Social Influence Does Matter: User Action Prediction for In-Feed Advertising

Hongyang Wang,[1] Qingfei Meng,[1,2] Ju Fan,[*1] Yuchen Li,[3]
Laizhong Cui,[4] Xiaoman Zhao,[1] Chong Peng,[2] Gong Chen,[2] Xiaoyong Du[1]
[1]Renmin University of China, [2]Tencent, [3]Singapore Management University, [4]Shenzhen University
{harpuia, fanj, xiaomanzhao, duyong}@ruc.edu.cn, {zoemeng, michaelpeng, natchen}@tencent.com,
yuchenli@smu.edu.sg, cuilz@szu.edu.cn

Abstract

Social in-feed advertising delivers ads that seamlessly fit inside a user's feed, and allows users to engage in social actions (likes or comments) with the ads. Many businesses pay higher attention to "engagement marketing" that maximizes social actions, as social actions can effectively promote brand awareness. This paper studies social action prediction for in-feed advertising. Most existing works overlook the *social influence* as a user's action may be affected by her friends' actions. This paper introduces an end-to-end approach that leverages social influence for action prediction, and focuses on addressing the high sparsity challenge for in-feed ads. We propose to learn influence structure that models who tends to be influenced. We extract a subgraph with the near neighbors a user interacts with, and learn topological features of the subgraph by developing structure-aware graph encoding methods. We also introduce graph attention networks to learn influence dynamics that models how a user is influenced by neighbors' actions. We conduct extensive experiments on real datasets from the commercial advertising platform of WeChat and a public dataset. The experimental results demonstrate that social influence learned by our approach can significantly boost performance of social action prediction.

Introduction

Social media *in-feed advertising* delivers ads in the form of a native feeds stream for smooth user experience. This emerging format of advertising has become one of the most effective advertising methods (Fan, Lu, and Gupta 2017), and been widely adopted by social platform market leaders like Facebook, Twitter and WeChat.

One key characteristic of in-feed advertising is that a user can engage in *social actions* with an ad as with other common contents, such as *likes* and *comments*. And such actions are visible to the user's friends, who are also exposed with the same ad. Then, the users can interact with their friends by liking the same ad or replying to friends' comments. Evidently, the more people engage with the ads, the more likely they have interest in the products/business in the ads. Thus, many businesses pay higher attention to the "engagement marketing" that maximizes social actions; gaint social media platforms, such as Facebook, also take engagement as a crucial advertising metric[1].

As a crucial step to promote user engagements, this paper studies *social action prediction*: given an ad exposure to a user, it predicts whether the user will perform social action on the ad. This problem is inherently different from click-through rate (CTR) prediction, which is studied in conventional advertising (Guo et al. 2017; Chen et al. 2016). Unlike the click behavior, social actions of a user in the scenario of in-feed advertising would be very likely influenced by her friends. For example, after noticing that some friends have liked an ad, the user may be encouraged to join her friends and interact with the ad. According to some marketing studies (Edelman 2013), consumers trust their peers' opinions of products over companies'. Unfortunately, the existing prediction techniques overlook such *social influence*.

However, it is very challenging to utilize social influence for in-feed advertising, as social interactions among users on the same ads are highly sparse. Firstly, in the social interaction network (SIN) built on same-ad interaction data, users have very few neighbors: the average degree of the SIN is much less than that of general social networks. Secondly, similar users may not be connected or may be far away from each other in the SIN. The prominent sparsity challenge makes the existing methods inadequate for in-feed ads. For example, DeepInf (Qiu et al. 2018) is the state-of-the-art social influence prediction approach designed for general social network settings. It predicts action status of a user by considering her neighbors' features and action status. However, it is not effective for in-feed advertising as very limited neighbor information is available due to the sparsity.

To address the challenge, we introduce an end-to-end approach to learn predictive signals in social influence for boosting social action prediction. First, we propose to model *who* tends to be influenced. We introduce the *topological* feature learning into in-feed advertising by extracting a subgraph of each user as its neighborhood and encoding topology of the neighborhood. The intuition is that the topology reflects whether the user is in the "center" or "peripheral" of a neighborhood that tends to influence her. The topolog-

[*]Ju Fan is the corresponding author.

[1]https://www.facebook.com/business/news/pageinsights

ical features are robust and effective for social action prediction even though the network is very sparse, as validated by our experiments. Furthermore, we consider *how* a user is influenced by her neighbors. We utilize graph attention networks to combine neighbors' features and their action status to learn *influence dynamics*. Finally, we combine social influence with ad exposure features, and leverage state-of-the-art classification models to predict social actions.

To summarize, we make the following contributions.

(1) To the best of our knowledge, we are the first to systemically study social action prediction on *large-scale* and *real* in-feed advertising datasets, which are collected from WeChat, the largest social platform in China.

(2) We study the social influence modeling problem for action prediction in social in-feed advertising. We propose to learn topological features of social influence structure, and develop structure-aware graph encoding methods.

(3) We conduct extensive experiments on real datasets. The results show that the learned topological features are effective to improve prediction performance, and useful to provide insights by visualization. The results also demonstrate that consideration of social influence can boost $4-5\%$ improvement on AUC for social action prediction.

Problem Formulation

This paper studies *social action prediction* for in-feed advertising. We refer to the delivery of an ad a to a user u as an *ad exposure* $\epsilon = \langle a, u \rangle$, and define the problem as below.

Definition 1 (Social Action Prediction) *Given an ad exposure $\epsilon = \langle a, u \rangle$, social action of u on a is defined as a binary variable y_ϵ such that $y_\epsilon = 1$ if u performs actions ("like" or "comment") on a, or $y_\epsilon = 0$ otherwise. Social action prediction is to develop a model $\hat{y}_\epsilon = f(\epsilon)$ to estimate the probability that user u performs social action on ad a.*

For effective prediction, this paper focuses on learning predictive signals in *social influence*. A naïve method to learn social influence is simply based on the utilization of users' social connections, i.e., the friendships or following relationships. However, social connections don't necessarily lead to similar behaviors among the users. We propose to learn social influence from the *historical interaction* among users. The intuition is that if two users are friends and they "co-like" (i.e., have social actions on) the same ads very recently, we consider that they are more likely to influence each other on the incoming ad exposures. Formally, we introduce a *social influence network* (SIN) to model historical interactions among users.

Definition 2 (Social Interaction Network, SIN) *Social interaction network (SIN) is defined as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is a set of users and $\mathcal{E}$ is an edge set. Edges in $\mathcal{E}$ denote the* influence *relationship between users: an edge e exists from u to v, if 1) u and v have social relationship (they are friends or v follows u) and 2) v has actions on an ad on which u already has actions.*

Based on the SIN, our objective is to learn social influence features of a target user u from it and fed the features to the model $\hat{y}_\epsilon = f(\epsilon)$ to produce accurate prediction.

Our Approach

We develop an end-to-end approach to learn predictive signals in social influence. Figure 1 provides an overview of our approach. It first constructs a social interaction network (SIN) from users' historical interactions with the ads, based on the definition of SIN. In our work, we use users' interactions within a period (e.g., two months) before the dates of training/prediction instances to construct the SIN. Then, our approach aims to model two types of social influence features from the constructed SIN for a target user u.

- We model *who* tends to be influenced by the neighbors. To this end, we propose to learn *topological* features $\boldsymbol{x}_u^{(s)}$ of the target user's neighbors from the SIN, as illustrated in Figure 1 (b). The intuition is that the topology reflects whether the user is in the "center" or "peripheral" of a neighborhood that tends to influence her. For effectively learning topological features, we develop structure-aware graph encoding methods.
- We consider *how* a user is influenced by her neighbors. The intuition is that, even if two users have similar topological structures, the influences to them may be different because of various features and action statuses of the neighbors, as shown in Figure 1 (c). Thus, we utilize graph attention networks to combine neighbors' features and their action status to learn *influence dynamics* $\boldsymbol{x}_u^{(d)}$.

To effectively predict action status $\hat{y}_\epsilon$ for an ad exposure instance $\epsilon = \langle a, u \rangle$, besides the above social influence features, we also consider *intrinsic* features of ϵ, denoted by $\boldsymbol{x}_\epsilon^{(i)}$, so as to capture whether u is interested in a. More details of intrinsic features can be referred to our experiments. Finally, a classification model, such as Logistic Regression (LR) or DeepFM (Guo et al. 2017), is utilized to predict social action (Figure 1 (d)). The model concatenates both social influence and intrinsic features, and outputs the prediction result $\hat{y}_\epsilon = f(\boldsymbol{x}_u^{(s)}, \boldsymbol{x}_u^{(d)}, \boldsymbol{x}_\epsilon^{(i)})$.

Modeling Topological Influence Structure

Influence structure modeling aims to learn topological features of the neighborhood of target user u in SIN. Taking the SIN in Figure 1 (a) as an example, we observe that users u and w have different structural features. User u is in the "center" of her local structure, which means that u tends to be influenced by many neighbors in historical user interactions. In contrast, user w is in the "peripheral", which shows that w is less likely to be influenced previously. We consider such influence structure would affect their action statuses in the future. However, the challenge is how to embed users with similar *structural roles* closely together. The existing methods, such as DeepWalk (Perozzi, Al-Rfou, and Skiena 2014), LINE (Tang et al. 2015) may not be adequate, as they focus on learning homophily of users. Node2vec (Grover and Leskovec 2016) can lead to *structure equivalence* embeddings using breath-first sampling (BFS). However, it may not work well in our sparse SIN where nodes with similar structural roles may not be connected or may be far away from each other, due to the high sparsity.

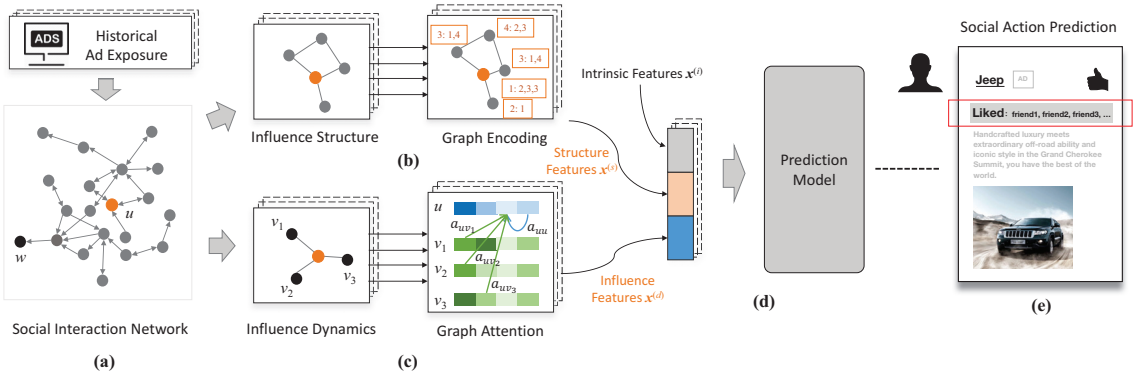


Figure 1: Framework of our proposed approach.

We propose a method to learn topological features of u's local structure. It first extracts a subgraph of u as its *neighborhood* and then encodes this neighborhood based on the Weisfeiler-Lehman (WL) algorithm (Douglas 2011). The goal of the encoding is to assign subgraphs of similar structure with similar numerical vectors as representation.

To learn structural role of target user u, we first introduce *neighborhood* of u, i.e., the subgraph that encloses u.

Definition 3 (Neighborhood) *Given an SIN $\mathcal{G}$ and a target user u, the in-neighborhood (out-neighborhood) of u, denoted by $\Gamma_u^{\mathtt{I}}$ ($\Gamma_u^{\mathtt{O}}$), is defined as a subgraph of $\mathcal{G}$ with: 1) node set $\mathcal{V}_u = \{v | v \in \mathcal{V}, d(v,u) \leq r\}$ ($\mathcal{V}_u = \{v | v \in \mathcal{V}, d(u,v) \leq r\}$) where $d(v,u)$ is the shortest distance from v to u and r is a user-defined threshold, and 2) edge set $\mathcal{E}_u$ containing edges among the nodes in $\mathcal{V}_u$.*

We decide the value of r empirically and use $r = 2$ as it achieves the best performance in our experiments. For simplicity, this section only considers in-neighborhood, denoted as *neighborhood* Γ_u and omits edge directions.

Neighborhood Labeling. A straightforward way to represent topological structure is to use graph isomorphism test: isomorphic neighborhoods should be encoded closely together. However, as graph isomorphism is proved to be NP-hard, we apply the Weisfeiler-Lehman (WL) algorithm (Zhang and Chen 2017; Yanardag and Vishwanathan 2015; Douglas 2011). The idea is to decompose a neighborhood into its subtree patterns, and neighborhoods with similar patterns could be assigned with similar representations. To this end, it iteratively labels a neighborhood using the following *node coloring* process.

(1) Initially, the algorithm colors target user u and other nodes Γ_u based on their shortest path distances to u, i.e., $label^{(0)}[v] = \mathtt{dist}(v,u)$. Then, for each node in Γ_u, it collects a multi-set $mul^{(0)}[v]$ that contains its neighbors' labels,

$$mul^{(0)}[v] = \{label^{(0)}[w] | (w,v) \in \mathcal{E}\}, \tag{1}$$

where the labels in $mul^{(0)}[v]$ are sorted in ascending order.

(2) In the i-th iteration, the algorithm assigns every node v with a new label that reflects its previous label and the multi-set of its neighbors. Specifically, it defines signature of v by concatenating v's label $label^{(i-1)}[v]$ with $mul^{(i-1)}[v]$, i.e.,

$$sig^{(i)}[v] = \left\{label^{(i-1)}[v] \mid mul^{(i-1)}[v]\right\} \tag{2}$$

It sorts the nodes according to their signatures using set comparison: $sig[v_1] < sig[v_2]$[2] iff $label[v_1] < label[v_2]$ or $\exists\ l, mul[v_1][l] < mul[v_2][l], \forall j < l, mul[v_1][j] = mul[v_2][j]$. After sorting, it assigns new label $label^{(i)}[v]$ to each node where assigning the same label to nodes with the same signature, and updates $mul^{(i)}[v]$ accordingly.

(3) The algorithm terminates if the node labels are stable, or a pre-defined number of iterations is reached.

Neighborhood Encoding. It aims to encode u's neighborhood based on its labeling results and produce a numerical vector $\boldsymbol{x}_u^{(s)}$ as its representation. To this end, we introduce two encoding methods described as follows.

(1) *Frequency-based encoding:* It utilizes frequencies of the subtree patterns in the neighborhood. As subtree patterns are reflected by node labels, the method counts the occurrences of labels in each labeling iteration. Suppose that there are h iterations in the above labeling algorithm. In the i-th iteration, it generates a vector $\boldsymbol{x}_i$ where the j-th element of $\boldsymbol{x}_i$ is the occurrence of label j in this iteration, i.e.,

$$\boldsymbol{x}_i[j] = \sum_{v \in \Gamma_u} \boldsymbol{I}(label^{(i)}[v] = j), \tag{3}$$

where $\boldsymbol{I}(label^{(i)}[v] = j)$ is an indicator function that equals 1 if $label^{(i)}[v] = j$ or 0 otherwise. Finally, it concatenates the vectors generated in all iterations as the final neighborhood encoding, i.e., $\boldsymbol{x}_u^{(s)} = [\boldsymbol{x}_1; \boldsymbol{x}_2; \ldots; \boldsymbol{x}_h]$.

There are two limitations for frequency-based encoding. First, node labels would have correlations, which will then affect performance of the classification model. Second, it may not be generalized to capture the topological structures that appear few or are not in the training set.

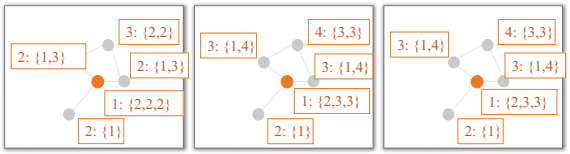
(2) *Embedding-based encoding:* It aims to learn the latent representation of the structures which can overcome the above limitations. The method first learns embeddings for node labels, inspired by (Yanardag and Vishwanathan 2015). Then, it aggregates label embeddings to encode the

[2]For simplicity, we omit the superscript if the context is clear.

neighborhood. Specifically, given one neighborhood, it considers its node labels in each iteration as a *sequence*. Then, by considering all neighborhoods in our SIN, it includes all sequences together to form a "corpus", where each label is like a "word" and each sequence is like a "sentence". It applies word embedding techniques (Mikolov et al. 2013) on the corpus to obtain embedding for each label. Finally, neighborhood encoding $\boldsymbol{x}_u^{(s)}$ can be obtained by averaging the label embeddings in the neighborhood.

Algorithm 1 GRAPHENCODE ($\mathcal{G}, h$)

Require: $\mathcal{G}$: an SIN; h: number of iterations
1: **for** each target user u in $\mathcal{G}$ **do**
2: Extract neighborhood Γ_u from $\mathcal{G}$ for u
3: Initialize $label^{(0)}[v]$, $mul^{(0)}[v]$ for each $v \in \Gamma_u$
4: **for** i iterates from 1 to h **do**
5: Generate $sig^{(i)}[v]$ for each node v in Γ_u
6: Sort nodes in ascending order of $sig^{(i)}[v]$
7: Update $label^{(i)}[v]$ and $mul^{(i)}[v]$
8: **end for**
9: **end for**
10: **for** each target user u in $\mathcal{G}$ **do**
11: Generate $\boldsymbol{x}_u^{(s)}$ based on node labels using frequency or embedding-based methods
12: Add $\{(u, \boldsymbol{x}_u^{(s)})\}$ into $\mathcal{O}$
13: **end for**
14: Return $\mathcal{O}$



(a) Initial coloring. (b) 1st iteration. (c) 2nd iteration.

Figure 2: Illustration of neighborhood labeling.

To summarize, the pseudo-code of our method is summarized in Algorithm 1. Figure 2 shows an example of the algorithm. It initializes node labels according to their distances to target user u (the orange node), and collects multi-sets of their neighbor labels. For example, the label and multi-set of user u are respectively 1 and $\{2, 2, 2\}$. Then, it sorts the nodes based on the signatures in ascending order, i.e., $1 : \{2, 2, 2\} < 2 : \{1\} < 2 : \{1, 3\} = 2 : \{1, 3\} < 3 : \{2, 2\}$, and assigns the nodes with new labels as $1, 2, 3, 3, 4$ where two nodes have the same label 3 as they have the same signature. The algorithm can terminate when the labels are stable. Based on the labeling results, frequency-based encoding generates $\boldsymbol{x}_u^{(s)} = [1, 1, 2, 1; 1, 1, 2, 1]$ if only two iterations are considered, as label 3 occurs twice while other labels occur once in each iteration. For embedding-based encoding, the method generates two sequences, $\langle 1\_1, 1\_2, 1\_3, 1\_3, 1\_3\rangle$ and $\langle 2\_1, 2\_2, 2\_3, 2\_3, 2\_3\rangle$, where each label, say 2_1, represents that a node is labeled with 1 in the second iteration. Then, it trains label embeddings based on the sequences and then computes $\boldsymbol{x}_u^{(s)}$ by averaging label embeddings.

Time complexity of the algorithm is $O((m_r + n_r \log n_r) \cdot h \cdot |\mathcal{V}|)$ where m_r and n_r are respectively average edge number and node number in a neighborhood with parameter r, and h is the number of iterations in neighborhood labeling.

Modeling Social Influence Dynamics

We aim to learn influence *dynamics* for target user u that captures both features of u's neighbors and their influences to u. The motivation is that, if u observes some "close" neighbors have performed actions, u may be more likely to follow her neighbors. Similar to (Qiu et al. 2018), we leverage a recently proposed technique graph attention networks (GAT) (Velickovic et al. 2017) to model influence dynamics.

GAT introduces the attention mechanism into Graph Convolutional Network (GCN) (Kipf and Welling 2016), using self-attention strategy to compute the hidden representation of each node by attending over its neighbors. We apply GAT over the neighborhood Γ_u of user u. Formally, GAT defines a matrix $A = [a_{ij}]_{n\times n}$ to measure importance between any two users v_i and v_j in neighborhood Γ_u. It computes this matrix as follows. Let $\mathcal{H}_u = \{\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_n\}$ denote features of n nodes in u's neighborhood (we will describe the details of node features later). Based on $\mathcal{H}_u$, GAT computes attention coefficient e_{ij} as $e_{ij} = \texttt{attn}(W\boldsymbol{h}_i, W\boldsymbol{h}_j)$, where a shared matrix W is applied to every node to transform original vertex features into more expressive ones, and `attn` is the attention function. The design of the attention function in GAT follows Velickovic et al. (Velickovic et al. 2017) using a single-layer feed-forward neural network, i.e.,

$$e_{ij} = LeakyReLU(c^T[W\boldsymbol{h}_i || W\boldsymbol{h}_j]), \tag{4}$$

where $||$ is vector concatenation, c^T is weight parameters, and $LeakyReLU$ is a nonlinearity activation function.

Then, a softmax function is applied to normalize the attention coefficients, $a_{ij} = softmax(e_{ij}) = \frac{exp(e_{ij})}{\sum_{v_k\in\Gamma_u} exp(e_{ik})}$.

Based on matrix $A = [a_{ij}]_{n\times n}$, we obtain influence dynamics, through stacked GAT layers, as a linear combination of its neighbors with a non-linear ReLU activation function.

$$\boldsymbol{h}_u^{(l+1)} = \sigma(\sum_{v_j\in\Gamma_u} a_{uj} W\boldsymbol{h}_j^{(l)}). \tag{5}$$

where $\boldsymbol{h}_j^{(l)}$ refers to the representation of node v_j in the l-th GAT layer. In particular, $\boldsymbol{h}_j^{(0)} = \boldsymbol{h}_j$. Moreover, suppose that we have L GAT layers, we generate influence dynamic features $\boldsymbol{x}_u^{(d)}$ as the output of the last layer.

To complement node features $\boldsymbol{h}_j$, we consider three kinds of features. The first one is user profile features. We want users with similar profiles would have larger attention coefficient e_{ij}. The second one is network homophily features captured by node2vec (Grover and Leskovec 2016). The intuition is that users close to each other in the network should have larger attention coefficient. The third one is v_j's status on performing social actions on recent ads. We concatenate

Table 1: Dataset statistics. $|V|$, $|E|$ and Deg are # of users and # of edges and average degree of the SIN respectively. N is # of ad exposure instances.

Dataset	$\|V\|$	$\|E\|$	Deg	N
WechatDay	4,815,987	15,119,817	3.14	1,448,735
WechatWeek	4,815,987	15,119,817	3.14	5,311,495
Weibo	624,687	4,681,881	7.49	2,269,140

three feature vectors as $\boldsymbol{h}_j$ and using the vectors to feed into the GAT model. We implement multi-head GAT (Velickovic et al. 2017), which is an extension of the above single-head graph attention.

Experiments

Experiments Setup

Datasets. We evaluate the approaches on three real datasets from two domains, *social in-feed advertising* and *Weibo*. Table 1 shows the statistics of the datasets.

(1) Advertising Datasets. We construct large-scale datasets based on the real ad exposure logs from WeChat. Each element in the log is an ad exposure with social action status. The intrinsic features of each ad exposure include: 1) user profile of u, e.g., age, gender, and education; 2) ad attributes of a, e.g., industry category and media format; 3) context features, e.g., OS and network types. We prepare two datasets, WechatDay and WechatWeek, by extracting *one day* and *one week* data from the logs respectively. As the datasets are very imbalanced in label class, we apply the negative sampling method to keep the ratio between positive and negative instances to be 1:5. We construct an SIN for the datasets by considering user historical interactions within 2 months *immediately before* the periods of the datasets. The SIN is very sparse as observed in Table 1.

(2) Weibo dataset. As social in-feed advertising datasets contain sensitive personal information of users, the datasets WechatDay and WechatWeek cannot be published for testing the reproducibility of our proposed approach. Thus, we also conduct experiments on an open dataset Weibo[3] (Zhang et al. 2013), which is from the most popular Chinese microblogging platform. We regard social action on this dataset as user's *repost* (retweet) behavior. We also use negative sampling with the ratio of 1:5. We build an SIN by using historical repost records, and define *influence* relationship from user u to v if v follows u and v reposts at least one microblog of u's reposts.

Comparison methods. We implement our framework shown in Figure 1 that first learns social influence features and then utilizes a prediction model to produce the results. For a comprehensive evaluation, we examine three prediction models. (1) LR, the simple logistic regression model. (2) DeepFM (Guo et al. 2017), a factorization-machine based neural network (3) ResFM, an improvement of DeepFM model, replacing DNN in DeepFM with residual network (He et al. 2016) to solve the problems of gradient disappearance/explosion and network degradation in DNN with the increase of network depth.

For social influence learning, we compare with the state-of-the-art methods. We tune the parameters of the methods to achieve the best performance. First, for modeling influence structure $\boldsymbol{x}_u^{(s)}$, we evaluate the following methods. (1) Node2vec utilizes network embedding techniques (Grover and Leskovec 2016) to generate structure embedding of target user u with the parameter setting: *window-size* is 10, *walk-length* is 20, *num-walks* is 20, p is 1 and q is 2. (2) WL_{Frq} and WL_{Emb} are our proposed methods based on the WL algorithm, where WL_{Frq} is frequency-based encoding and WL_{Emb} is embedding-based encoding.

Second, for modeling influence dynamics $\boldsymbol{x}_u^{(d)}$, we evaluate the following three methods. (1) GCN utilizes the graph convolutional network (Kipf and Welling 2016) to learn influence from near neighbors. GCN is implemented by stacking two GCN layers and 16 neurons per layer. (2) GAT is the graph attention network method. We implement multi-head GAT (Velickovic et al. 2017) in which each GAT layer consists of $K = 8$ attention heads. Node features $\boldsymbol{h}_j$ of GCN and GAT are described previously. (3) We also compare the method WL+GAT that includes both graph encoding and GAT for influence learning.

To evaluate the effect of social influence features, we compare with plain-LR, plain-DeepFM and plain-ResFM that *only* consider intrinsic ad exposure features (i.e., the gray vector in Figure 1). Furthermore, we compare the proposed framework against a very recent approach DeepInf (Qiu et al. 2018), which also utilizes social influence for predicting user actions.

Evaluation metrics. We use *Area Under Curve* (AUC), which is defined as the area enclosed by the coordinate axis under the ROC curve, as the evaluation metric[4].

Hyper-parameter settings. For LR, we add the L1 and the L2 regularization to prevent model over-fitting. The DeepFM model uses a two-layer neural network with 32 hidden units and an embedding size of 16. Batch normalization with decay 0.99 is also used for deep learning models. For WL-based graph encoding, we use $r = 2$ to generate neighborhoods and set the default dimension of $\boldsymbol{x}_u^{(s)}$ as 60. For GAT-based influence dynamics modeling, we use two GAT layers where each layer has 16 neurons. We use elu as activation function in GCN/GAT and DeepFM. All parameters are initialized using a random normalization. The models are trained using Adam optimizer with logloss function, learning rate 0.001 and mini-batch size 1024. All the ad exposure instances in the datasets are split into two parts, i.e., 70% for training and 30% for testing. We use cross validation over the training set to tune hyper-parameters.

Significance test. We perform paired t-test at a 99% confidence level ($p\_value < 0.01$) to test the result differences obtained by our approach and every baseline. The test result shows significant differences of our approach with respect to all the baselines.

[3] http://aminer.org/Influencelocality

[4] We also evaluate the performance on F-measure and find similar trends with AUC. We omit the results due to the space limit.

Evaluation on Social Influence Learning

We evaluate alternative methods for social influence learning. Figures 3(a), 3(b) and 3(c) report the results under different classification models respectively.

We first evaluate the methods for modeling social influence structure. Our methods $\texttt{WL}_{\texttt{Frq}}$ and $\texttt{WL}_{\texttt{Emb}}$ significantly outperform baseline Node2vec on the advertising datasets. This is because Node2vec mainly focuses on learning homophily of the users. However, users closely connected together in the network may not have similar behaviors on social actions. In contrast, WL-based methods focus on learning topological features, which are more important than homophily for predicting actions on advertising datasets. $\texttt{WL}_{\texttt{Emb}}$ achieves better performance than $\texttt{WL}_{\texttt{Frq}}$ under the LR model. The reason is that $\texttt{WL}_{\texttt{Emb}}$ learns embeddings of node labels and can overcome the sparsity and non-independent issues caused by simply counting frequencies in $\texttt{WL}_{\texttt{Frq}}$. Nevertheless, the superiority of $\texttt{WL}_{\texttt{Emb}}$ becomes less significant under DeepFM and ResFM, because the deep models can learn the latent states of label frequencies. We also find that the improvement of $\texttt{WL}_{\texttt{Emb}}$ over Node2vec is more significant on the advertising datasets, compared with the Weibo dataset. This is attributed to the reposting behavior in Weibo. Different from social actions in advertising, reposting tends to have community-based diffusion, and Node2vec is capable of learning nodes within common communities.

Then, we examine the methods for modeling social influence dynamics. Our proposed method WL+GAT combining $\texttt{WL}_{\texttt{Emb}}$ and GAT to learn both social influence structure and dynamics performs the best. The results verify topological features of user's neighbors are indispensable for action prediction. We also find the improvement is more remarkable on the advertising datasets. This is because social actions in advertising are very sparse and limited information of neighbor statuses and features can be used. In this case, topological features play a more important role.

Comparison on Action Prediction

As observed in Table 2, compared with the plain-LR plain-DeepFM, and plain-ResFM, the consideration of social influence features can improve the prediction performance with large margin, e.g., $4-5\%$ improvements on AUC on WechatDay and WechatWeek. We also observe that a simple LR model considering social influence can beat a complicated DeepFM model without social influence features. The experimental results validate that social influence plays an important role in advertising action prediction.

We also observe that the classification model ResFM achieves superior performance on the advertising datasets. That is because most of the advertising data are sparse features, which are easy to produce the gradient disappearance/explosion phenomenon. The introduction of residual network can solve this problem, and the performance is improved compared with the ordinary DNN in DeepFM model. With the increase of network depth, the performance improvement of residual network would be more significant. Moreover, we find DeepInf cannot achieve good performance in our tasks. This is attributed to the *close world* assumption of DeepInf: Although it utilizes GAT to model predictive signals in social influence, it solely relies on these signals to produce prediction results. However, this may not be sufficient for prediction, due to the high sparsity.

Table 2: Social Action Prediction Performance on AUC.

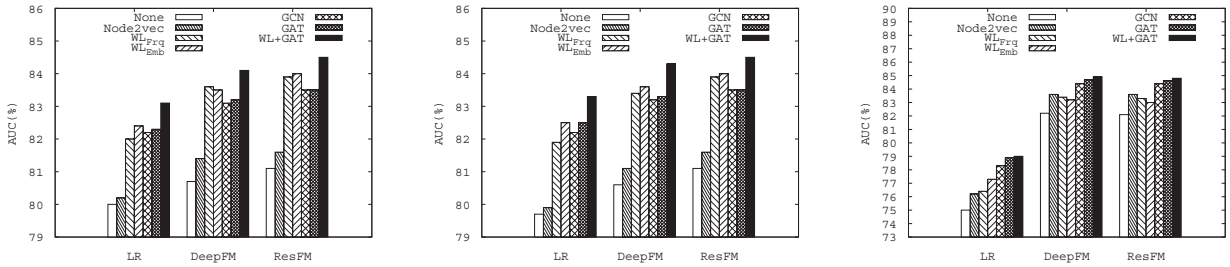
Dataset	Method	AUC (%)
WechatDay	plain-LR	80.0
	plain-DeepFM	80.7
	plain-ResFM	81.0
	DeepInf	69.4
	Ours (LR)	83.1
	Ours (DeepFM)	84.1
	Ours (ResFM)	**84.4**
WechatWeek	plain-LR	79.7
	plain-DeepFM	80.6
	plain-ResFM	81.0
	DeepInf	70.5
	Ours (LR)	83.3
	Ours (DeepFM)	84.3
	Ours (ResFM)	**84.4**
Weibo	plain-LR	75.0
	plain-DeepFM	82.2
	plain-ResFM	82.0
	DeepInf	77.1
	Ours (LR)	79.0
	Ours (DeepFM)	**84.9**
	Ours (ResFM)	84.8

Visualization of Influence Structure

We use the WechatDay dataset to visualize the *topological structures* learned by our graph encoding approach. Then, given ad exposure instances, we select the most frequent $\boldsymbol{x}_u^{(s)}$ vectors in the positive and negative instances respectively. Figure 4 depicts some extracted enclosing neighborhoods corresponding to these encoding vectors where the target users are denoted by the orange circles. We have an interesting observation that the most frequent *positive* neighborhoods enclose target user u in the "center", while the most frequent *negative* ones enclose u in the "peripheral". This supports our claims: users, who are influenced by various neighbors in the history, tend to be influenced in the future. In contrast, social influence may not affect the ones who are rarely influenced by neighbors previously.

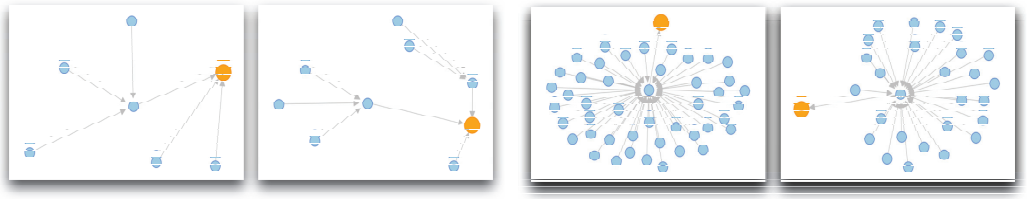
Hyper-parameter Analysis

We examine prediction performance under different hyper-parameters on the WechatDay dataset. We first examine graph encoding dimension in Figure 5(a): as the dimension increases, the prediction performance first improves and then declines. This is because if the dimension is too large, the topological features will be too sparse, which will affect the prediction performance. As a result, we set the default dimension as 60 in our experiments. We also examine embedding size in DeepFM and report the results in Figure 5(b). Different embedding sizes have an impact on the results, but our approach WL+GAT consistently performs the best.



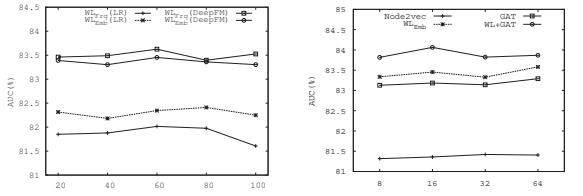
(a) WechatDay dataset. (b) WechatWeek dataset. (c) Weibo dataset.

Figure 3: Evaluating social influence learning.



(a) Most frequent positive influence structures. (b) Most frequent negative influence structures.

Figure 4: Visualization of learned social influence structures.



(a) Encoding dimension. (b) DeepFM embedding size.

Figure 5: Hyper-parameter analysis.

Related Work

Click-Through Rate (CTR) Prediction. Recently, CTR prediction has been extensively studied. A large number of deep models are introduced with a general framework that consists of *embedding* and *deep neural networks*. Wide & Deep (Cheng et al. 2016) and DeepFM (Guo et al. 2017) combine embeddings and deep neural networks in parallel while FNN (Zhang, Du, and Wang 2016) and PNN (Qu et al. 2016) combine these two parts in series. NFM (He and Chua 2017) adds element-by-element multiplication to make full use of the information of the second-order feature interaction. Unlike click behavior, social actions of a user in the scenario of in-feed advertising would be very likely influenced by her friends. The existing works do not consider social influence for prediction.

Social Influence Modeling. Conventional social influence studies can be categorized into pairwise (Goyal, Bonchi, and Lakshmanan 2010; Li et al. 2018b), topic-level (Tang et al. 2009; Chen et al. 2015; Fan et al. 2018) and structure-level influence (Zhang et al. 2013; Li et al. 2017). DeepInf (Qiu et al. 2018) is a recently proposed deep learning approach that utilizes graph attention networks (GAT) for micro-level influence prediction. In this paper, we propose to learn topological features of users' historical interaction on social influence, and develop a structure-aware graph encoding approach, which shows superiority in our experiments.

Network Embedding. Network Embedding learns low-dimensional representation of nodes in the network. Proximity based embedding methods such as deep walk (Perozzi, Al-Rfou, and Skiena 2014), node2vec (Grover and Leskovec 2016) are based on random walk. LINE (Tang et al. 2015) and SDNE (Tu et al. 2018) find proximity among nodes. Graph convolution based embedding includes non-spectral approaches (Niepert, Ahmed, and Kutzkov 2016) and spectral approaches (Defferrard, Bresson, and Vandergheynst 2016; Li et al. 2018a). Structural equivalence based methods include subgraphs (Shervashidze et al. 2009), random walks or paths (Vishwanathan et al. 2010) and subtree patterns (Shervashidze and Borgwardt 2009; Yanardag and Vishwanathan 2015). In our paper, we leverage graph encoding techniques inspired by Weisfeiler-Lehman Kernel (Shervashidze et al. 2011; Yanardag and Vishwanathan 2015) to solve a new problem, i.e., learning influence structure.

Conclusion

In this paper, we have studied the problem of social action prediction for in-feed advertising. We introduced an end-to-end approach to learn predictive signals in social influence. We focused on learning topological features to model

social influence structures, and developed structure-aware graph encoding methods. We also modeled influence dynamics features by combining neighbors' features and action influences. We conducted extensive experiments on the real in-feed adverting datasets and an open dataset from Weibo to show the performance superiority of our approach.

Acknowledgement. This work was partly supported by National Key Research and Development Plan of China (2018YFB1800302), NSF China (61602488, 61632016, U1711261, 61772345), and Tencent Marketing Solution Rhino-Bird Focused Research Program.

References

Chen, S.; Fan, J.; Li, G.; Feng, J.; Tan, K.; and Tang, J. 2015. Online topic-aware influence maximization. *PVLDB* 8(6):666–677.

Chen, J.; Sun, B.; Li, H.; Lu, H.; and Hua, X. 2016. Deep CTR prediction in display advertising. In *ACM MM*, 811–820.

Cheng, H.; Koc, L.; Harmsen, J.; Shaked, T.; Chandra, T.; Aradhye, H.; Anderson, G.; Corrado, G.; Chai, W.; Ispir, M.; Anil, R.; Haque, Z.; Hong, L.; Jain, V.; Liu, X.; and Shah, H. 2016. Wide & deep learning for recommender systems. In *DLRS@RecSys*, 7–10.

Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, 3837–3845.

Douglas, B. L. 2011. The Weisfeiler-Lehman Method and Graph Isomorphism Testing. *ArXiv e-prints*.

Edelman. 2013. 2013 edelman trust barometer. https://www.edelman.com/research/2013-edelman-trust-barometer.

Fan, J.; Qiu, J.; Li, Y.; Meng, Q.; Zhang, D.; Li, G.; Tan, K.; and Du, X. 2018. OCTOPUS: an online topic-aware influence analysis system for social networks. In *ICDE*, 1569–1572.

Fan, S.; Lu, Y.; and Gupta, S. 2017. Social media in-feed advertising: the impacts of consistency and sociability on ad avoidance. In *PACIS*, 190.

Goyal, A.; Bonchi, F.; and Lakshmanan, L. V. S. 2010. Learning influence probabilities in social networks. In *WSDM 2010*, 241–250.

Grover, A., and Leskovec, J. 2016. node2vec: Scalable feature learning for networks. In *SIGKDD*, 855–864.

Guo, H.; Tang, R.; Ye, Y.; Li, Z.; and He, X. 2017. Deepfm: A factorization-machine based neural network for CTR prediction. In *IJCAI 2017*, 1725–1731.

He, X., and Chua, T. 2017. Neural factorization machines for sparse predictive analytics. In *SIGIR*, 355–364.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*, 770–778.

Kipf, T. N., and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *CoRR* abs/1609.02907.

Li, Y.; Fan, J.; Zhang, D.; and Tan, K. 2017. Discovering your selling points: Personalized social influential tags exploration. In *SIGMOD*, 619–634.

Li, R.; Wang, S.; Zhu, F.; and Huang, J. 2018a. Adaptive graph convolutional neural networks. In *AAAI*, 3546–3553.

Li, Y.; Fan, J.; Wang, Y.; and Tan, K. 2018b. Influence maximization on social graphs: A survey. *IEEE Trans. Knowl. Data Eng.* 30(10):1852–1872.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient estimation of word representations in vector space. *CoRR* abs/1301.3781.

Niepert, M.; Ahmed, M.; and Kutzkov, K. 2016. Learning convolutional neural networks for graphs. In *ICML*, 2014–2023.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. Deepwalk: online learning of social representations. In *SIGKDD*, 701–710.

Qiu, J.; Tang, J.; Ma, H.; Dong, Y.; Wang, K.; and Tang, J. 2018. Deepinf: Social influence prediction with deep learning. In *SIGKDD*, 2110–2119.

Qu, Y.; Cai, H.; Ren, K.; Zhang, W.; Yu, Y.; Wen, Y.; and Wang, J. 2016. Product-based neural networks for user response prediction. In *ICDM*, 1149–1154.

Shervashidze, N., and Borgwardt, K. M. 2009. Fast subtree kernels on graphs. In *NIPS*, 1660–1668.

Shervashidze, N.; Vishwanathan, S. V. N.; Petri, T.; Mehlhorn, K.; and Borgwardt, K. M. 2009. Efficient graphlet kernels for large graph comparison. In *AISTATS*, 488–495.

Shervashidze, N.; Schweitzer, P.; van Leeuwen, E. J.; Mehlhorn, K.; and Borgwardt, K. M. 2011. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research* 12:2539–2561.

Tang, J.; Sun, J.; Wang, C.; and Yang, Z. 2009. Social influence analysis in large-scale networks. In *KDD*, 807–816.

Tang, J.; Qu, M.; Wang, M.; Zhang, M.; Yan, J.; and Mei, Q. 2015. LINE: large-scale information network embedding. In *WWW*, 1067–1077.

Tu, K.; Cui, P.; Wang, X.; Wang, F.; and Zhu, W. 2018. Structural deep embedding for hyper-networks. In *AAAI*, 426–433.

Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2017. Graph attention networks. *CoRR* abs/1710.10903.

Vishwanathan, S. V. N.; Schraudolph, N. N.; Kondor, R.; and Borgwardt, K. M. 2010. Graph kernels. *Journal of Machine Learning Research* 11:1201–1242.

Yanardag, P., and Vishwanathan, S. V. N. 2015. Deep graph kernels. In *SIGKDD*, 1365–1374.

Zhang, M., and Chen, Y. 2017. Weisfeiler-lehman neural machine for link prediction. In *SIGKDD*, 575–583.

Zhang, J.; Liu, B.; Tang, J.; Chen, T.; and Li, J. 2013. Social influence locality for modeling retweeting behaviors. In *IJCAI*, 2761–2767.

Zhang, W.; Du, T.; and Wang, J. 2016. Deep learning over multi-field categorical data - - A case study on user response prediction. In *ECIR*, 45–57.