbf{T} \mathbf{t}_k\right)}{\sum_{i',j'} \exp\left(\mathbf{u}_{i'j'}^T \mathbf{U}^T \mathbf{T} \mathbf{t}_k\right)}. \tag{4}$$

After minimizing the objective function J in Eq. (3), we can obtain the feature projection matrices $\mathbf{U}$ and $\mathbf{T}$ to predict the influential probability for any new triplet (u_i, u_j, t_k).

The denominator of Eq. (4) is the normalization term that assures $p\left(u_i \rightarrow u_j | t_k\right)$ to be a valid probability. Note that the validity does not only constrain the range of Eq. (4) to be [0, 1], but also endows discriminative ability for the objective function that encourages the distance between two CSIF, $\mathbf{U}\mathbf{u}_{ij}$ and $\mathbf{T}\mathbf{t}_k$, to be closer, while penalizing the distance between other CSIF (e.g., when $i' \rightarrow j' \neq i \rightarrow j$). In other words, the influence model in Eq. (4) indicates that the distance between the expected influential features should respect both user–user influence (i.e., larger value of a training triplet) and global-level influence (i.e., valid probability regarding the topology of propagation paths).

However, minimizing J is impractical because the computation cost of the normalization term in Eq. (4) is proportional to the number of user pairs $u_i \rightarrow u_j$, which is often very large. Take a typical social network with 10^6 users as an example, the number of retweet user pairs could be over 10^9 in one day. To achieve feasible computation, we use an effective approximation called Hierarchical Softmax, which is widely used in neural computation [23]. The main advantage is that instead of calculating large P pairs of users, we only need to calculate about $\log_2 P$ pairs, thanks to a tree representation.

Specifically, we assign pairs $u_i \rightarrow u_j$ to the leaves of a binary tree, and the probability estimation of $p\left(u_i \rightarrow u_j | t_k\right)$ is turned into a series of binary decisions from the root to the target leaf. Let $n_{ij}(m)$ be the m-th node on the path from the root to $u_i \rightarrow u_j$, and let L_{ij} be the length of this path. In particular, we have $n_{ij}(1)$ is root and $n_{ij}(L_{ij})$ is $u_i \rightarrow u_j$. In addition, we denote $lc(n)$ as the left child of node n and let $I(x)$ be an indicator function such that it is 1 if x is true and -1 otherwise. Then, we can define a hierarchical softmax version of $p\left(u_i \rightarrow u_j | t_k\right)$ as

$$h\left(u_i \rightarrow u_j | t_k\right) = \prod_{m=1}^{L_{ij}-1} \sigma\left(I\left(n_{ij}(m+1) = lc(n_{ij}(m))\right) \cdot \mathbf{a}_{n_{ij}(m)}^T \mathbf{T}\mathbf{t}_k\right), \tag{5}$$

where $\sigma(x) = 1/(1+\exp(-x))$ is the sigmoid function, which is widely used in binary-valued probabilities and $\mathbf{a}_{n_{ij}(m)}$ is an additional parameter corresponding to the node $n_{ij}(m)$. Intuitively, Eq. (5) interprets the mechanism that u_i influences u_j by tweet t_k is the result of a series of binary decisions, each of which is parameterized by an auxiliary feature $\mathbf{a}_{n_{ij}(m)}^T$. The higher the level of feature in the hierarchy, the more the number of $(u_i \rightarrow u_j, t_k)$ shares it, and that means the more fundamental the feature is. Note that this intuition is consistent with the essence of deep feature learning [22].

Although hierarchical softmax $h\left(u_i \rightarrow u_j | t_k\right)$ is a valid probability,[1] the parameter $\mathbf{U}$ is missing during the binary decisions and hence it violates the softmax definition in Eq. (4). We compensate this by building a binary tree for all the training triplets, i.e., we treat both $u_i \rightarrow u_j$ and t_k as the leaves. Since the swap of the multiplications of the features does not change the value of Eq. (4), we can also define another hierarchical softmax for $p\left(u_i \rightarrow u_j | t_k\right)$ as

$$h\left(t_k | u_i \rightarrow u_j\right) = \prod_{m=1}^{L_k-1} \sigma\left(I\left(n_k(m+1) = lc(n_k(m))\right) \cdot \mathbf{a}_{n_k(m)}^T \mathbf{U}\mathbf{u}_{ij}\right), \tag{6}$$

where $n_k(m)$ denotes the m-th node from the root to the target leaf t_k, and L_k denotes the length of this path. Therefore, the overall hierarchical softmax definition is

$$p\left(u_i \rightarrow u_j | t_k\right) = \frac{1}{2} h\left(u_i \rightarrow u_j | t_k\right) + \frac{1}{2} h\left(t_k | u_i \rightarrow u_j\right). \tag{7}$$

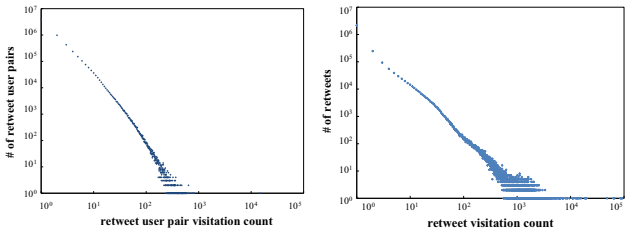
We can find that hierarchical softmax drastically reduces the computation from $\mathcal{O}(P)$ to $\mathcal{O}(\log_2(P))$ but also introduces significantly more auxiliary parameters of the size $\mathcal{O}(2P)$. However, as shown in many related studies, these additional parameters will even boost the generalization ability of the model [22,24]. As aforementioned, one possible reason is that because the parameters are hierarchically shared by different concepts (e.g., user pairs and tweets), the model inherits many excellences of deep feature learning [3].

4.2 Algorithm

The number of training triplets for a typical social network is often over millions or even billions. Therefore, traditional serial implementation of stochastic gradient descent for optimizing Eq. (3) is impractical. Here, we design a fast algorithm for tackling such large-scale training set. The main idea of our algorithm is that we deploy an *asynchronously paralleled* stochastic gradient descent that can significantly reduce the time of scanning the triplets.

When many computing threads scan the triplets asynchronously, updating the shared parameters of the triplets at the same time may give rise to conflicts. Fortunately, in practice, it is safe to apply such parallelization with no performance loss as shown in literature [22,24]. The reasons are two folds. First, as shown in Fig. 2, the frequency distributions of the

[1] By inductively apply the fact $\sigma(x) + \sigma(-x) = 1$.



(a) (b)

Fig. 2 The distribution of: **a** user pairs who retweet and **b** the retweeted tweets follows a power-law. This observation supports our design of efficient asynchronous parallel algorithms

Algorithm 1: CSIF Learning

Input: Training set features $u_{ij} \in R^{2b}$, $t_k \in R^c$, feature dimension d.
Output: Feature projection matrices $U \in R^{d\times 2b}$, and $T \in R^{d\times c}$.
Initialization: Randomly set parameters $\mathbf{a}_{n_{ij}(m)} \in \mathcal{A}^{(0)}$, $\mathbf{U}^{(0)}$ and $\mathbf{T}^{(0)}$ to $[0, 1]$, $t \leftarrow 0$
repeat
  Phase 1: $\mathcal{A}^{(t+1)} \leftarrow \text{UpdateAux}\left(\mathcal{A}^{(t)}, \mathbf{U}^{(t)}, \mathbf{T}^{(t)}\right)$
  Phase 2: $\mathbf{U}^{(t+1)}, \mathbf{T}^{(t+1)} \leftarrow \text{UpdateMat}\left(\mathcal{A}^{(t+1)}\right)$
  $t \leftarrow t + 1$
until *converges*;

user pairs and the tweets follow a power law. This means that we have a very long tail of infrequent triplets, and thus the chance of two threads scan the same triplet is rare. Second, thanks to the tree-structured parameters, the number of shared parameters between two triplets is limited. For example, even if two triplets corresponds to sibling leaves, the number of shared parameters is only $\log_2 N - 1$. That is to say, when $N = 10^7$, the chance of conflict is only around 0.00002 %, which is negligible.

Algorithm 2: UpdateAux $\left(\mathcal{A}^{(0)}, \mathbf{U}, \mathbf{T}\right)$

1 **Initialization**: $\mathbf{a}_{u_i\to u_j} \leftarrow \mathbf{U}\mathbf{u}_{ij}$, $\mathbf{a}_k \leftarrow \mathbf{T}\mathbf{t}_k$, $\mathcal{T}^{(0)} \leftarrow \{\mathcal{A}^{(0)}, \mathbf{a}_{u_i\to u_j}, \mathbf{a}_k\}$, $t \leftarrow 0$, momentum $\Delta^{(0)} \leftarrow 0$, weight-decay factor β, learning rate η
2 **repeat**
3 Online gradient descent:
4 **foreach** $(u_i \to u_j, t_k)$ **do**
5 **foreach** $\mathbf{a} \in \mathcal{T}$ **do**
6 $\Delta^{(t+1)} = 0.9\Delta^{(t)} - \beta \cdot \eta \cdot \mathbf{a}^{(t)} - \eta\nabla_{\mathbf{a}} J(\mathcal{T}^{(t)})$,
7 $\mathbf{a}^{(t+1)} = \Delta^{(t+1)} + \mathbf{a}^{(t)}$,
8 **end**
9 **end**
10 $t \leftarrow t + 1$
11 **until** *converges*;
12 **return** $\mathcal{A}^{(t)}$

However, the feature projection matrices $\mathbf{U}$ and $\mathbf{T}$ are parameters shared by all the triplets. To elimination this obstacle

Algorithm 3: UpdateMat $(\mathcal{A})$

1 **Initialization**: $t \leftarrow 0$, momentum $\Delta^{(0)} \leftarrow 0$, weight-decay factor β, learning rate η
2 **repeat**
3 Stochastic Gradient descent:
4 **foreach** *randomly selected mini-batch from* $\mathcal{S}$ **do**
5 $\Delta^{(t+1)} = 0.9\Delta^{(t)} - \beta \cdot \eta \cdot \left(\mathbf{U}^{(t)}, \mathbf{T}^{(t)}\right) - \eta\nabla_{(\mathbf{U},\mathbf{T})} J\left(\mathbf{U}^{(t)}, \mathbf{T}^{(t)}\right)$,
6 $\left(\mathbf{U}^{(t+1)}, \mathbf{T}^{(t+1)}\right) = \Delta^{(t+1)} + \left(\mathbf{U}^{(t)}, \mathbf{T}^{(t)}\right)$,
7 Row-wise ℓ_2-normalize for $\mathbf{U}^{(t+1)}$ and $\mathbf{T}^{(t+1)}$,
8 **end**
9 $t \leftarrow t + 1$
10 **until** *converges*;
11 **return** $\left(\mathbf{U}^{(t)}, \mathbf{T}^{(t)}\right)$

for parallization, we propose a two-phase learning algorithm as shown in Algorithm 1. Specifically, we first assume $\mathbf{T}\mathbf{t}_k$ in Eq. (5) and $\mathbf{U}\mathbf{u}_{ij}$ in Eq. (6) as auxiliary tree node parameters, and they can be solved by Algorithm 2, where Steps 2–11 can be run asynchronously with multiple threads. In general, Algorithm 2 requires 100–200 iterations for convergence. Then, as shown in Algorithm 3, we solve for $\mathbf{U}$ and $\mathbf{T}$ given the auxiliary parameters optimized by Algorithm 2. Although $\mathbf{U}$ and $\mathbf{T}$ are shared by all the triplets and cannot be run asynchronously, it is easy to see that Algorithm 3 solves a convex problem which can be efficiently optimized in only 1 iteration. We use the momentum-based gradient descent as Steps 6–7 in Algorithm 2 and Steps 5–6 in Algorithm 3. This method has been shown to result in faster learning paces [25]. In the experiments, we set the starting learning rate η to $1e^{-5}$ with dynamic momentum, and used l_2-norm weight decay with $5e^{-5}$ coefficient.

4.3 Inference of influence

So far, we have introduced how to optimize our influence model in Eq. (3) and solve for the influential feature projection matrices $\mathbf{U}$ and $\mathbf{T}$. Therefore, (1) given any triplet $(u_i \to u_j, t_k)$ out of the training set, we can *predict* the probability that user u_j will re-tweet tweet t_k from user u_i; and (2) given a social network $\mathcal{G}$ and a tweet t_k from an unknown topic, we can also *predict* who are the most influential users on the entire social network. We call the first and the second predictions as user–user and global-level inferences.

User–user influence A straightforward way to calculate $p(u_i \to u_j, t_k)$ is according to Eq. (7). Despite the fact that, it requires accumulation along the tree, it cannot handle triplet out of the training set since it does not encode the triplet during training. Instead, we directly measure the similarity between the learned features $\mathbf{U}\mathbf{u}_{ij}$ and $\mathbf{T}\mathbf{t}_k$ as

$$p(u_i \to u_j, t_k) = \sigma\left(\mathbf{u}_{ij}^T \mathbf{U}^T \mathbf{T}\mathbf{t}_k\right). \tag{8}$$

We can also train a binary classifier (e.g., SVM) to predict the probability. To achieve this, we should collect positive training triplets that are facts and negative training triplets that do not exists. Therefore, by considering $[\mathbf{u}_{ij}^T\mathbf{U}^T, \mathbf{t}_k^T\mathbf{T}^T]^T$ as training features, we can learn a binary linear model $\mathbf{w}$ such that

$$p(u_i \rightarrow u_j, t_k) = \sigma\left([\mathbf{u}_{ij}^T\mathbf{U}^T, \mathbf{t}_k^T\mathbf{T}^T]\mathbf{w}\right). \quad (9)$$

Global-level inference Given a tweet t_k (e.g., a commercial of a product), we would like to see how the social influence will propagate given any influence probability $p(u_i \rightarrow u_j|t_k)$. During the propagation, we would like to see how far and wide t_k influences others by a user. Intuitively, it is analogous to measure how popular a Web page can be accessed by other pages through hyper links. Therefore, it motivates us to use PageRank to infer the global-level influence. The measure of global-level influence of all the users $\mathbf{s}$ on a social network with respect to a tweet can be calculated iteratively by the following rule:

$$\mathbf{s} \leftarrow \alpha\mathbf{R} \cdot \mathbf{s} + (1-\alpha)\mathbf{t}, \quad (10)$$

where $\mathbf{R}$ is the transition probability matrix whose entry is defined as in Eq. (8), $\mathbf{t}$ is the re-exportation vector which can be defined as $\frac{1}{|\mathcal{V}|} \cdot \mathbf{1}$, and α is a parameter between 0 and 1 to control the probability of teleportation. The lower α is, the higher the probability that the random influence will teleport to users according to $\mathbf{t}$, and vice versa. In the experiments, we set α to 0.85 according to the setting in [33].

5 Experiments

In this section, we systematically evaluate the effectiveness of our proposed approach by conducting extensive experiments on a large-scale social network. In particular, we test the performances of pair-wise influence prediction and most influential users ranking, which belong to the user–user influence and global-level influence, respectively.

5.1 Data

We used the retweet behaviors as a measure of social influence as in many related studies [1,6,18,29]. The reasons are two folds. First, Twitter-like on-line micro-blog services are the most popular social networks. This allows us to easily collect sufficient data for experiments. Second, the ground truth of social influence, i.e., whether a user is influenced by another user, in the form of retweeting is much more objective to be collected than other forms such as product promotion. For example, it is not easy to judge whether a user buys a product as her friends do.

Our real-world data set was collected from Tencent Weibo, one of the China's leading Twitter-like microblog services. It should be noted that the data set is a snapshot of the entire Tencent Weibo network including 53 million user profiles, 12 billion following relationships, and more than 838 million tweets posted from 15th November to 4th December 2011.

5.1.1 Data pre-process

Since there are many noisy users such as *zombies* or *bots*[2] in the huge dataset, we need to filter out such noise. In particular, we first discarded users who have never posted a tweet during the time period of experiment and then discarded users who have no followees or followers. As a result, this gives rise to 25,285,161 users, 85,270,579 social connections and 211,436,999 tweets.

For constructing the triplets as in Eq. (2), we consider u_i as the sender user (who posts or retweet a tweet) and u_j as the receiver user (who retweet the tweet), and t_k as the tweet. For example, in Fig. 1b, each row of the table represents a retweet triplet, and the right arrow indicates the tweet direction. The total number of triplets is 16,569,067, containing 2,912,987 users and 14,867,245 tweets.

For user features $\mathbf{u}_{ij}$, we used the approach in [24] to extract user features, which are latent social representations of nodes in social network; they encode social relations in a continuous vector space with a low dimension. By setting the size of feature dimension to 100, we obtained a 100-d feature for each user, and thus the dimension of $\mathbf{u}_{ij}$ is 200-d. In fact, we could extract traditional user features such as user profiles. However, since most of the user profiles are incomplete, we did not use the traditional methods. It is noted that our method is generic to any kind of user features. For tweet features, the extraction pipeline was conducted as follows. First, we employed word segmentation to split Chinese characters into words, and transformed each word into a 300-d vector using the language model [22]. Second, we learned a 128-d dictionary from all the words using k-means. Third, for each word in a tweet, we obtained a 128-d sparse codes using any off-the-shelf sparse coding toolbox. Finally, each tweet can be transformed into a 128-d textual feature by square-root pooling of all the words in it.

5.1.2 Tweet topic distribution

Obviously, topics play an important role in social influence [30,33]. Therefore, it is interesting to cluster the tweets into topics and investigate how social influences are undergoing in different topic domains. As in [33], we applied an

[2] Zombies refer to soul-less accounts that post no original content, run by the shady individuals who take customers' money in exchange for these new "fans". Bots refer to machine generated user accounts.

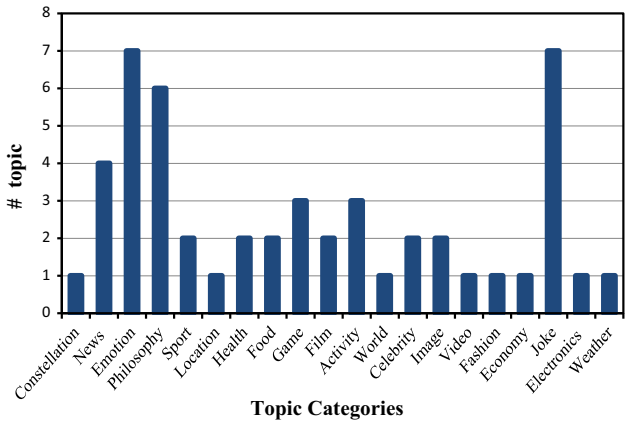


Fig. 3 The distribution of topics over 20 human named topic categories

automatic topic clustering method such as K-means to distill 50 topics for the tweets. Each topic cluster has its own semantic meanings but has no nameable topic names. To show the distribution of the topics, we manually summarized an English name for each cluster. Figure 3 illustrates the distribution of automatic generated topics over 20 human named topic categories. As can be seen, although the 50 topics are automatically generated, they generally cover the diversity and richness of real-world topics on a typical social network.

5.2 Compared methods and metrics

5.2.1 Propagation user prediction

To demonstrate the performance of our learned content–social influential features in estimating user–user influence (cf. Eq. (4)), we conducted a fundamental task in social influence analysis—*Propagation User Prediction* [4,21,35], to predict the user–user influence. The goal of this task is that given a user u_i who posts a tweet t_k, we want to predict how likely an user u_j will retweet it. It can be considered as a classification problem. We used our learned CSIF to compute $p(u_i \rightarrow u_j|t_k)$.

To comprehensively evaluate the effectiveness of our method in both time and topic dimensional prediction, we conducted experiments based on two schemes: time-related prediction and topic-related prediction:

Time-related prediction In this scheme, we divided the data set described above into training and testing sets by time. The retweeting triplets during the first 15 days were treated as training set; while the rest during the last 5 days was treated as testing set. As a result, there are 3,163,399 and 828,554 triplets in the training and testing set, respectively. We considered the triplets exist in the dataset as positive, and, we randomly synthesized triplets as negative. We assured the negative triplets never appeared in the positive set. As a result, we respectively generated 3,163,399 and 828,554 negative triplets for training and testing of this task.

Topic-related prediction To demonstrate that our method can also be conducted on different topics, we split the training set and testing set according to topics. Specially, we randomly selected 25 topics for training and kept the rest for testing.[3] It should be highlighted that the topics in testing are unseen in training. As a result, we obtained 8,178,004 positive training triplets and 5,183,086 positive testing triplets. As described above, in this task we created 8,178,004 and 5,183,086 negative triplets for training and testing.

To evaluate the effectiveness of our proposed approach, we compared the following methods. The first three methods are baselines and the last two methods are two versions of our proposed approach.

- Static model [13]: the influential probability is defined as:

$$p(u_i \rightarrow u_j|t_k) = N_{u_i \rightarrow u_j}/N_{u_i}, \tag{11}$$

  where N_{u_i} is the number of *retweets* posted by u_i in the training set, and $N_{u_i \rightarrow u_j}$ is the number of tweets that u_j retweets from u_i in the training set. In this model, the influence of u_i on its inactive neighbor u_j at any time is fixed, and it assumes the action that u_i tries to influences u_j as Bernoulli trial. Note that this model neglects the factors of topic.
- Continuous model [13]: the influential probability is defined as:

$$p(u_i \rightarrow u_j|t_k, t) = p(u_i \rightarrow u_j|t_k, 0)e^{-(t-t_{u_i})/\tau_{u_i,u_j}}, \tag{12}$$

  where t is the time slot, $p(u_i, u_j, 0)$ is the maximum strength of u_i influencing u_j, which is estimated in exactly the same way as in the static model. t_{u_i} denotes the time when user u_j posted the tweet, and τ_{u_i,u_j} is called the *mean life time*, which corresponds to the expected time delay between u_i posting a tweet and u_j retweeting the same tweet. In this model, the influential probability that a user u_i successfully influences its inactive neighbor u_j in a certain time interval (e.g., ten minutes, one hour, one week) decays in an exponential rate. We adopted the interval of one day in the experiments. This model also neglects the topic factors. Note that we only employed this method in evaluating time-related prediction as it is not reasonable to perform topic-related prediction.
- SVM [34]: we concatenated the user features and the tweet features as a feature vector for retweet triplet. As

[3] We did not find any significant performance variance by using 10 different random split. Therefore, we just arbitrarily chose one split.

this task is a classification problem, we adopted a binary linear SVM classifier to predict the influential probability.
- PIF+Dot: we used our proposed two CSIF ($\mathbf{Uu}_{ij}$ and $\mathbf{Tt}_k$) to and the dot product of them as in Eq. (8) to predict the probability.
- PIF+SVM: we used our proposed two CSIF and a linear SVM as in Eq. (9) to predict the probability.

Note that there are other methods [4,21,30,35] related to influence prediction, we did not compare with these works [4,21,35] because they attempt to predict which user will retweet a tweet without considering the sender of the tweet, which is not a reasonable scenario for micro-level prediction. Also, we have tried the code released by [30], but it seems that the implementation is not scalable to the large-scale network used in our experiment.

We adopted the widely used metrics ROC (receiver operating characteristic) curve and AUC (area under ROC curve). ROC plots the true positive rate against the false positive rate at various threshold settings, and the hump of ROC curve close to the point (0,1) indicates better prediction. AUC indicates how well the prediction model outperforms the prediction by chance (i.e., AUC = 0.5), and larger AUC indicates better prediction models.

5.2.2 Domain expert identification

To illustrate the performance of our learned content–social influential features in estimating global-level influence, we employed another important task in social influence analysis—*Domain Expert Identification* [8,30]. The goal of this task is that given a topic t_k, find the top K influential users which we named as "Domain Expert" with respect to this topic. In this task, we merged the user–user influential probabilities, that are calculated by our learned CSIF, into a Pagerank algorithm to obtain the ranking of users under a topic. The dataset split in this task is the same as that in topic-related prediction. Hence, we had 8,178,004 training and 5,183,086 testing triplets. It should be highlighted that as compared to the work of [30] in which the testing data is a subset of training data, the testing triplets in our work are unseen in training, so as to simulate the real environment approximately.

To evaluate the effectiveness of our proposed approach, we compared the following methods. The first five methods are baselines and the last two methods are two versions of our proposed approach.

- # Followers [6]: consider the number of followers of a user as his/her influence score, which is used to rank influential users.
- ABUI [36]: an Action-Based User Influence model, which incorporates the static model described above with Pagerank algorithm (cf. Eq. (10)). We used this model to compute the user–user influence probabilities of all the testing triplets, then averaged all the probabilities of the same directed user pair in turn to obtain value in corresponding position of the transition probability matrix in Eq. (10). The teleportation vector is defined as $\frac{\mathbf{1}}{n}$, where n is the number of users.
- SVM [34]: we used SVM method described in [34] to predict user–user influence probabilities for all the testing triplets, and then constructed a transition probability matrix related to a topic by averaging all the probabilities of each directed user pair under this topic as entries of the matrix. The teleportation vector is again defined as $\frac{\mathbf{1}}{n}$.
- TSPR [15]: the topic-sensitive Pagerank algorithm, which has been widely used to measure the topic-specific influence by calculating Pagerank vector for each topic. Specially, it considers that the transition probability from one user to another is the same for different topics, while the teleportation vector of the random surfer in topic t_k is topic-biased.
- TwitterRank [33]: a variation of PageRank over the follower network induced by a particular topic. The main different between this method and Pagerank is that its transition probability from one user u_i to another u_j is topic-special, which can be formulated as:

$$\mathbf{R}_t(i,j) = \frac{|\mathcal{T}_j|}{\sum a : u_i \rightarrow u_a |\mathcal{T}_a|} * \mathrm{sim}_t(i,j), \tag{13}$$

  where $u_i \rightarrow u_a$ denotes u_i is a follower of u_a, $|\mathcal{T}_j|$ is number of tweets posted by u_j, and $\sum |\mathcal{T}_a|$ sums up the number of tweets posted by all of u_i's followees. $\mathrm{sim}_t(i,j)$ is the similarity between u_i and u_j in topic t. In addition, its teleportation vector of the random surfer in topic t_k is also topic-special.
- PIF+Dot: we used the proposed CSIF and the dot product of them to compute user–user influential probabilities, and then constructed transition probability matrix for each testing topic by averaging all the probabilities of each directed user pair under this topic as entries of the matrix.
- PIF+SVM: we trained a linear SVM using our proposed CSIF to predict the user–user influential probabilities, and then constructed topic-special transition probability matrix in the same way as that in PIF+Dot.

We adopted average NDCG at top K ranking users (average NDCG@K) for all the testing topics as the performance evaluation measure. Since the dataset has no known ground truth, we instead generated ground truth based on statistics of retweet as in [6]. That is, we computed the influence score of a user by summing up the retweet times of all the tweets he

posted. We then ordered the scores in ascending sequence and equally partitioned them into three level, labeled by values 0, 1, 2, representing non-influential, influential, very influential, respectively [21].

5.3 Experimental results

5.3.1 Evaluations of propagation user prediction

Figure 4a shows the comparison results for our proposed methods and other methods in predicting propagation user in the time dimension. We can see that our proposed two methods outperforms the other methods. Since it is time-related prediction, the time continuous model performs better than static model. SVM with basic user and tweet features obtains a poor performance, while integrating SVM with our learned features CSIF achieves the best performance, which indicates that our proposed CSIF do help boost the prediction accuracy. Particularly, both of our two methods improve the average AUC value by more than 45, 26, 16 % as compared to SVM, static model and continuous model, respectively. Interestingly, only using the dot product of our proposed CSIF, the performance is better than SVM. These results demonstrate the effectiveness of our proposed CSIF in measuring the user–user influence with respect to time, leading to more accurate prediction of pairwise influence propagation.

Figure 4b shows the comparison results in predicting propagation user in the topic dimension. We can observe that our proposed two methods outperform others. Since the static model is not sensitive to topic, it results in a poor performance which is similar to that of random guess. Integrating our proposed CSIF into SVM achieves a great improvement. In particular, both of our two methods improve the average AUC measure by more than 52 and 24 % as compared to SVM and static model, respectively.

The above two comparison results clearly demonstrate that our proposed CSIF offer good prediction performance on both time and topic dimensions.

5.3.2 Evaluations of domain expert identification

Figure 5 presents the comparison results for our proposed methods and other methods in identifying domain experts. We can observe that our two related methods outperform others. The number of followers which indicates a user's popularity performs the worst. This finding suggests that high user popularity does not lead to high influence, which is consistent to the observation in [6]. ABUI and TSPR methods achieve poor performances because they do not consider the topic in constructing the transition probability matrix. As TwitterRank takes topic into consideration, it has an improvement as compared to the former two. Finally, our two proposed methods based on the proposed CSIF outperform TwitterRank; we attribute the superior results to the fact that topic-special transition probability used in TwitterRank is based on simple statistics of the number of retweeted tweets related to the topic, which does not capture the higher order

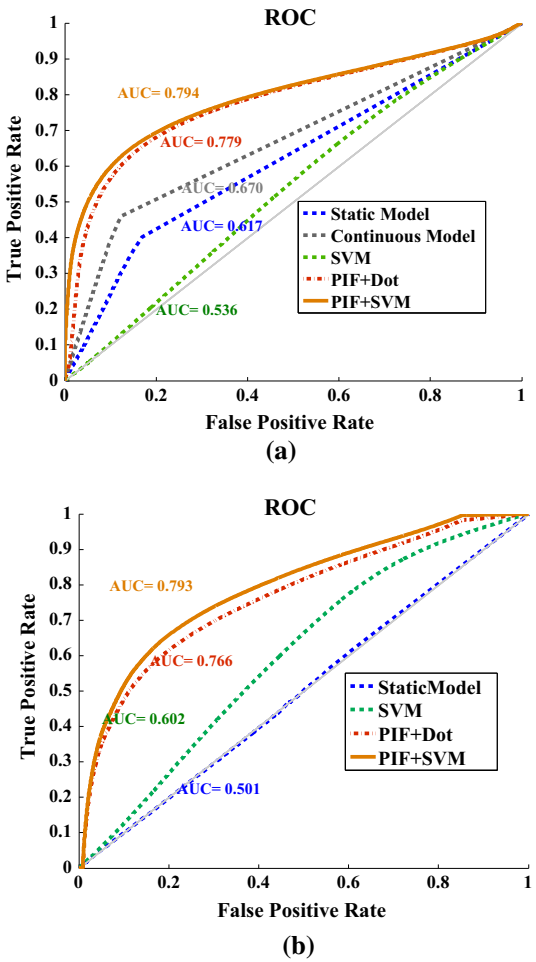


Fig. 4 Performance (ROC) of various methods in propagation user prediction. **a** Time dimension. **b** Topic dimension

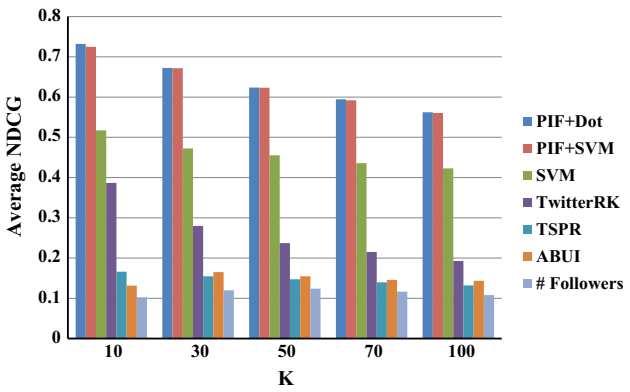


Fig. 5 Performance (average NDCG@K) of various methods in domain expert identification

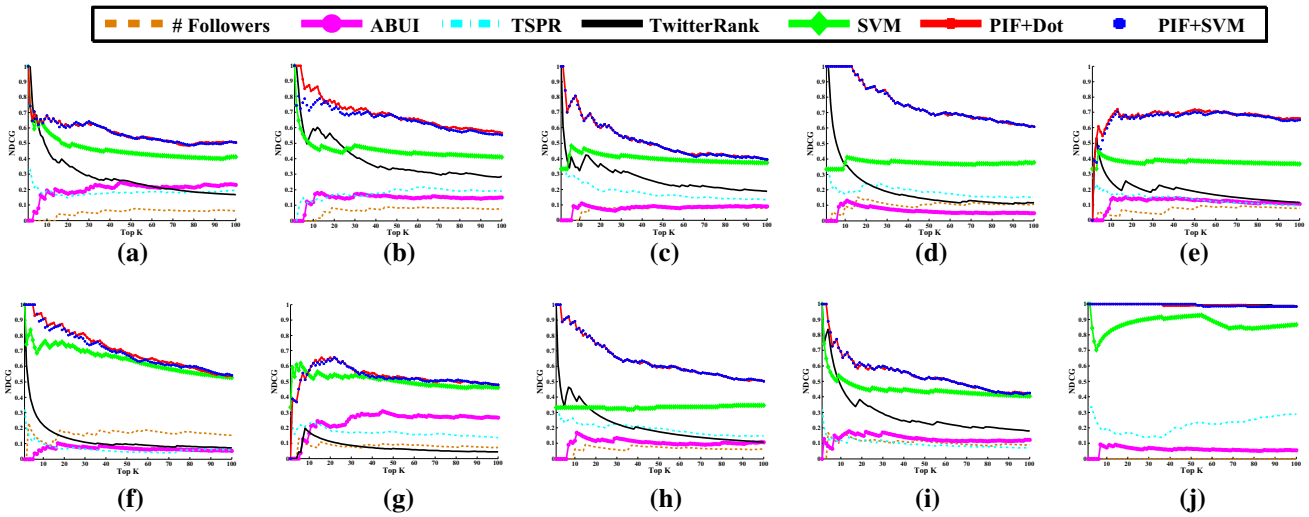


Fig. 6 Performance (NDCG) of various methods in domain expert identification under different topics. **a** #3: Celebrity, **b** #4: Video, **c** #7: Fashion, **d** #9: Economy, **e** #13: Philosophy, **f** #16: Sport, **g** #17: Food, **h** #18: Joke, **i** #24: Electronics, **j** #25: Weather

relations between retweets and topics. On the other side, our method correctly models such higher order relations.

Figure 6 shows the details of 10 selected topics from the 25 testing topics. Our proposed methods outperform other methods in every topic. However, it is noted that the performance of our proposed methods do not lead to obvious improvement in the topic "food"; the reason is that when users talk about food in the tweets, they usually post a related image of food rather than related text about food, which results in lots of irrelevant text in this topic. In the future work, we will extend to deal with image content, such as concept recognition. On the other hand, our proposed methods and TwitterRank, which explicitly exploit topic information, almost perform perfectly in the topic "weather"; this is because all the tweets talking about weather is highly relevant in the text dimension.

5.3.3 Evaluations of scalability

Figure 7 illustrates the training time of parallelizing our method under different feature sizes in off-line training stage, where there are about 8 million training triplets. Our experiments are implemented on a server with one 24-core CPU. It is obvious that the speed up in training is consistent with the increase in the number of threads, and the runtime is constantly lowered when we reduce the feature dimension. Note that the runtime scales linearly with the number of the training triplets (not reported in the figure); and one can half the training time by adding one more CPU.

5.3.4 Evaluations of feature dimension

Finally, we evaluated the sensitivity of various feature dimensions. Figure 8 illustrates the effects of different feature dimension d on different tasks. We can see that the per-

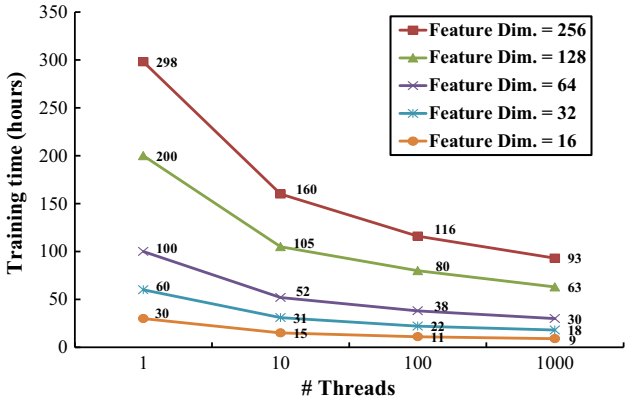


Fig. 7 Training time of parallelizing our method with different feature sizes in off-line training stage (8,000,000 training triplets), on one 24-core CPU

Feature Dimension	2^4	2^5	2^6	2^7	2^8
AUC	0.7780	0.7782	0.7783	0.7781	0.7787

(a)

Feature Dimension	2^4	2^5	2^6	2^7	2^8
AUC	0.7659	0.7661	0.7656	0.7660	0.7662

(b)

Feature Dimension	2^4	2^5	2^6	2^7	2^8
Average NDCG@10	0.7295	0.7289	0.7294	0.7294	0.7320

(c)

Fig. 8 Performance comparison of feature dimension d on different applications. **a** d on time-related propagation user prediction. **b** d on topic-related propagation user prediction. **c** d on domain expert identification

formances of different tasks are not sensitive to feature dimension. In practice, we can choose a smaller dimension for efficient training.

6 Conclusion

In this paper, we explored at a novel approach that learns the unified feature representations for user pair and content, where the feature similarity between them can be used to predict how likely a user will be influenced by another to retweet a tweet. Our approach fundamentally models how users are influencing each others by social relations and topics, and hence it has broad applications in analyzing the information dynamics in social networks. We also developed an efficient and scalable algorithm for learning the features on large-scale social networks. Extensive experiments conducted on a real-world social network comprising 53 million users and 838 million tweets. The promising results on predicting user–user influence and global-level most influential users demonstrated the effectiveness of our approach.

The key difference between previous approaches and our work is that while the topic of influence studied in theirs are only based on implicit and heuristic statistics, such as the number followers or simple statistical models based on heuristics, our influence is explicitly inferred from the propagation users and content in the entire network. This influence probability is based on the nature of how influence is taking place in social networks. Therefore, our proposed content–social influential feature is able to effectively predict the social influence of an unseen user relation or a tweet on any topic.

Our future work will focus on developing a prescriptive method for social influence. That is, we wish to investigate which subsets of influential features will maximize the social influence. The result will be able to help design a flexible campaign strategy for marketers, politicians or even casual users to enhance their influences in social networks.

Acknowledgments This work was partially supported by the NUS-Tsinghua Extreme Search (NExT) project (Grant R-252-300-001-490). NExT research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its IRC@SG Funding Initiative.

References

1. Bakshy E, Hofman JM, Mason WA, Watts DJ (2011) Everyone's an influencer: quantifying influence on twitter. In: Proceedings of the 4th ACM international conference on Web search and data mining. ACM, pp 65–74
2. Barbieri N, Bonchi F, Manco G (2012) Topic-aware social influence propagation models. In: Proceedings of the 12th IEEE international conference on data mining. IEEE, pp 81–90
3. Bengio Y (2009) Learning deep architectures for AI. Found Trends Mach Learn 2(1):1–127
4. Bian J, Yang Y, Chua TS (2014) Predicting trending messages and diffusion participants in microblogging network. In: Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval. ACM, pp 537–546
5. Bordes A, Weston J, Collobert R, Bengio Y et al (2011) Learning structured embeddings of knowledge bases. In: Conference on artificial intelligence
6. Cha M, Haddadi H, Benevenuto F, Gummadi PK (2010) Measuring user influence in twitter: the million follower fallacy. ICWSM 10:10–17
7. Domingos P, Richardson M (2001) Mining the network value of customers. In: Proceedings of the 7th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, pp 57–66
8. Dong W, Pentland A (2007) Modeling influence between experts. In: Artifical intelligence for human computing. Springer, Berlin, pp 170–189
9. Du N, Song L, Yuan M, Smola AJ (2012) Learning networks of heterogeneous influence. In: Advances in neural information processing systems
10. Freeman LC (2004) The development of social network analysis: a study in the sociology of science, vol 1. Empirical Press, Vancouver
11. Goldenberg J, Libai B, Muller E (2001) Talk of the network: a complex systems look at the underlying process of word-of-mouth. Mark Lett 12(3):211–223
12. Gomez Rodriguez M, Leskovec J, Krause A (2010) Inferring networks of diffusion and influence. In: Proceedings of the 16th ACM SIGKDD international conference on knowledge discovery and data mining
13. Goyal A, Bonchi F, Lakshmanan LV (2010) Learning influence probabilities in social networks. In: Proceedings of the 3rd ACM international conference on Web search and data mining. ACM, pp 241–250
14. Goyal A, Lu W, Lakshmanan LV (2011) Celf++: optimizing the greedy algorithm for influence maximization in social networks. In: Proceedings of the 20th international conference companion on World wide web. ACM, pp 47–48
15. Haveliwala TH (2002) Topic-sensitive pagerank. In: Proceedings of the 11th international conference on World Wide Web. ACM, pp 517–526
16. Heidemann J, Klier M, Probst F. Identifying key users in online social networks: a pagerank based approach. In: Proceedings of 31st international conference on information systems (ICIS)
17. Kempe D, Kleinberg J, Tardos É (2003) Maximizing the spread of influence through a social network. In: Proceedings of the 9th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 137–146
18. Kwak H, Lee C, Park H, Moon S (2010) What is twitter, a social network or a news media? In: Proceedings of the 19th international conference on World wide web. ACM, pp 591–600
19. Li N, Gillet D (2013) Identifying influential scholars in academic social media platforms. In: Proceedings of the 2013 IEEE/ACM international conference on advances in social networks analysis and mining. ACM, pp 608–614
20. Liu B, Cong G, Zeng Y, Xu D, Chee YM (2014) Influence spreading path and its application to the time constrained social influence maximization problem and beyond. IEEE Trans Knowl Data Eng 26(8):1904–1917
21. Liu L, Tang J, Han J, Jiang M, Yang S (2010) Mining topic-level influence in heterogeneous networks. In: Proceedings of the 19th ACM international conference on Information and knowledge management. ACM, pp 199–208
22. Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J (2013) Distributed representations of words and phrases and their compo-

sitionality. In: Advances in neural information processing systems, pp 3111–3119
23. Morin F, Bengio Y (2005) Hierarchical probabilistic neural network language model. In: AISTATS, vol 5. Citeseer, pp 246–252
24. Perozzi B, Al-Rfou R, Skiena S (2014) Deepwalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining
25. Qian N (1999) On the momentum term in gradient descent learning algorithms. Neural Netw 12(1):145–151
26. Rashotte L (2007) Social influence. Blackwell Encycl Soc Psychol 9:562–563
27. Richardson M, Domingos P (2002) Mining knowledge-sharing sites for viral marketing. In: Proceedings of the 8th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 61–70
28. Rodriguez MG, Balduzzi D, Schölkopf B (2011) Uncovering the temporal dynamics of diffusion networks. arXiv preprint arXiv:1105.0697
29. Rudat A, Buder J (2015) Making retweeting social: the influence of content and context information on sharing news in twitter. Comput Hum Behav 46:75–84
30. Tang J, Sun J, Wang C, Yang Z (2009) Social influence analysis in large-scale networks. In: Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 807–816
31. Tang L, Liu H (2009) Relational learning via latent social dimensions. In: Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 817–826
32. Wang S, Hu X, Yu PS, Li Z (2014) Mmrate: inferring multi-aspect diffusion networks with multi-pattern cascades. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining
33. Weng J, Lim EP, Jiang J, He Q (2010) Twitterrank: finding topic-sensitive influential twitterers. In: Proceedings of the 3rd ACM international conference on Web search and data mining. ACM, pp 261–270
34. Xiao C, Zhang Y, Zeng X, Wu Y (2013) Predicting user influence in social media. J Netw 8(11):2649–2655
35. Zhang J, Liu B, Tang J, Chen T, Li J (2013) Social influence locality for modeling retweeting behaviors. In: Proceedings of the 23rd international joint conference on artificial intelligence. AAAI Press, pp 2761–2767
36. Zhang M, Sun C, Liu W. Identifying influential users of microblogging services: a dynamic action-based network approach. In: Proceedings of PACIS
37. Zhou C, Zhang P, Zang W, Guo L (2015) On the upper bounds of spread for greedy algorithms in social network influence maximization. IEEE Trans Knowl Data Eng 1904–1917