AtNE-Trust: Attributed Trust Network Embedding for Trust Prediction in Online Social Networks

Qi Wang*, Weiliang Zhao†*, Jian Yang*, Jia Wu*, Chuan Zhou‡, and Qianli Xing*
*Department of Computing, Faculty of Science and Engineering, Macquarie University, Sydney, Australia
†College of Computer Science and Technology, Donghua University, Shanghai 201600, China
‡Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, China
Email: qi.wang20@students.mq.edu.au, {weiliang.zhao, jian.yang, jia.wu}@mq.edu.au,
zhouchuan@amss.ac.cn, qianli.xing@students.mq.edu.au

***Abstract*—Trust relationship prediction among people provides valuable supports for decision making, information dissemination, and product promotion in online social networks. Network embedding has achieved promising performance for link prediction by learning node representations that encode intrinsic network structures. However, most of the existing network embedding solutions cannot effectively capture the properties of a trust network that has directed edges and nodes with in/out links. Furthermore, there usually exist rich user attributes in trust networks, such as ratings, reviews, and the rated/reviewed items, which may exert significant impacts on the formation of trust relationships. It is still lacking a network embedding-based method that can adequately integrate these properties for trust prediction. In this work, we develop an AtNE-Trust model to address these issues. We firstly capture user embedding from both the trust network structures and user attributes. Then we design a deep multi-view representation learning module to further mine and fuse the obtained user embedding. Finally, a trust evaluation module is developed to predict the trust relationships between users. Representation learning and trust evaluation are optimized together to capture high-quality user embedding and make accurate predictions simultaneously. A set of experiments against the real-world datasets demonstrates the effectiveness of the proposed approach.**

***Index Terms*—Trust prediction, Attributed network embedding, Online social networks**

I. Introduction

Trust prediction among users provides critical supports for reliable marketing, information dissemination, and recommendation on social networks. An emerging solution for link (trust) prediction in complicated networks is network embedding, which learns representations for each user by reconstructing the network structures in low-dimensional spaces.

Trust relationships between people are asymmetric, i.e., that A trusts B does not necessarily mean B trusts A the same way. For example, the users in the online review website *Epinions* can add other users to a *trust* list when finding their reviews useful. Such relationships have directions in which we need to distinguish the user's role as a *trustor* or as a *trustee*. Link directions are important information in trust networks [1]. However, most of the existing network embedding methods mainly focus on networks with symmetric relationships. Furthermore, we often observe from a social network that a large number of users only maintain several relationships while a small number of users maintain a large

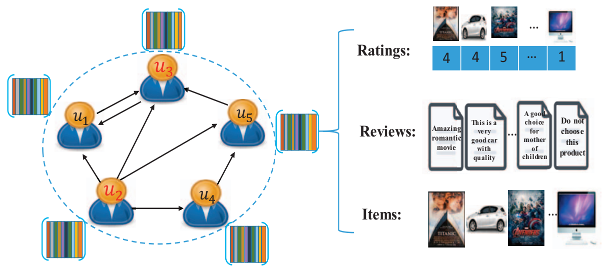

u_1
u_2
u_3
u_4
u_5
Ratings:
4
4
5
...
1
Reviews:
Amazing romantic movie
This is a very good car with quality
A good choice for mother of children
Do not choose this product
Items:

Fig. 1. An example of attributed trust network. As shown in the figure, there exist trust relationships among users and each user may have dual roles as a *trustor* or *trustee*. Also, different users have different connectivity properties, e.g., u_2 has more out-links while u_3 has more in-links. In addition, there are user attributes including ratings, reviews, and rated/reviewed items for each user.

number of relationships. For example, a well-known celebrity tends to have more in-links while an advertiser has more out-links to disseminate information (See Fig.1). Therefore, capturing the roles in a relationship and the connectivity properties of users is crucial for trust network embedding [2].

Trust networks have their particular structures on how the nodes are connected and node attributes with information about the users. As an example, Fig.1 shows that users provide ratings and comments/reviews on items. In addition, the items rated/reviewed by the users also reflect users' interests/preferences. We call the information revealing the users' characteristics as user *attributes*. Based on the social science theory [3], attributes usually exert crucial impacts on the formation of trust relationships. There have been some researches on analyzing the importance of attributes such as user demographics [4] and user subjective preferences [5]. The structure of a trust network and user attributes provide useful sources of information for trust relationship prediction. It is necessary to capture both of them in network embedding to learn a comprehensive representation of the user's social characteristics for trust prediction.

Learning attributed trust network representations faces the following challenges: (1) Trust Properties-preserving: different from the normal symmetric social networks without directions, the underlying structure of the trust network is usually complex. How to simultaneously preserve the dual roles and the connectivity properties of users is a hard problem; (2) Multi-view Non-linearity User Attributes: social network users are generally associated with the multi-view data such as the ratings, reviews/comments, and the rated/reviewed items. All the views combined form the characteristics of a user in a social network, but it is a challenging task to represent and fuse these high-dimension and non-linearity [6] data; (3) Data Sparsity: in reality, many trust networks are often very sparse with a limited number of observed links, which is not enough to obtain informative user embedding.

To address the above challenges, we propose a novel attributed trust network embedding model AtNE-Trust for trust relationship prediction in online social networks. AtNE-Trust firstly performs a uniformed high non-linear trust network embedding to capture the dual roles and the connectivity properties of users. A set of rich user attributes including user ratings, reviews, and the rated/reviewed items are pre-processed for user attributes embedding. Different from the previous works that learn representations separately, we propose a multi-view representation learning module including a set of auto-encoders and a feature fusion unit to further learn user embedding. Finally, the learned embedding for pairs of users is concatenated and fed into a trust evaluation module including an MLP (Multi-Layer Perceptron) unit to predict their trust relationships. Representation learning and trust evaluation are jointly trained. The trust evaluation loss from MLP is propagated back to representation learning and guides it to capture trust-related features. Since we utilize the user attributes and integrate the diverse information, the data sparsity problem is alleviated. The main contributions of this work are summarized as follows:

- We formally specify the problem of attributed trust network embedding, based on which we propose a trust prediction solution.
- We propose an attributed network embedding-based model to capture the trust network structures and user attributes simultaneously: for trust network embedding, the dual roles and the connectivity properties of users are considered; for user attributes embedding, the attributes are considered based on user behaviors including ratings, reviews, and the rated/reviewed items. The consideration of rich user attributes alleviates the issue of data sparsity.
- We develop a unified deep learning approach by integrating representation learning and trust evaluation. The mutual refinement between the two modules ensures the effectiveness of both generating high-quality representations and achieving better trust prediction results.
- Experiments on real-world datasets demonstrate the superior performance of the proposed trust prediction model over both classic and state-of-the-art solutions.

The rest of the paper is structured as follows. Section II reviews the related work. Section III provides the preliminaries including notation, problem specification, and some analysis on links and attributes. Section IV proposes the AtNE-Trust approach. Section V describes the experiments conducted and analyses the results. Finally, Section VI concludes the work.

II. Related Work

In this section, we review the existing trust prediction approaches in the following three categories: (1) trust network structure-based approaches, (2) low-rank approximation-based approaches and (3) network embedding-based approaches.

A. Trust Network Structure-based Approaches

Trust network structure has been widely exploited by existing trust prediction methods. The solution proposed in [7] utilizes the transitivity property of trust to propagate trust values from a source user to a target user along a path between them and treats all the propagation paths equally. Later on, researchers find that shorter propagation paths and paths with higher trust values produce more accurate trust evaluations [8]. By averaging the trust values along social paths, algorithms for inferring the trust relationships between users that are not directly connected are proposed in [9]. Different from the above propagation-based approaches, the neighborhood structure of a trust network is considered in some studies. For example, the trust value between a pair of user (u_i,u_j) is calculated according to the suggestions from u_i's neighbors. In detail, the stronger u_i trusts her/his trustees, the higher weighs these trustees carry when aggregating their suggestions [8], [10].

Network structure-based trust prediction approaches normally suffer from the data sparsity problem since the number of trust relationships may be too small.

B. Low-rank Approximation-based Approaches

Low-rank approximation based method is widely employed in various applications such as collaborative filtering [11], [12] and document clustering [13]. Matrix Factorization (MF) is the most widely employed low-rank approximation method to generate the low-rank representations of users and their correlations by factorizing a trust matrix [13]. Furthermore, by incorporating prior knowledge and additional user-generated content, the performance of low-rank approximation-based approaches can be further improved. Homophily effect is studied and incorporated as the rating similarity regularization to matrix factorization [3]. Social status regularized matrix factorization is proposed in [14]. This is based on the assumption that users with lower social status are more likely to trust users with higher status. Emotional information for trust/distrust prediction is investigated [15] by regularizing trust based on users' positive/negative emotions. Recently, a power-law distribution aware trust prediction model is proposed under the framework of matrix factorization [16].

The low-rank approximation-based approaches suffer from the data sparsity problem since these approaches conduct factorization directly on the sparse trust matrix.

C. Network Embedding-based Approaches

Recent works focus more on leveraging neural network models to embed an existing network into a low-dimensional

TABLE I

Notation	Description
$U = \{u_1, \cdots, u_m\}$	the set of m users
$V = \{v_1, \cdots, v_n\}$	the set of n items
$C = \{c_1, \cdots, c_q\}$	the set of q item categories
$E_{m\times m}$	trust relationship matrix of users
$e_{ij} \in E_{m\times m}$	the trust value of user i on user j
$R_{m\times n}$	rating matrix of users on items
$r_{ij} \in R_{m\times n}$	the rating value of user i on item j
$RE_{m\times n}$	review matrix of users on items
$re_{ij} \in RE_{m\times n}$	the piece of review of user i written for item j
$IT_{m\times q}$	user and item category matrix
it_{ij}	the items of category j rated/reviewed by user i

space for link prediction. In [17], the authors deploy truncated random walks on networks to generate node sequences, which is treated like that sentences are fed to the Skip-Gram model to learn the embedding in language models. The work in [18] modifies the way of generating node sequences by balancing breadth-first sampling and depth-first sampling, which achieves significant performance improvement. Instead of performing simulated "walks" on the networks, [19] proposes clear objective functions to preserve the first-order proximity and second-order proximity of nodes. Recently, a representation learning method to capture the sign and direction information is proposed in [20]. In online social networks, there usually exists rich user attributes and purely structure-based methods fail to capture such valuable information. Accordingly, there have been some works to integrate contents for informative representations learning [21]. For example, TADW [22] proposes a text-associated DeepWalk to incorporate text features into the matrix factorization framework. ASNE is proposed to capture the structure proximity and the attributes proximity simultaneously, of which the attributes are mainly obtained from user profiles for friendship-based networks [23]. However, user profile information is usually limited. More rich user attributes should be considered and combined together for obtaining better user embedding. Most importantly, almost all the network embedding-based approaches are designed for the normal link prediction task [24]. However, trust prediction is a special kind of link prediction task, of which the trust relationships among users are stronger than the normal links (e.g., friendship). Although existing network embedding-based works can be used for trust prediction task but a more effective network embedding-based method that specially considers the influential factors on trust relationships formation is essential for trust prediction.

III. Preliminaries

In this section, we first introduce the notations and specify the attributed trust network embedding problem. Then we investigate the relations between attributes and link formation in trust networks against real-world datasets.

A. Notation

For the presentation of the problem specification and methodology in the following sections, we list the notations of the raw input data in Table I.

TABLE II
Average of user attribute similarities

Dataset	*Metric*	*T*	*R*
Epinions	CA_sim	70.49	22.67
	CS_sim	0.0583	0.0124
Ciao	CA_sim	53.14	17.46
	CS_sim	0.0315	0.0116

B. Problem Specification

Attributed Trust Network Embedding: An attributed trust network is $G = (U, E, X)$, where U is the set of users (nodes) and E is the set of trust relationships (edges) between the users (nodes). Each $e_{ij} \in E$ is associated with a trust value 0 or 1. e_{ij} = 1 if there exits an edge from u_i to u_j. Otherwise, e_{ij} = 0. X is the matrix representing the attributes associated with each user (node). Attributed trust network embedding (AtNE) aims to learn an embedding function f as follows:

$$f : U \rightarrow \mathbb{R}^d \tag{1}$$

which preserves both the trust network structures and user attributes.

C. Analysis on Trust Relationships and User Attributes

As it has been widely studied in the literature that users with similar attributes are more likely to have trust relationships. This shows that attributes play an important role in the formation of trust relationships. Following the research in [25], we perform data analysis to investigate the relationships between trust relationships and user attributes on dataset *Epinions* and *Ciao*.

Epinions and *Ciao* are two popular online product review websites, where users establish trust relationships with other users and provide reviews for products. In this subsection, we use the reviews written by users to construct a user-attributes matrix X by bag-of-words. For each user u_i, we construct a trust relationships set T and an unobserved links set R as follows:

$$\begin{aligned} T = (u_i, u_j)|u_j \in T_i, i = 1, ..., n \\ R = (u_i, u_j)|u_j \in R_i, i = 1, ..., n \end{aligned} \tag{2}$$

where T_i and R_i correspond to the users who have trust relationships with u_i and the randomly selected users who have unobserved relationships with u_i, respectively. We can then calculate the user attributes similarity for each pair of users (u_i, u_j) in T and R. In this work, we investigate two ways of calculating the similarity between user pairs (u_i, u_j) as follows:

- *Common Attributes*: We compute the the number of common attributes between two users (u_i, u_j) as their common attributes similarity $CA\_sim(u_i, u_j)$;
- *Cosine Similarity*: We compute the cosine similarity between the attributes vectors of two users (u_i, u_j) as their cosine similarity $CS\_sim(u_i, u_j)$.

We then compute the similarities between each pair of users in P and R by these two ways, which form two similarity vectors denoted as s^t and s^r. The average values of CA_sim and CS_sim are shown in Table II, where we find that users are more likely to have more similar attributes with their trusted users than other users.

To further verify the above observation statistically, we conduct a two-sample t-test on two similarity vectors s^t, s^r. The null hypothesis H_0 and the alternative hypothesis H_1 are defined as follows:

$$H_0 : s^t \leqslant s^r \qquad H_1 : s^t > s^r \tag{3}$$

H_0 indicates that the users with trust relationships have less common attributes than that of randomly selected users with unobserved links while H_1 has the opposite meaning. According to calculation, H_0 is rejected at significance level $\alpha = 0.01$ with $p = 7.32e-283$, which verifies that users with trust relationships have more common attributes than users with unobserved links. It shows that user attributes do have impacts on link formation and have the potential to learn better user embedding for trust relationship prediction.

The observation can be easily explained from a user's perspective. Consider a case that u_i has more common attributes with u_j, which may be explained by their common interest in a particular subject such as *Sports*. In such a case, it is more likely for u_i to know u_j and have interactions with u_j, e.g., constructing trust relationships. On the contrary, if u_i has few attributes in common with u_j, then u_i is not likely to know u_j at all, let alone have interactions.

IV. Methodology

In this section, we present our AtNE-Trust prediction approach. The AtNE-Trust consists of four components which are illustrated in Fig. 2 as: 1) **Trust Network Embedding** covers the dual roles and the connectivity properties of users based on the trust network structures from the *Input Layer*; 2) **User Attributes Embedding** covers user attributes from available data including users' ratings, reviews and rated/reviewed items from the *Input Layer*; 3) **Representation Learning** feeds the obtained embedding into the *Embedding Layer* that includes a set of auto-encoders and a *Fusion Layer* for further representation learning. 4) **Trust Relationship Evaluation** concatenates the features for each pair of users and predicts their trust relationships by a *Prediction Layer*.

A. Trust Network Embedding

Trust network embedding is conducted by preserving the dual roles and the connectivity properties of users. It mainly consists of two steps: (1) Random Walk Generation and (2) Likelihood Optimization. Algorithm 1 performs the trust network embedding.

1) Random Walk Generation: In the first stage of trust network embedding, multiple truncated random walks are generated from each seed node on the graph derived from the trust network. Each step of the walk follows directed edges according to transition probabilities proportional to weights on edges until the required length l is satisfied. If the random walk encounters dead end, the remaining steps will restart

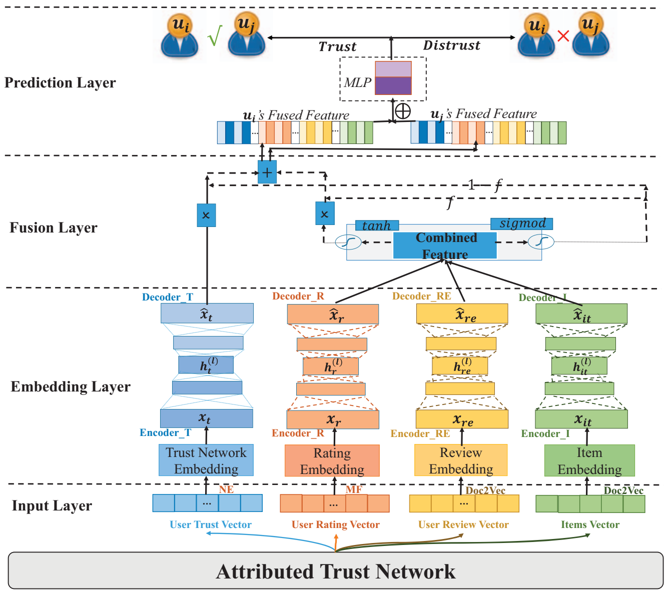


Fig. 2. Overview of the proposed AtNE-Trust model.

from the seed node. The resulting walk sequences cover all the visited users on the followed edges. For each node, the random walk process may perform w times. Two nodes are defined to be a co-occurring pair if they are placed within the short distance in the generated random walk sequences. Then all the co-occurring pairs within a window size c are selected from the walk sequences. All these co-occurring pairs will be used for the following likelihood optimization process.

2) Likelihood Optimization: In the second stage, the vector representations are learned with a neural language model Skip-Gram with negative sampling (SGNS) [26], [27]. SGNS is formulated as a likelihood maximization model by predicting whether a pair of nodes co-occur or not in the simulated random walk. Direct prediction on neighboring nodes from the target node requires an infeasible amount of parameter updates for each node pair. For example, when the user pair (u_i, u_j) is trained, the co-occurrence probability of u_j with u_i is 1 while everything else is 0. In order to limit the number of parameters updated in each step, SGNS samples some users to update their weights. The weights of other users will not be calculated. For each pair of co-occurrence users (u, v) generated in the random walk process, the likelihood formulation function is defined as follows:

$$\begin{aligned} J = \sum_{(u,v)\in D} [-logP(u,v) + \sum_{j=1}^{n} -logP(u, v_j^{'})] \\ + \frac{\lambda}{2}(||b^{in}||^2 + ||b^{out}||^2) \end{aligned} \tag{4}$$

where D represents the user pair set which contains all the co-occurrence users generated in the random walk process. For each pair $(u, v) \in D$, s noise samples $v_j^{'}$ are randomly sampled as noise pairs. The latter part of the objective function regularizes bias terms in the likelihood function. The likelihood

Algorithm 1 Trust Network Embedding Learning Algorithm

```
Input: Directed trust network $G = (V, E)$,
dimension $d$, the number of walks per node $w$, the number of the steps per walk $l$, the size of context $c$,
negative sampling size $s$, regularization parameter $\lambda$
Output: trust embedding matrix $W^{out}$, $W^{in}$, out-link bias $b^{out}$, in-link bias $b^{in}$:
1: initialize $W^{out}$, $W^{in}$ with random values and $b^{out}$, $b^{in}$ with zeros.
2: Walks = {}
3: for $i$ from 1 to $w$ do
4:   for all $v \in V$ do
5:     Generate a random walk of length $l$ start from $v$ and append to Walks
6:   end for
7: end for
8: for all $walks \in Walks$ do
9:   for all $(u, v)$ within distance $c$ in $walk$ do
10:    update $W_u^{out}$, $W_v^{in}$ and $b_u^{out}$, $b_v^{in}$ according to Equation (6)
11:    for $j$ from 1 to $s$ do
12:      Randomly sample $v^{'} \in V$
13:      Update $W_u^{out}$, $W_v^{in}$ according to the first two steps of Equation (6)
14:    end for
15:  end for
16: end for
```

$P(u, v)$ is defined as follows:

$$P(u,v) = \begin{cases} \sigma(W_u^{out} \cdot W_v^{in} + b_u^{out} + b_v^{in}) & \textbf{if}(u,v) \in D \\ \sigma(-W_u^{out} \cdot W_v^{in}) & otherwise \end{cases} \tag{5}$$

As can be seen from the above function: the first component is the inner product by W_u^{out} and W_v^{in}, where W_u^{out} represents user's *trustor* role and W_v^{in} represents user's *trustee* role. The inner product by W_u^{out} and W_v^{in} indicates the similarity score (e.g., distance) of user pair (u, v). By maximizing the objective function (5), the likelihood of the co-occurrence user pairs increases as the inner product increases. It means that trusted users are closely placed while users without connections are placed far apart. Such results are consistent with the social homophily theory, which shows that trusted users have higher similarity (e.g., shorter distance) while disconnected users have lower similarity (e.g., larger distance); the second component of the objective function (5) is bias terms b_u^{out}, b_v^{in} modeling the connectivity property of users. According to preferential attachment theory, larger connectivity induces higher likelihood of additional link formation and thus link formation likelihood increases when the bias term increases as calculated in Equation (5).

The model is trained using gradient descent optimization. For co-occurring pair (u, v), two weight vectors W_u^{out}, W_v^{in} and two bias factors b_u^{out}, b_v^{in} are updated. The derivative of objective function $J_{(u,v)} = -logP(u, v) + \frac{\lambda}{2}(||b_u^{out}||^2 + ||b_v^{in}||^2)$ corresponding to the co-occurring pair (u, v) is as follows:

$$\begin{aligned} \frac{\partial J_{(u,v)}}{\partial W_u^{out}} &= -W_v^{in}(1 - P(u,v)) \\ \frac{\partial J_{(u,v)}}{\partial W_v^{in}} &= -W_u^{out}(1 - P(u,v)) \\ \frac{\partial J_{(u,v)}}{\partial b_u^{out}} &= -(1 - P(u,v)) + \lambda b_u^{out} \\ \frac{\partial J_{(u,v)}}{\partial b_v^{in}} &= -(1 - P(u,v)) + \lambda b_v^{in} \end{aligned} \tag{6}$$

For the noise pair $(u, v^{'})$, only two weight vectors W_u^{out}, $W_{v^{'}}^{in}$ are updated. Finally, the weight vectors W_u^{out}, W_v^{in} for the dual roles of users and the bias factors b_u^{out}, b_v^{in} for the connectivity property of users are learned. For each user, the trust network embedding is denoted as x_t.

B. User Attributes Embedding

For an online review website such as *Epinions*, there exists rich information to show user attributes. We mainly consider user attributes by users' ratings, reviews, and the rated/reviewed items. Note that we do not use the user profiles such as gender, age, and location in this work. The main reasons are: a) the number of user profile information is usually limited; b) online users tend to provide fake profile information due to privacy considerations.

Modeling User Rating Behavior: Matrix Factorization is a widely used low-dimensional factor model. The basic idea is that k unobserved latent factors influence a user's attitudes and preferences. Therefore, users and items can be projected into a joint k-dimension latent space by factorizing the user-item-rating matrix into the inner product of the user-specific matrix and the item-specific matrix. The user-specific matrix represents the user's preference for the items on k latent factors and the item-specific matrix represents the k latent attributes belonging to the items that can attract users' preferences.

Therefore, in order to obtain the user attributes delivered from rating matrix for each user u_i, we factorize the rating matrix $R_{m\times n}$ into the inner product of a user-specific latent matrix $P_{m\times k}$ and an item-specific latent matrix $Q_{n\times k}$, which is represented as:

$$R_{m\times n} = P_{m\times k} * Q_{n\times k}^{\top}. \tag{7}$$

$P_{m\times k}$ represents the relations between m users and k-dimension latent factors. Each row vector P_i in $P_{m\times k}$ denotes user u_i's attributes vector learned from rating matrix, which is represented as x_r.

Modeling User Review Behavior: As a number, a rating can only express limited information. In most cases, users usually express their full opinions by providing reviews in the form of text that contain more information to reflect a user's attributes.

Doc2vec is an unsupervised algorithm to learn k-dimension embedding feature vectors for documents with variable lengths [28]. In a review matrix RE, the i-th row of RE denoted as RE_i is the review set containing all the reviews that u_i has written. RE_i is fed into the doc2vec model to obtain u_i's attributes vector, which is calculated as:

$$x_{re} = doc2vec(RE_i) \tag{8}$$

where we use x_{re} to represent u_i's attributes vector obtained from the user's provided reviews.

Modeling Item Properties: Items rated and reviewed by a user naturally reflect the interests of the user, thus the properties that lie behind an item should be treated as a part of user attributes. It is worth mentioning that we are one of the few works that consider the item's properties as part of user attributes.

$IT_{m\times q}$ is the user and item category matrix specified in Table I. For a specific user u_i, the i-th row of $IT_{m\times q}$ is represented as a set $\{IT_{i1}, IT_{i2}, \ldots, IT_{iq}\}$, where each element represents the items belonging to different categories that the user has rated/reviewed. Among this set, IT_{ik} that contains the biggest number of items is selected. Then all item names in IT_{ik} are put into I_i. I_i is fed to the doc2vec model to obtain the item embedding vector for the user as follows:

$$x_{it} = doc2vec(I_i) \tag{9}$$

where we use x_{it} to represent the item embedding vector for a specific user u_i and this item embedding vector x_{it} is considered as a part of user attributes embedding in this work.

C. Representation Learning

After the aforementioned embedding steps, trust network embedding x_t and user attributes embedding x_r, x_{re}, x_{it} are obtained in the low-dimensional space, respectively. To further mine and fuse these information, they are fed into a multi-view representation learning module including a set of auto-encoders and a feature fusion unit.

1) Encoder: Each user has four set of embedding x_t, x_r, x_{re}, and x_{it} from different views. Taking the trust network embedding view as an example, the input of its encoder is x_t, the output of $i\text{-}th$ hidden layer is $h_t^{(i)}$, the $i\text{-}th$ weight matrix is W_{t_i}, and the $i\text{-}th$ bias term by b_{t_i}. For an encoder with l layers, the l-th layer's output is:

$$h_t^{(l)} = f(W_{t_l} h_t^{l-1} + b_{t_l}) \tag{10}$$

where $ReLU$ is the activation function f at hidden layers and the output layer. Similarly, the outputs of other encoders with l layers are obtained corresponding to other attribute views denoted as $h_r^{(l)}, h_{re}^{(l)}, h_{it}^{(l)}$.

2) Feature Fusion Unit: Incorporating all the information across all the views straightforwardly may lose the unique characteristics of each view. Therefore, we employ a feature fusion unit to control the data fusion process. After obtaining four after-activation vectors $h_t^{(l)}, h_r^{(l)}, h_{re}^{(l)}, h_{it}^{(l)}$ from the top layers of the corresponding encoders, we design a feature fusion unit to fuse the information from other views for each view. Here we still take the trust network embedding view $h_t^{(l)}$ as an example, the detail fusion steps are as follows:

Step 1: At first, fuse the vectors from other views into a vector by the following equation.

$$h_{fuse}^{(l)} = tanh(W_r h_r^{(l)} + W_{re} h_{re}^{(l)} + W_{it} h_{it}^{(l)}) \tag{11}$$

where W_r, W_{re}, and W_{it} are trainable weights, $h_{fuse}^{(l)}$ is the weighted combination of the other three user attribute views.

Step 2: Then, calculate how much of the data from the other views will be fused with the current view:

$$f_t = sigmod(V_r h_r^{(l)} + V_{re} h_{re}^{(l)} + V_{it} h_{it}^{(l)}) \tag{12}$$

where V_r, V_{re}, and V_{it} are trainable weights. The output of $sigmoid$ function is a value in the range of $(0, 1)$, with 0 denoting let nothing through and 1 denoting let everything through.

Step 3: Finally, fuse the features from the target view and other views:

$$y_t = (1 - f_t) h_{fuse}^{(l)} + f_t h_t^{(l)} \tag{13}$$

where y_t is the fused representation of the current trust network embedding view and the other attributes embedding views. Then, the fused representations of the other attribute views can be obtained and represented as y_r, y_{re}, y_{it}. For simplicity, We will not give the details for calculating y_r, y_{re}, y_{it} as the steps are similar to that when we calculate y_t.

3) Decoder: The decoder takes the output vector from the top layer of the encoder as its input and decodes it with l layers. Taking the trust network embedding view as an example, the l-th layer's output of the decoder is:

$$\hat{x}_t = g(W_{t_l} h_t^{l-1} + b_{t_l}) \tag{14}$$

Correspondingly, the outputs of the other decoders corresponding to other views can be obtained and denoted as $\hat{x}_r$, $\hat{x}_{re}$, $\hat{x}_{it}$. Auto-encoder aims to reconstruct the output representation of its decoder with the input of its encoder. For example, the reconstruction loss for input x_t is calculated as follows:

$$L_t = \frac{1}{2}\sum_{i=1}^{N}(\hat{x}_t - x_t)^2 \tag{15}$$

Then the reconstruction loss of the other three views denoted as L_r, L_{re}, L_{it} can be obtained in the same way. The loss function of representation learning is the sum of the reconstruction losses from all the views as:

$$L_{emb} = L_t + L_r + L_{re} + L_{it} \tag{16}$$

D. Trust Relationship Evaluation

The concatenation representations from all the views is denoted as $concat_{u_i} = (y_t \oplus y_r \oplus y_{re} \oplus y_{it})$ for each user u_i. And thus for each user pair (u_i, u_j), there is $(concat_{u_i}, concat_{u_j})$.

Algorithm 2 AtNE-Trust Training Algorithm

Input: *Iter*:the number of training iterations, set of features $(x_t, x_r, x_{re}, x_{it})$ for each pair of users.

Output: W_{t_i},W_{r_i},W_{re_i},W_{it_i}: weight matrices for deep auto-encoders; b_{t_i},b_{r_i},b_{re_i},b_{it_i}: bias vectors for deep auto-encoders. $(i = 1, \ldots, N-1)$.

```
1:  initialize weight matrices and bias vectors;
2:  generate $n$ negative instances;
3:  set full training set;
4:  for each it from 1 to Iter do
5:      for each user of the pair $(u_i, u_j)$ in training set do
6:          set $h_t^{(l)}, h_r^{(l)}, h_{re}^{(l)}, h_{it}^{(l)}$ ← use Equation (10) with
            input $(x_t, x_r, x_{re}, x_{it})$;
7:          set fused feature $y_t, y_r, y_{re}, y_{it}$ ← use Equation (13)
            with input $h_t^{(l)}, h_r^{(l)}, h_{re}^{(l)}, h_{it}^{(l)}$;
8:          set $L_t, L_r, L_{re}, L_{it}$ ← use Equation (15);
9:          set $L_{pre}$ ← use Equation (18);
10:         set $L$ ← use Equation (19);
11:         use Adma algorithm to optimize model parameters
12:     end for
13: end for
```

The representations for each pair of user are further concatenated as the input of a single-layer MLP as:

$$y_{ij}^{'} = softmax(W_{mlp} \times (concat_{u_i} \oplus concat_{u_j}) + b_{mlp}) \quad (17)$$

where $\oplus$ is the concatenation operator, W_{mlp} is the weight matrix, b_{mlp} is the bias parameter, $y_{ij}^{'}$ is the predicted probabilities of the user pair (u_i, u_j) belonging to the trusted pair or distrusted pair. We choose cross-entropy as trust prediction loss function:

$$L_{pre} = -\sum_{ij} y_{ij} log(y_{ij}^{'}) \quad (18)$$

where we use L_{pre} to represent the trust prediction loss.

E. Optimization Objective

The final optimization objective of the AtNE-Trust model is to minimize the sum of the reconstruction loss and the trust evaluation loss:

$$L = L_{pre} + \gamma L_{emb} \quad (19)$$

where L_{pre} and L_{emb} are the trust relationship evaluation loss and reconstruction loss, respectively. $\gamma \geqslant 0$ is a coefficient that controls the balance between these two parts. Reconstruction loss ensures the model to learn the multi-view high non-linear embedding. Trust evaluation loss measures the current trust evaluation and it will be propagated back to the representation learning. The mutual refinement between the trust evaluation and the representation learning ensures superior trust prediction results. We utilize *Adma* optimization algorithm [29] to minimize the loss L. Algorithm 2 implements the AtNE-Trust training process.

TABLE III
STATISTICS OF DATASETS

Dataset	Epinions	Ciao
the number of Users	7,151	4457
the number of Items	21,661	10,957
the number of Ratings/Reviews	371,263	237,285
the number of Trust Relationships	125,008	24,731

V. Experiments and Analysis

We carry out a set of experiments against two real-world datasets *Epinions* and *Ciao* to investigate the following research questions:

RQ1: How does our proposed method perform in comparison with the state-of-the-art approaches?

RQ2: How do trust network embedding and user attributes embedding contribute to the trust prediction, respectively?

RQ3: How do different parameters affect the performance of our method?

A. Experimental Settings

1) Datasets for Evaluation: We evaluate our method against two widely used real-world datasets *Epinions* and *Ciao*, which are publicly accessible [3]. *Epinions* and *Ciao* are two knowledge-sharing websites. There are rating values ranging from 1 to 5, which denotes an overall preference that a user to an item. Besides, there are reviews that contain user attitude and preference in text. Also, there are items a user has rated/reviewed. In addition, these datasets contain explicit trust relationships between users as *trust* lists can be maintained on these websites. *Epinions* and *Ciao* have been used widely for trust prediction. For these two datasets, we retain the users with at least 15 rating/reviews. The statistics of the datasets are summarized in Table III.

2) Evaluation Metrics: In this work,we adopt the widely used *AUC* [30] score and *F1* [31] score as evaluation metrics. The higher the values of these metrics are, the better the prediction performs.

3) Implementation: We implement our proposed model with Tensorflow[1]. Our code is publicly available in github page[2]. For directed random walk generation, we use the parameter settings in [20]: $w = 80$, $l = 40$. c is set to 10 and s is set to 5 according to experimental results. Parameter λ is set differently for a specific dataset to gain the best performance. When training the model, the ratio of trust pairs and unobserved negative user pairs is $1:5$. For the deep auto-encoders, we set the hidden layer number d as 3 and 2 for $Epinions$ and $Ciao$, respectively. The parameters are updated based on the *Adma* optimizer algorithm with the learning rate of 0.001. We set the training batch size to 500. For baseline methods, we use the same parameter settings as suggested in their original work.

[1]https: //www.tensorflow.org
[2]https://github.com/AtNE-Trust/AtNE-Trust

TABLE IV
PERFORMANCE OF DIFFERENT TRUST PREDICTION METHODS

Methods	Epinions		Ciao	
	AUC	F1	AUC	F1
TP	0.658	0.796	0.639	0.702
MF	0.914	0.928	0.815	0.826
DeepTrust	0.916	0.937	0.813	0.830
LINE	0.794	0.897	0.765	0.791
Node2Vec	0.896	0.919	0.798	0.807
SIDE	0.919	0.936	0.824	0.831
ASNE	0.921	0.937	0.826	0.834
AtNE-Trust	**0.927**	**0.939**	**0.829**	**0.836**

B. *Effectiveness of Our Model*

To answer question **RQ1**, we compare our proposed AtNE-Trust model with some baseline approaches including both classical and state-of-the-art trust prediction methods. We also compare our method with some network embedding-based methods. The baseline methods are as follows:

- **TP**: Trust propagation evaluates trust relationships along a path between users, which is the most typical trust network structure-based approach [7].
- **MF**: Matrix factorization is a classical low-rank approximation-based approach, which performs matrix factorization on trust matrix [13].
- **DeepTrust**: DeepTrust is a deep user model of homophily effect for trust prediction. It is a deep learning-based approach that considers rich user attributes [32].
- **Node2Vec**: Node2Vec is based on the Skip-Gram model to obtain the node sequences generated by the biased random walk [18].
- **LINE**: LINE learns two embedding vectors for each node by preserving the first-order and second-order proximity of the network, respectively [19].
- **SIDE**: SIDE simultaneously considers the signs and directions of links for network embedding [20].
- **ASNE**: Attributed social network embedding (ASNE) learns representations for social actors by preserving both the structure proximity and attributes proximity. This work mainly focuses on the symmetric social relationships and user attributes are obtained from user profiles [23].

Note that ASNE adopts eight different kinds of anonymized information from the user profile for attributes embedding on friendship-based networks including user ID, status, gender, major, second major, dorm/house, high school, and class year. However, some of these user profile attributes are not included in *Epinions* and *Ciao* datasets. The dataset *Epinions* includes the user ID, location, self-description, and favorite websites. The dataset *Ciao* includes the user ID, the number of reviews, the number of trustors, and the number of trustees for each user.

As there are only positive links (e.g., trust relationships) in our *Epinions* and *Ciao* datasets, a set of unlinked user pairs is randomly selected as the negative instance set for training and testing our model. The trust prediction results of our model and baseline models are summarized in Table IV (The ratio of training size and testing size is $90\% : 10\%$). We have the following observations from table IV:

- The AtNE-Trust has a better performance compared with all the baseline methods. In particular, the AtNE-Trust can achieve better performance compared with the state-of-the-art attributed network embedding method ASNE.
- AtNE-Trust, ASNE, and DeepTrust achieve relatively better performance than the other baseline methods by considering user attributes.
- SIDE, Node2Vec, LINE, MF, and TP are the methods that have not considered user attributes. SIDE outperforms the other methods in this group due to it considers both the sign and the direction of the links. As a classical low-rank approximation-based method, MF achieves quite good performance. TP gets the worst performance in these methods due to it heavily suffers the data sparsity issue.

Our AtNE-Trust achieves better performance compared with existing methods due to: (1) it captures both trust network structures and user attributes; (2) it integrates representation learning and trust evaluation into a unified deep learning method. The mutual refinement between the representation learning and trust evaluation helps to achieve good trust prediction results; (3) it employs user attributes to alleviate the suffering of the data sparsity issue.

C. *Impact of Training Size*

To further answer research question **RQ1**, we conduct experiments with different training set sizes. The user pairs are sorted according to when they establish trust relationships. The first $x\%$ user pairs are put in the training set and the remaining $(1 - x)\%$ pairs are put in the testing set. The x value is in $\{50, 60, 70, 80, 90\}$. The results against two datasets are shown in Table V, where we have the following observations:

- In general, the performance of all the methods increases when the training data size becomes bigger. When the training size is 90%, our model achieves the best performance against both datasets.
- As the training size varies from 50% to 90%, AtNE-Trust and ASNE achieve better performance compared with the other methods. Such a result shows the advantage of the attributed network embedding-based methods. In addition, our AtNE-Trust outperforms ASNE against both *Epinions* and *Ciao* datasets.

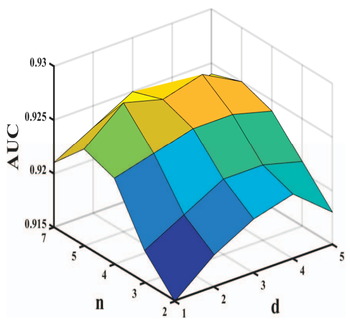
These observations show that the performance of our proposed model is not only superior but also robust with different training sizes.

D. *Impact of Trust Network Embedding and User Attributes Embedding*

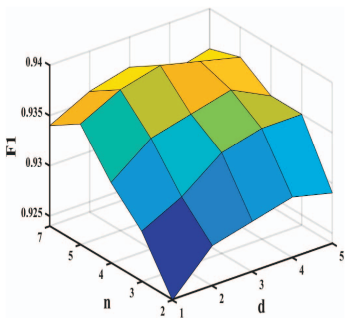
To answer research question **RQ2** about the significance of including trust network embedding and user attributes embedding, we conduct experiments on different configurations of the AtNE-Trust. For convenience, let $AtNE\text{-}Trust_{tr}$ and $AtNE\text{-}Trust_{at}$ represent the customized AtNE-Trust model

TABLE V
Area under the ROC curve (AUC) score and F1 score of different trust prediction methods with different training sizes against Epinions and Ciao. The best results are indicated in bold typeface.

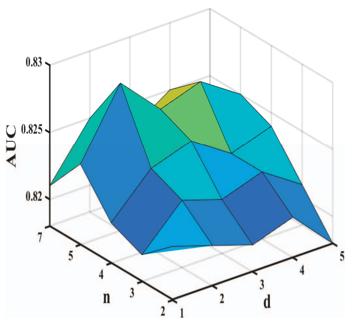
	AUC-Score (Epinions)					**F1-Score (Epinions)**					**AUC-Score (Ciao)**					**F1-Score (Ciao)**				
	50%	60%	70%	80%	90%	50%	60%	70%	80%	90%	50%	60%	70%	80%	90%	50%	60%	70%	80%	90%
TP	0.551	0.562	0.617	0.634	0.658	0.775	0.788	0.794	0.798	0.796	0.521	0.572	0.607	0.615	0.639	0.665	0.684	0.691	0.698	0.702
MF	0.901	0.906	0.911	0.912	0.914	0.912	0.914	0.923	0.926	0.928	0.742	0.803	0.810	0.814	0.815	0.802	0.803	0.809	0.817	0.826
LINE	0.754	0.768	0.769	0.772	0.794	0.846	0.851	0.872	0.893	0.897	0.700	0.709	0.734	0.741	0.765	0.712	0.723	0.754	0.767	0.791
Node2Vec	0.863	0.875	0.882	0.894	0.896	0.902	0.907	0.915	0.918	0.919	0.706	0.711	0.772	0.783	0.798	0.725	0.741	0.790	0.792	0.807
DeepTrust	0.901	0.905	0.907	0.913	0.916	0.924	0.923	0.929	0.931	0.937	0.758	0.800	0.812	0.814	0.813	0.801	0.804	0.817	0.825	0.830
SIDE	0.906	0.910	0.912	0.916	0.919	0.926	0.927	0.932	0.934	0.936	0.800	0.812	0.818	0.822	0.824	0.803	0.810	0.827	0.830	0.831
ASNE	0.910	**0.914**	0.917	0.919	0.921	**0.928**	0.930	0.934	0.936	0.937	0.801	0.818	0.819	0.822	0.826	0.805	0.814	**0.830**	0.832	0.834
AtNE-Trust	**0.913**	0.911	**0.919**	**0.924**	**0.927**	**0.928**	**0.932**	**0.936**	**0.937**	**0.939**	**0.814**	**0.819**	**0.821**	**0.823**	**0.829**	**0.809**	**0.823**	0.828	**0.834**	**0.836**



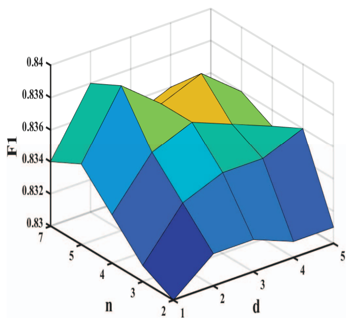
(a) AUC Score on Epinions



(b) F1 Score on Epinions



(c) AUC Score on Ciao.



(d) F1 Score on Ciao

Fig. 3. AUC and F1 scores of our proposed AtNE-Trust model with different parameters against Epinions (a, b) and Ciao (c, d).

TABLE VI
Significance of trust network embedding and user attributes embedding

Methods	Epinions		Ciao	
	AUC	F1	AUC	F1
$AtNE\text{-}Trust_{tr}$	0.908	0.920	0.812	0.819
$AtNE\text{-}Trust_{at}$	0.871	0.883	0.779	0.762
$AtNE\text{-}Trust$	**0.927**	**0.939**	**0.829**	**0.836**

only with the trust network embedding and the user attributes embedding, respectively.

As can be seen in Table VI, compared with that of AtNE-Trust, the AUC score and $F1$ score of $AtNE\text{-}Trust_{tr}$ and $AtNE\text{-}Trust_{at}$ decrease by an average of around 2% and 6%, respectively. These results show that trust network embedding contributes more than user attributes embedding for trust prediction against both *Epinions* and *Ciao* datasets. In particular, as we mentioned previously, although trust prediction is different from the normal link prediction task, still the trust network structures are quite important for the trust prediction task. Such results prove that network structures have their common but important contribution for not only the normal link prediction task but also the specific trust prediction task.

E. Parameters Sensitivity

To answer research question **RQ3**, we explore how some important parameters in our model affect the final results. The parameters investigated are as follows: γ controls the relative contributions of different losses; n is the number of negative instances for training in Algorithm 2; d is the number of hidden layers; c is the context window size of selected neighbors; s is the number of negative samples in the likelihood optimization procedure in Algorithm 1. The detailed process for investigating and analyzing the impact of these key parameters are illustrated as the following subsections.

1) Parameter γ: We investigate how the parameter γ in Equation (19) has an impact on the final results. γ controls the relative contributions of two losses and balance the whole loss function. For the training set with $x = 90$, AUC and $F1$ are calculated when γ varies in the set of $\{0.001, 0.01, 0.1, 1, 10\}$. The results of trust prediction show that the γ set to 0.1 gives the best performance.

2) Parameter n: As mentioned in Algorithm 2, for per trusted user pair, it is necessary to generate user pairs with unobserved trust relationships as negative instances for training. We conduct experiments with a different number of negative instances against the two datasets. As can be seen in Fig.3, the performance of our model increases when the number of negative instances increases from 2 to 5. When the negative instance number increases to 7, the performance begins to decrease. These results show that a proper number of the negative instances are effective for the training process while too much negative instances may introduce noises. Therefore, the optimal negative instances number is 5 in our model. Such results is also consistent with the previous research [32].

3) Parameter d: We conduct experiments to investigate the proper number of different hidden layers d. d varies from 1 to 5 in experiments as shown in Fig.3. In general, the performance of our model increases with the increase of deep neural network layers. However, there is no much further improvement in the performance when the deep neural network layer $d > 3$ and $d > 2$ for *Epinions* and *Ciao*, respectively. Therefore, the optimal depth of layers d for *Epinions* and *Ciao* are 3 and 2, respectively.

4) Parameters c and s: In Algorithm 1, there are two key parameters c and s, we conduct experiments to investigate how these two parameters affect the final results. c controls the number of selected users on the left and right context windows in a generated walk sequence. s is the selected number of negative samples in the likelihood optimization process. We vary c in the set $\{5, 10, 15, 20\}$ to investigate its impact. Results show that the performance improves at a small gradient and then becomes stable when c increases from 5 to 20. However, a larger c leads to more calculations. An acceptable trade-off is $c = 10$ against the datasets used in our paper. Similarly, s is set in the range of $\{1, 5, 10, 15, 20\}$. The performance achieves the best when s is set to 5.

VI. Conclusion

In this work, we propose an AtNE-Trust approach for trust relationship prediction in online social networks. The attributed trust network embedding is employed to capture trust network structures and user attributes simultaneously. For trust network embedding, we consider the dual roles and the connectivity properties of users. For attributes embedding, we consider users' attributes based on their ratings, reviews, and the rated/reviewed items. The obtained user embedding is fed into a representation learning module including a set of auto-encoders and a feature fusion unit to further learn the user representations. The trust evaluation module uses the learned embedding to generate the prediction. Representation learning and trust evaluation are optimized together, where the mutual refinement between them ensures the effectiveness of the final trust prediction results. The employment of user attributes alleviates the suffering of the data sparsity issue. Experiments on real-world datasets demonstrate the better performance of the proposed trust prediction model over both classic and state-of-the-art approaches.

Acknowledgements

This work was supported by the ARC DECRA Project (No. DE200100964), MQRSG (No. 95109718), and the Investigative Analytics Collaborative Research Project between Macquarie University and Data61 CSIRO. The work was also supported by the NSFC (No. 61872360) and the Youth Innovation Promotion Association CAS (No. 2017210).

References

[1] S. A. Golder and S. Yardi, "Structural predictors of tie formation in twitter: Transitivity and mutuality," in *SocialCom*, 2010, pp. 88–95.
[2] J. Tang and H. Liu, "Trust in social media," *Synthesis Lectures on Information Security, Privacy, & Trust*, vol. 10, no. 1, pp. 1–129, 2015.
[3] J. Tang, H. Gao, X. Hu, and H. Liu, "Exploiting homophily effect for trust prediction," in *WSDM*, 2013, pp. 53–62.
[4] J. D. Burger, J. Henderson, G. Kim, and G. Zarrella, "Discriminating gender on twitter," in *EMNLP*, 2011, pp. 1301–1309.
[5] M. Pennacchiotti and A.-m. Popescu, "A.: Democrats, republicans and starbucks afficionados," in *KDD*, 2011.
[6] D. Luo, F. Nie, H. Huang, and C. H. Ding, "Cauchy graph embedding," in *ICML*, 2011, pp. 553–560.
[7] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins, "Propagation of trust and distrust," in *WWW*, 2004, pp. 403–412.
[8] J. Golbeck, J. Hendler *et al.*, "Filmtrust: Movie recommendations using trust in web-based social networks," in *CCNC*, 2006, pp. 282–286.
[9] J. Golbeck and J. Hendler, "Inferring binary trust relationships in web-based social networks," *ACM Transactions on Internet Technology*, vol. 6, no. 4, pp. 497–529, 2006.
[10] P. Massa and P. Avesani, "Controversial users demand local trust metrics: An experimental study on epinions. com community," in *AAAI*, 2005, pp. 121–126.
[11] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in *SIGKDD*, 2008, pp. 426–434.
[12] J. Tang, H. Gao, and H. Liu, "mtrust: discerning multi-faceted trust in a connected world," in *WSDM*, 2012, pp. 93–102.
[13] S. Zhu, K. Yu, Y. Chi, and Y. Gong, "Combining content and link for classification using matrix factorization," in *SIGIR*, 2007, pp. 487–494.
[14] Y. Wang, X. Wang, J. Tang, W. Zuo, and G. Cai, "Modeling status theory in trust prediction." in *AAAI*, 2015, pp. 1875–1881.
[15] G. Beigi, J. Tang, S. Wang, and H. Liu, "Exploiting emotional information for trust/distrust prediction," in *SIAM*, 2016, pp. 81–89.
[16] X. Wang, Z. Zhang, J. Wang, P. Cui, and S. Yang, "Power-law distribution aware trust prediction." in *IJCAI*, 2018, pp. 3564–3570.
[17] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *SIGKDD*, 2014, pp. 701–710.
[18] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *SIGKDD*, 2016, pp. 855–864.
[19] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *WWW*, 2015, pp. 1067–1077.
[20] J. Kim, H. Park, J.-E. Lee, and U. Kang, "Side: Representation learning in signed directed networks," in *WWW*, 2018, pp. 509–518.
[21] X. Huang, J. Li, and X. Hu, "Accelerated attributed network embedding," in *SDM*, 2017, pp. 633–641.
[22] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Chang, "Network representation learning with rich text information," in *IJCAI*, 2015.
[23] L. Liao, X. He, H. Zhang, and T.-S. Chua, "Attributed social network embedding," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 12, pp. 2257–2270, 2018.
[24] F. Liu, S. Xue, J. Wu, C. Zhou, W. Hu, C. Paris, S. Nepal, J. Yang, and P. S. Yu, "Deep learning for community detection: Progress, challenges and opportunities," in *IJCAI*, 2020, pp. 4981–4987.
[25] S. Wang, C. Aggarwal, J. Tang, and H. Liu, "Attributed signed network embedding," in *CIKM*, 2017, pp. 137–146.
[26] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.
[27] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NeurIPS*, 2013, pp. 3111–3119.
[28] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *ICML*, 2014, pp. 1188–1196.
[29] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[30] J. A. Hanley and B. J. McNeil, "The meaning and use of the area under a receiver operating characteristic (roc) curve." *Radiology*, vol. 143, no. 1, pp. 29–36, 1982.
[31] M. R. Islam, B. A. Prakash, and N. Ramakrishnan, "Signet: Scalable embeddings for signed networks," in *PAKDD*, 2018, pp. 157–169.
[32] Q. Wang, W. Zhao, J. Yang, J. Wu, W. Hu, and Q. Xing, "Deeptrust: A deep user model of homophily effect for trust prediction," in *ICDM*, 2019, pp. 618–627.