This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2877748, IEEE Transactions on Knowledge and Data Engineering

DeepDirect: Learning Directions of Social Ties with Edge-based Network Embedding

Changping Wang, Chaokun Wang, *Member, IEEE*, Zheng Wang, Xiaojun Ye, Jeffrey Xu Yu, *Senior Member, IEEE*, and Bin Wang

Abstract—There exists a lot of research work on social ties, few of which is about the directionality of social ties. However, the directionality is actually a basic but important attribute of social ties. In this paper, we present a supervised learning problem, the tie direction learning (TDL) problem, which aims to learn the directionality function of directed social networks. Two ways are introduced to solve the TDL problem: one is based on handcrafted features and the other, named **DeepDirect**, learns the social tie representation through the topological information of the network. In **DeepDirect**, a novel network embedding approach, which directly maps the social ties to low-dimensional embedding vectors through deep learning techniques, is proposed. **DeepDirect** embeds the network considering three different aspects: preserving network topology, utilizing labeled data, and generating pseudo-labels based on observed directionality patterns. Two novel applications are proposed for the learned directionality function, i.e., direction discovery on undirected ties and direction quantification on bidirectional ties. Experiments are conducted on five different real-world data sets about these two tasks. The experimental results demonstrate our methods, especially **DeepDirect**, are effective and promising.

Index Terms—Tie Direction Learning; Network Embedding; Social Networks; Social Ties

1 Introduction

Online social networks are changing people's life, and these changes, such as making friends, coauthoring papers, recommending books and discussing anecdotes, are propagated through social ties. Each social tie is a relationship between two individuals. In different social networks, social ties have different meanings, e.g., friendship in social media, and co-author relationship in academic social networks. From the graph-theoretic viewpoint, a social tie is an edge of the graph which represents the whole social network.

Analysis of social ties can help people understand the social network better and improve the results of many tasks such as link prediction [1] and community detection [2]. Thus, this area has attracted tremendous interest from both academic and industrial communities. Researchers have studied some attributes of social ties, e.g., strength [3] [4] and sign [5] [6] [7]. However, as an important property of social ties, the directionality of social ties is rarely studied. In this paper, we focus on modeling the directionality of social ties in directed social networks for two requirements.

First, there are often bidirectional social ties, whose directionality could be quantized, in a directed social network. We argue that the two directions of a bidirectional tie are not always equal, i.e., one of the directions may be stronger than the other [8]. Thus, we could quantize the two directions for a bidirectional tie to understand more about this relationship, e.g., who is dominant in this relationship? Through quantizing all the bidirectional ties in a directed network, we could know more about it so that some mining tasks, e.g., link prediction, can get better performance on this network.

Second, there may exist undirected ties whose directions need to be predicted in a newly formed directed social network. When we create a new social network for some reasons, it is easier to find social ties between individuals than to know the direction of these ties. For example, we can build a social network containing all the relationship from different social media, e.g., Facebook, Instagram, LinkedIn, and Twitter. It is easy to discover who and who are friends through mining the social ties in different social media [9], but it may be difficult to distinguish who is the proposer in a relationship because social ties in some social media, e.g., Facebook, are undirected. Thus, besides some directed ties, there are many undirected ties in this newly formed network. We need to predict their directions to make this network complete, since knowing the proposers of undirected social ties benefits mining tasks on the network [10].

These two requirements motivate us to study how to model the directionality information in directed social networks. In this study, we propose a novel problem, which is called the *tie direction learning* (TDL) problem. The goal of TDL is to learn the *directionality function*, which is used to model the social ties, of a given directed social network. Then the directionality function can be used either for discovering the directions of undirected social ties or for quantizing the directionality of bidirectional social ties in this network.

The most related work to this paper is our previous study [10] which considers the directions of social ties are existing but hidden in undirected social networks and defines a new problem, i.e., the *tie direction inference* (TDI) problem. Through investigating the directionality patterns discovered in directed social networks, we propose four consistency patterns (the Degree Consistency Pattern, the Triad Status Consistency Pattern, the Similarity Consistency Pattern as well as the Collaborative Consistency Pattern) and a

- *Changping Wang, Chaokun Wang, Zheng Wang, Xiaojun Ye and Bin Wang are with the School of Software, Tsinghua University, Beijing 100084, P. R. China.*
- *Jeffrey Xu Yu is with the Department of Systems Engineering and Engineering Management, Chinese University of Hong Kong, Hong Kong.*
- *E-mails: wcpvincent@gmail.com, wz1900@gmail.com, {chaokun,yexj,wangbins}@tsinghua.edu.cn, yu@se.cuhk.edu.hk.*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2877748, IEEE Transactions on Knowledge and Data Engineering

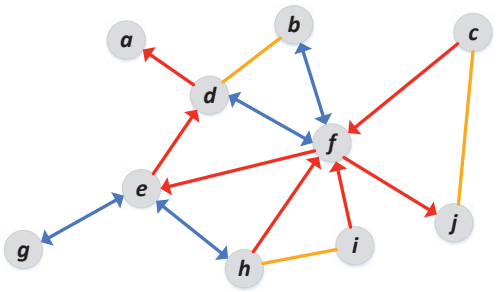


Fig. 1. A mixed social network $G = (V, E_d \bigcup E_b \bigcup E_u)$, where $V = \{a, b, c, d, e, f, g, h, i, j\}$, $E_d = \{(d, a), (c, f), (e, d), (f, e), (h, f), (i, f), (f, j)\}$, $E_b = \{(b, f), (d, f), (e, g), (e, h)\}$, and $E_u = \{(b, d), (c, j), (h, i)\}$.

framework named *ReDirect* based on these patterns. ReDirect recovers the hidden directions by finding the redirected network which accords with the four patterns as much as possible.

However, ReDirect is inappropriate to the TDL problem for a main reason: It heavily depends on the four patterns and gives them equal weights. The foundation of ReDirect is that most networks have the four patterns, but for a given network it is difficult to guarantee that there exists no other pattern and the four existing patterns are equally important. The reason for this shortcoming is that ReDirect is designed for undirected networks which do not have any directionality information. The TDL problem is defined on networks with directed ties, which means there are labeled data. Thus, it is natural to leverage the labeled data to overcome the weakness of ReDirect. We propose our own approaches in this paper, which are not restricted to the four patterns.

How to represent social ties as feature vectors is a very important part in our study to solve the TDL problem. In general, social ties contain two kinds of information: topological information and content information. Content information is usually easy to be represented as features, but it is much harder to represent topological information of social ties. This study focuses on topological information because it is very challenging. In order to fully understand the topological information, content information is not considered in this paper.

In this work, we address the challenge of representing social ties with only topological information in two different ways: one is based on hand-crafted features and the other is with the help of a novel network embedding approach. Hand-crafted features are widely adopted in the studies about social ties [5], [7]. But the embedding technique is used in this area for the first time. This paper employs the embedding technique to generate feature vectors for social ties, since embedding methods are proved effective in many other areas such as natural language processing [20] and node based network analysis [16].

Traditional network embedding approaches map the nodes to low-dimensional vectors. Thus, if we want to represent a given social tie we have to generate a representation vector according to the embedding vectors of the two endpoints of this tie. We argue this indirect way may cause the loss of topological information of social ties. In our novel network embedding approach, we directly map the social ties of a social network to low-dimensional vectors, and the embedding vectors are learned through preserving the network topology.

The main contributions of this paper are summarized as follows.

- This paper defines a novel problem, tie direction learning (TDL) problem, which aims to learn the directionality function in a given directed social network.
- Two methods are proposed in this paper to solve the TDL problem. One is based on handcrafted features and the other, named **DeepDirect**, is based on a novel edge-based network embedding method. The **DeepDirect** learns the tie embedding vectors with the consideration of three aspects, i.e., preserving network topology, utilizing labeled data, and introducing a priori knowledge.
- This paper proposes two applications for directionality function, i.e., direction discovery on undirected ties and direction quantification on bidirectional ties. Experiments are conducted on five data sets for the two applications. The results show our **DeepDirect** model could learn the directionality functions well for different social networks.

The rest of this paper is organized as follows. The definition of TDL and some related concepts are presented in Sec. 2. In Sec. 3.2, a simple but effective solution based on hand-crafted features is introduced to solve the TDL problem. In Sec. 4, we propose a model named **DeepDirect** for the TDL problem, and **DeepDirect** is based on a novel graph embedding method which maps social ties to low-dimensional vectors. Two applications of the TDL problem are shown in Sec. 5. The experimental results about our proposed methods are demonstrated in Sec. 6. Sec. 7 reviews the related work. At last, Sec. 8 concludes this paper.

2 Problem Definition

In this section, some useful concepts and the problem description of TDL are presented.

As discussed in Sec. 1, there exist directed social networks with undirected ties as well as those with bidirectional ties. Note that these two types of social ties are totally different in this paper, i.e., undirected ties are those whose directions are unknown but directions of bidirectional ties are clear. This study proposes a general network model, named *mixed social network*, which can be applied in both situations.

Definition 1 (Mixed social network)**.** *A mixed social network is denoted by a graph $G = (V, E)$, where $E = E_d \bigcup E_b \bigcup E_u$ and $E_d \bigcap E_b = E_d \bigcap E_u = E_b \bigcap E_u = \emptyset$. V is the set of individuals. $E_d \subseteq V \times V$ is the set corresponding to directed social ties ($|E_d| > 0$), $E_b \subseteq V \times V$ is the set corresponding to bidirectional social ties, and $E_u \subseteq V \times V$ is the set corresponding to undirected social ties. Each $(u, v) \in E_d \bigcup E_b$ represents a directed social tie, and each $(u, v) \in E_u$ represents an undirected social tie. $\forall (u, v) \in E_b \bigcup E_u$, we have $(v, u) \in E$, and both (u, v) and (v, u) represent the same social tie. However, $\forall (u, v) \in E_d$, $(v, u) \notin E$.*

Notice that E_d should be a non-empty set, while E_b and E_u can be empty. Fig. 1 shows an example of mixed social networks. Then the definition of the directionality function on mixed social networks is as follows.

Definition 2 (Directionality function)**.** *In a given mixed social network $G = (V, E)$, there exists a directionality function $d : E \to [0, 1]$.*

The directionality function is the universal model proposed in this paper to describe the directionality information in mixed social networks. It has different meanings for different types of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2877748, IEEE Transactions on Knowledge and Data Engineering

TABLE 1
Notations

Notation	Meaning
G	a mixed social network
V	node sets of the network
E	social tie sets of the network
$d(,)$	directionality function
$deg_{out}(u)$	the out-degree of node u
$deg_{in}(u)$	the in-degree of node u
$deg(u)$	the degree of node u
$cc(u)$	closeness centrality of node u
$bc(u)$	betweenness centrality of node u
$ee_i(u,v)$	number of the i_{th} type of triangles for tie (u,v)
x_{uv}	vector of hand-crafted features of tie (u,v)
l	number of dimensions of embedding vectors
M	embedding matrix
m_e	embedding vector for tie e
N	connection matrix
n_e	connection vector for tie e
L_{topo}	loss about network topology
L_{label}	loss about labeled data
$L_{pattern}$	loss about directionality patters
L	total loss for E-Step
w	weights of the logistic regression LR in D-Step
b	bias of the logistic regression LR in D-Step
$c(e)$	connected ties of tie e
$C(G)$	all the connected tie pairs
γ	maximum of common neighbors for $L_{pattern}$
λ	the number of negative samples
α	weight of L_{label} in L
β	weight of $L_{pattern}$ in L
y_e	label of social tie e
$\overline{y}_e$	predicted label of tie e
y_e^d	pseudo-label w.r.t. Degree Consistency Pattern
y_e^t	pseudo-label w.r.t. Triad Status Consistency Pattern
P_c	distribution for sampling a tie in begin of each iteration
P_n	distribution for the negative sampling

social ties. For a tie $e = (u,v) \in E_d$ or E_u, $d(e)$ represents the probability that this tie is from u to v. For bidirectional ties $e = (u,v), e' = (v,u) \in E_b$, $d(e)$ and $d(e')$ are the directionality weights which can quantitatively measure the two directions of this tie pair.

Definition 3 (Tie direction learning problem)**.** *Given a mixed social network G, the tie direction learning (TDL) problem is to learn the directionality function d of G.*

From the above discussion, it is easy to discover that the TDL problem is a supervised learning problem, where ties in E_d are the labeled training data.

Before presenting the rest of this paper, we summarize the notations used in this paper and show them in Table 1. For convenience, given a social tie $e = (u,v)$, the subscripts uv and e have the same meaning, e.g., $m_{uv} = m_e$.

3 Learning Based on Handcrafted Features

In this section, a simple but effective way is introduced to solve the TDL problem.

3.1 Generating Handcrafted Features

We construct handcrafted features for all the social ties with network topology. The following handcrafted features based on the statistics of the network are chosen, some of which are used in previous work about sign prediction of social ties [5], [7]:

Node degree. It is natural to use the degrees of the two nodes of a social tie as a part of its features. We slightly modify the common definitions of out-degree (denoted as deg_{out}) and in-degree (denoted as deg_{in}) for our mixed social networks:

$$deg_{out}(u) = \Big|\{v|(u,v) \in E_d \cup E_b\}\Big| + \frac{1}{2}\Big|\{v|(u,v) \in E_u\}\Big| \quad (1)$$

$$deg_{in}(u) = \Big|\{v|(v,u) \in E_d \cup E_b\}\Big| + \frac{1}{2}\Big|\{v|(v,u) \in E_u\}\Big| \quad (2)$$

The definitions on directed and bidirectional social ties remain. However, if (u,v) is an undirected social tie, it contributes '$1/2$' to both deg_{out} and deg_{in} for both nodes. Thus, for a given social tie (u,v), both the out-degrees and the in-degrees for u and v, i.e., $deg_{out}(u)$, $deg_{out}(v)$, $deg_{in}(u)$ and $deg_{in}(v)$, are employed as four degree features.

Node centrality. The closeness centrality (cc) and betweenness centrality (bc) are two widely used metrics to measure the centrality of nodes in a network. They are defined as:

$$cc(u) = \frac{1}{\sum_{v \neq u} dis_{uv}} \quad (3)$$

$$bc(u) = \sum_{i \neq u \neq j} \frac{\sigma_{ij}(u)}{\sigma_{ij}} \quad (4)$$

where dis_{uv} is the distance between nodes u and v through a shortest path, σ_{ij} denotes the number of shortest paths between nodes i and j, and $\sigma_{ij}(u)$ denotes the number of shortest paths between nodes i and j that involve node u. For convenience, the network is regarded as an undirected graph when calculating shortest paths. For a given social tie (u,v), $cc(u)$, $cc(v)$, $bc(u)$ and $bc(v)$ are employed as four centrality features.

Directed triad count. Inspired by the status theory of directed networks [34] and the triad status consistency pattern in [10], we propose the directed triad count features for the TDL problem. Similar features are used in [7], [5] and show the effectiveness. For a social tie (u,v), the directed triad count is a category of features w.r.t. the triads involving (u,v). For each common neighbor w of u and v, nodes w, u and v form a triad. Considering the directionality of (w,u) and (w,v), each of them has 4 types, and thus we can obtain $2^4 = 16$ types of triads for a tie. Therefore, for a social tie (u,v), 16 directed triad count features, denoted as $ee_i(u,v)(i = 1,2,...16)$, are employed to represent the number of those 16 types of triads formed by u, v and all their common neighbors. For instance, if there are five triads belonging to the first type, then $ee_1(u,v) = 5$. Notice that the directionality of (u,v) is not taken into consideration when generating features, since the directions of some ties are unknown.

The above three categories of features, i.e., node degree features, node centrality features and directed triad count features, are put together to form the feature vectors for all the social ties. The feature vector for a social tie e is denoted as x_e. Note that the feature vector for a tie (u,v) is different from that for (v,u) because of the directionality difference.

3.2 Modeling the Directionality Function

We model the directionality function with the logistic regression:

$$d(e) = \sigma(w \cdot x_e + b) = \frac{1}{1 + e^{-(w \cdot x_e + b)}}, \quad (5)$$

where σ is the sigmoid function, x_e indicates the feature vector, and w as well as b are the parameters of the logistic regression, which need to be learned.

For each directed social tie (u,v) in E_d, we construct two instances to train the model, one with features for (u,v) as well

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2877748, IEEE Transactions on Knowledge and Data Engineering

as label '1' and the other with features for (v, u) as well as label '0'. Thus, through training this logistic regression model, the directionality function of the given network can be learned.

4 Learning with Deep Model

Different from generating handcrafted features, an alternative way to obtain features of social ties is learning the features automatically from the raw data. A popular technique, which is called graph embedding [11], [12], aims to solve this automatic feature learning problem on graphs. A typical graph embedding method maps the graph into a low-dimensional space, usually through unsupervised learning, to obtain the embedding vector for each node. The embedding vectors are treated as the feature vectors and used in follow-up vector-based machine learning algorithms.

Recent graph embedding methods [13], [14], [15], [16], [17] all utilize deep learning techniques, because deep learning is very good at representation learning and graph embedding can be seen as the graph representation learning. However, the feature vectors for social ties are necessary to solve the TDL problem and existing graph embedding methods are all node-based, which means they map nodes rather than edges into a low-dimensional space.

There are two indirect ways to represent social ties with existing node-based graph embedding methods. The first one is to generate the vector for a social tie based through some operations (average, Hadamard product, etc.) on the two embedding vectors corresponding the two endpoints of this tie [16]. However, this approach does not define the edge-level similarity, and the edge embeddings are obtained based on the node-level similarity [18], which means it cannot capture edge features well in our TDL problem.

The second one is to convert the given graph to its line graph [19] and run node-based graph embedding methods on this line graph. The definition of the line graph is as follows: the nodes of the line graph correspond to the edges of the original graph, and there is a directed edge in the line graph from e_1 to e_2, if and only if in the original graph the target node of e_1 is the source node of e_2. From the definition, it is easy to find out that the line graph is larger than the original graph from the aspects of both nodes and edges. From the node aspect, $|V_{line}|$ equals to $|E_{original}|$, which is always much larger than $|V_{orginal}|$ (the subscripts 'line' and 'original' indicate the line graph and the original graph respectively). From the edge aspect, for a node in the original graph, if its in-degree and out-degree are d_1 and d_2 respectively, then there will be $d_1 \times d_2$ edges corresponding to this node in the line graph. Thus, handling large line graphs may take unacceptable time for node-based graph embedding methods.

Besides, existing node-based graph embedding methods are not optimized for the mixed social network in the TDL problem, i.e., they cannot make full use of the directionality information in mixed social networks.

Thus, this study proposes a model named **DeepDirect** to solve the TDL problem, and it is based on a novel edge-based graph embedding method, which is designed for the TDL problem to map the social ties in a mixed social network to low-dimensional feature vectors.

In the rest of this section, our **DeepDirect** model is presented. We first introduce the overview of **DeepDirect** in Sec. 4.1. In Sec. 4.2, Sec. 4.3 and Sec. 4.4, the details of the major part, i.e., E-Step, are discussed. Next, Sec. 4.5 shows how to train **DeepDirect**. At last, we give the complete algorithm of learning directionality function with the **DeepDirect** model and discuss its time complexity in Sec. 4.6.

4.1 The Overview of DeepDirect

Fig. 2 shows the overview of **DeepDirect**. It consists of two parts: E-Step and D-Step. In E-Step, for a given mixed social network G, we maps the social ties to low-dimensional embedding vectors which form an embedding matrix $M \in R^{|E| \times l}$ (l is the length of embedding vectors), e.g., the social tie (u, v) corresponds to the row vector m_{uv} in M. In D-Step, the directionality function d of G is learned based on M through a logistic regression model with the labeled data generated from E_d, and the logistic regression model is the same as in Sec. 3.2.

Since E-Step is the main part, some of its details are presented as follows. Before learning the embedding matrix, we need to preprocess the social ties in E_d. For each social tie $(u, v) \in E_d$, we add (v, u) to E_d and record their labels (i.e., 1 and 0, respectively) additionally to regard them as labeled data. The values in the embedding matrix M are randomly initialized and are optimized through minimizing a loss function L, which consists of three parts, i.e., L_{topo}, L_{label} and $L_{pattern}$.

Ensuring the effectiveness of the embedding matrix from three different aspects, i.e., network topology, labeled data and a priori knowledge, is the key idea of E-Step. Three loss functions (L_{topo}, L_{label} and $L_{pattern}$) are designed w.r.t. those three aspects, and the embedding matrix M is optimized through minimizing the combination of the loss functions, which is L.

The first loss function, L_{topo}, focuses on all the social ties in the mixed social network G, and it preserves the topology of G, which means the social ties that are connected (the formal definition is presented in Sec. 4.2) should be mapped close to each other in the embedding space. The basic idea of optimizing the embedding matrix M is minimizing L_{topo}. If we only consider L_{topo} and ignore L_{label} and $L_{pattern}$, the embedding process is unsupervised.

There exist labeled data in E_d, and thus they should be leveraged to help optimize the embedding matrix. To do so, a classifier is introduced to predict the label of directed social ties (ties in E_d). We jointly learn the embedding and classification, and thus the embedding vectors will be more discriminative in order to be classified correctly. The second loss function L_{label} is used to define the classification error on the labeled data. Combining L_{label} and L_{topo} could improve the discrimination of embedding vectors with the topology of G preserved.

The labeled data are not always sufficient, and there may be far more undirected social ties than the directed social ties in a mixed social network. Thus, these undirected social ties also need to be leveraged. Two patterns of directionality, which are discovered in ReDirect [10], are introduced, and pseudo-labels for undirected social ties are generated based on the two patterns. The last loss function, $L_{pattern}$, is then defined based on the classification error on the undirected ties with pseudo-labels. This loss is an effective supplement to L_{label}.

In the following three subsections, we will present these three loss functions in detail.

4.2 Preserving the topology of the network

The network topology is the most important information of our mixed social network, and thus the first goal of **DeepDirect** is to preserve the topology of a given network after it is embedded. A local structure named connected tie is defined as follows.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2877748, IEEE Transactions on Knowledge and Data Engineering

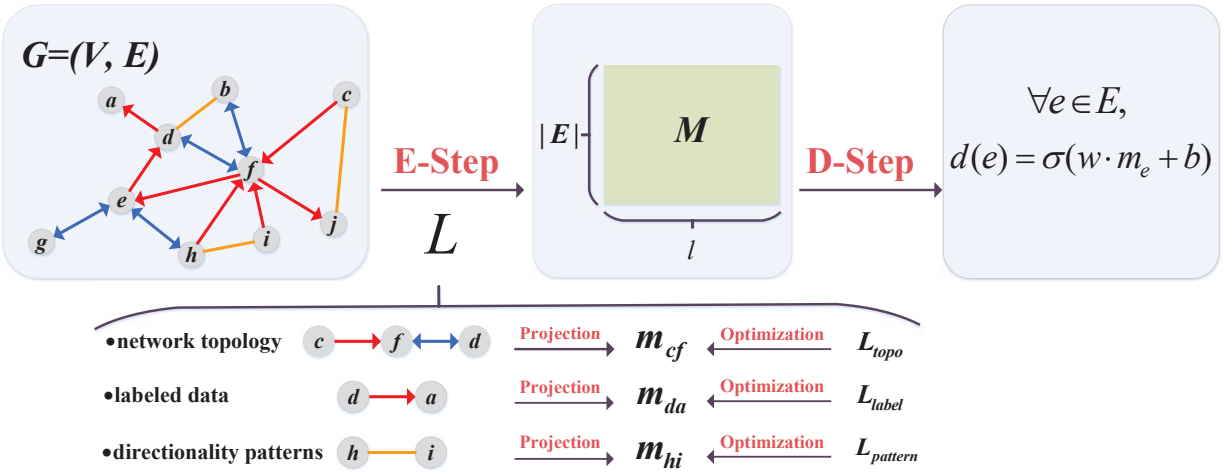


Fig. 2. The overview of **DeepDirect**. It consists of two parts: E-Step and D-Step. In E-Step, the embedding matrix M w.r.t. the given network G is obtained, and in D-Step the directionality function d of G is learned based on the embedding matrix M. The embedding is considered from three different aspects in E-Step. We preserve the network topology through the loss L_{topo}, utilize the labeled data with the loss L_{label}, and introduce a priori knowledge, i.e., directionality patterns, by $L_{pattern}$.

Definition 4 (Connected Tie)**.** *Given a mixed social network $G = (V, E)$ and two social ties $e_1 = (u_1, v_1) \in E$ and $e_2 = (u_2, v_2) \in E$, if $v_1 = u_2$ and $u_1 \neq v_2$, we call e_2 a connected tie of e_1, and we call the ordered pair (e_1, e_2) a connected tie pair. The set of all the connected ties of e_1 is denoted as $c(e_1)$. The set of all the connected tie pairs in G is denoted as $C(G)$.*

The tie degree of a social tie $e = (u, v)$ is defined as follows:

$$deg_{tie}(e) = \left|\{v' | (v, v') \in E\}\right| . \tag{6}$$

It is obvious that $deg_{tie}(e) = \left|c(e)\right|$.

The connected tie pairs could represent the topology of a mixed social network in the view of social ties.

Inspired by the idea of skip-gram model [20], a connection matrix $N \in R^{|E| \times l}$ is introduced, and each tie e corresponds to a row vector n_e, called the connection vector. Then the probability that e' is a connected tie of e according to the embedding result is defined as follows:

$$p(e'|e) = \frac{\exp(m_e \cdot n_{e'})}{\sum_{e_t \in E} \exp(m_e \cdot n_{e_t})} , \tag{7}$$

where m_e is the embedding vector of e, and $n_{e'}$ and n_{e_t} are the connection vectors of e' and e_t, respectively.

Based on Eq. 7, we define the objective function that maximizes the average log probability:

$$O = \sum_{e \in E} \sum_{e' \in c(e)} \log p(e'|e) . \tag{8}$$

The computation cost to maximize O is quite high due to the calculation of $\nabla \log p$. Thus, we use the negative sampling method [20] and replace $\log p(e'|e)$ by

$$\log \sigma(m_e \cdot n_{e'}) + \sum_{i=1}^{\lambda} E_{e_i \sim P_n(f)} \log \sigma(-m_e \cdot n_{e_i}), \tag{9}$$

where λ is the number of negative ties, and $P_n(f) \propto deg_{tie}(f)^{3/4}$ is the noise distribution which e_i is drawn from.

Thus, the first loss function is obtained as:

$$\begin{aligned} L_{topo} = -\sum_{e \in E} \sum_{e' \in c(e)} & (\log \sigma(m_e \cdot n_{e'}) \\ & + \sum_{i=1}^{\lambda} E_{e_i \sim P_n(e)} (\log \sigma(-m_e \cdot n_{e_i}))) . \end{aligned} \tag{10}$$

If the embedding matrix is optimized with L_{topo}, the network topology could be well preserved. However, the label information is only used in D-Step if L_{topo} is leveraged as the only loss in E-Step. To achieve better performance, we think of the idea to use the label information in E-Step. Thus, this study proposes the second loss function L_{label} to do so, which is presented in the next subsection.

4.3 Improving discrimination with labeled data

A semi-supervised embedding model could be more discriminative than an unsupervised one, because it is able to map the ties with same directions closer and the ones with different directions farther. Moreover, the label information can propagate among the unlabeled ties through the network topology. That is why the label information is introduced in E-Step.

The way we leverage the label information is to jointly train the embedding matrix with a logistic regression which predicts the labels of ties in E_d:

$$\overline{y}_e = \sigma(w' \cdot m_e + b') = \frac{1}{1 + e^{-(w' \cdot m_e + b')}} , \tag{11}$$

where w' and b' are the parameters of the logistic regression.

The cross-entropy is used as the prediction loss for a directed social tie:

$$J(e) = -(y_e \log \overline{y}_e + (1 - y_e) \log(1 - \overline{y}_e)) . \tag{12}$$

Thus, L_{label} is defined as follows:

$$\begin{aligned} L_{label} &= -\sum_{e \in E_d} deg_{tie}(e) J(e) \\ &= -\sum_{e \in E_d} deg_{tie}(e) [y_e \log \overline{y}_e + (1 - y_e) \log(1 - \overline{y}_e)]. \end{aligned} \tag{13}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2877748, IEEE Transactions on Knowledge and Data Engineering

It can be seen that the directed ties are assigned different weights, which are equal to their tie degrees. The social ties with higher tie degrees are connected to more social ties, and thus, if their embedding vectors are more discriminative, then more unlabeled ties could be discriminative through the effort of L_{topo}. Therefore, the tie degrees are used as the weights of labeled ties in Eq. 13 to make sure that ties with higher tie degrees are more valued.

The label information is not always sufficient, sometimes we have to face the situation that in a mixed social network there are few directed social ties but lots of undirected social ties. In the next subsection, how to leverage the undirected ties with a priori knowledge to further improve the effectiveness of the embedding matrix is discussed, and the last loss function, i.e., $L_{pattern}$, is presented.

4.4 Modeling directionality patterns for unlabeled data

The directionality information of directed social ties is represented as their label information, and is leveraged to improve the discrimination of the embedding vectors through L_{label}. However, the undirected social ties have no label information, which means they are not involved in L_{label}. In this subsection, this paper discusses how to discover the latent directionality information of undirected social ties through a priori knowledge to supplement the label information on directed social ties.

ReDirect [10] proposes four directionality patterns observed in real-world directed social networks, i.e., the **Degree Consistency Pattern**, the **Triad Status Consistency Pattern**, the **Similarity Consistency Pattern** and the **Collaborative Consistency Pattern**. The first two patterns are introduced to our embedding method, because they are simple, effective, and compatible with existing parts in E-Step.

Note that the importance of the directionality patterns is different in ReDirect and our **DeepDirect**. The four directionality patterns are the cornerstone of ReDirect model. However, in this paper they are just used to discover some additional information as the supplement to the existing topology information and label information, because it is noticed that the patterns are of different significance in different networks, which means these patterns have limitations.

The definitions of the **Degree Consistency Pattern** and the **Triad Status Consistency Pattern** are as follows:

Definition 5 (Degree Consistency Pattern)**.** *The directed ties usually link from nodes with lower degrees to those with higher degrees.*

According to this pattern, given a directed social tie (u, v) in a mixed social network, u usually has lower degree than v. Therefore, nodes with low degrees tend to be the proposer of social ties, and those with high degrees tend to be responders.

Definition 6 (Triad Status Consistency Pattern)**.** *The directed social ties usually tend to avoid loops in social networks.*

This pattern suggests that given directed social ties (u, v) and (v, w) in a network, u is more likely than w to propose the social tie between u and w.

The above two patterns are proposed based on observations from real-world social networks, and are verified through comprehensive data analysis in our previous work [10]. They are both consistent with the status theory [34], which implies that in a directed tie the proposer (source node) usually views the responder (target node) as having higher status. Since this paper is not discussing these patterns, other details are not presented here, and can be found in [10].

In this study, these two directionality patterns are used to discover latent directionality information of undirected social ties. Some pseudo-labels for undirected social ties are generated based on these two patterns.

Given a pair of undirected social ties (u, v) and (v, u) in G, the pseudo-labels for them based on the **Degree Consistency Pattern** are defined as:

$$\begin{aligned} y^d_{uv} &= \frac{deg(u)}{deg(u) + deg(v)} \\ y^d_{vu} &= \frac{deg(v)}{deg(u) + deg(v)} \end{aligned} . \tag{14}$$

Besides, pseudo-labels for them based on the **Triad Status Consistency Pattern** are also generated:

$$\begin{aligned} y^t_{uv} &= \frac{1}{|t(u,v)|} \sum_{w \in t(u,v)} \frac{\overline{y}_{uw}}{\overline{y}_{uw} + \overline{y}_{vw}} \\ y^t_{vu} &= \frac{1}{|t(u,v)|} \sum_{w \in t(u,v)} \frac{\overline{y}_{vw}}{\overline{y}_{uw} + \overline{y}_{vw}} \end{aligned} , \tag{15}$$

where $t(u, v)$ is a set with up to γ nodes randomly sampled from the set of common neighbors of u and v.

The pseudo-labels are leveraged to train the same logistic regression model in Eq.11. The cross-entropy is used as the prediction loss and the last loss function $L_{pattern}$ is defined as follows:

$$\begin{aligned} L_{pattern} = -\sum_{e \in E_u} deg_{tie}(e)[(y^t_e \log \overline{y}_e + (1 - y^t_e) \log(1 - \overline{y}_e) \\ + \mathbf{1}_{y^d_e > T}(y^d_e \log \overline{y}_e + (1 - y^d_e) \log(1 - \overline{y}_e))] , \end{aligned} \tag{16}$$

where T is a threshold and $\mathbf{1}_q$ is an indicator function defined as:

$$\mathbf{1}_q = \begin{cases} 1 & \text{if } q \text{ is true} \\ 0 & \text{if } q \text{ is false} \end{cases} . \tag{17}$$

In Eq. 16, it can be seen that only the undirected social ties whose pseudo-labels for the **Degree Consistency Pattern** are over a threshold T are taken into consideration. That is because this pattern is much more significant on social ties with higher degree difference between the two corresponding nodes. Moreover, for the same reason in L_{label}, when calculating $L_{pattern}$ on undirected social ties, the ties are given weights which equal to their tie degrees.

The details of the three loss functions in E-Step are presented so far, and then we will discuss how to optimize them in the next subsection.

4.5 Training DeepDirect

4.5.1 Learning the embedding matrix

We combine the three loss functions to obtain the total loss of the network:

$$L = L_{topo} + \alpha L_{label} + \beta L_{pattern} , \tag{18}$$

where α and β are two hyper parameters to adjust the weights of L_{label} and $L_{pattern}$.

However, L_{topo} is a summation on all connected tie pairs, while L_{label} and $L_{pattern}$ are summations on parts of ties. In

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2877748, IEEE Transactions on Knowledge and Data Engineering

order to use SGD to minimize L, L_{label} and $L_{pattern}$ are rewritten according to the equation $deg_{tie}(e) = |c(e)|$:

$$\begin{aligned} L_{label} &= -\sum_{e\in E_d}\sum_{e'\in c(e)}[y_e \log \overline{y}_e + (1-y_e)\log(1-\overline{y}_e)] \\ L_{pattern} &= -\sum_{e\in E_u}\sum_{e'\in c(e)}[(y_e^t \log \overline{y}_e + (1-y_e^t)\log(1-\overline{y}_e) \\ &\quad + \mathbf{1}_{y_e^d>T}(y_e^d \log \overline{y}_e + (1-y_e^d)\log(1-\overline{y}_e))] \ . \end{aligned} \tag{19}$$

Thus, for a sampled connected tie pair (e, e'), its loss L' is obtained as:

$$\begin{aligned} L' = &-[\log\sigma(m_e \cdot n_{e'}) + \sum_{i=1}^{\lambda} E_{e_i\sim P_n(e)}(\log\sigma(-m_e \cdot n_{e_i}))] \\ &-\mathbf{1}_{e\in E_d}\cdot\alpha[y_e \log \overline{y}_e + (1-y_e)\log(1-\overline{y}_e)] \\ &-\mathbf{1}_{e\in E_u}\cdot\beta[\mathbf{1}_{\{y_e^d>T\}}(y_e^d \log \overline{y}_e + (1-y_e^d)\log(1-\overline{y}_e)) \\ &+ (y_e^t \log \overline{y}_e + (1-y_e^t)\log(1-\overline{y}_e))] \ . \end{aligned} \tag{20}$$

We use SGD to minimize L', and the partial derivative of variables is as follows:

$$\begin{aligned} \frac{\partial L'}{\partial b'} =& \mathbf{1}_{e\in E_d}\alpha(\sigma(w' \cdot m_e + b') - y_e) \\ &+ \mathbf{1}_{e\in E_u}\mathbf{1}_{y_e^d>T}\beta(\sigma(w' \cdot m_e + b') - y_e^d) \\ &+ \mathbf{1}_{e\in E_u}\beta(\sigma(w' \cdot m_e + b') - y_e^t) \ , \end{aligned} \tag{21}$$

$$\frac{\partial L'}{\partial w'} = \frac{\partial L'}{\partial b'} m_e \ , \tag{22}$$

$$\begin{aligned} \frac{\partial L'}{\partial m_e} =& (\sigma(m_e \cdot n_{e'}) - 1)n_{e'} + \sum_{i=1}^{\lambda}(\sigma(m_e \cdot n_{e_i}))n_{e_i} \\ &+ \frac{\partial L'}{\partial b'} w \ , \end{aligned} \tag{23}$$

$$\frac{\partial L'}{\partial n_{e'}} = (\sigma(m_e \cdot n_{e'}) - 1)m_e \ , \tag{24}$$

$$\frac{\partial L'}{\partial n_{e_i}} = (\sigma(m_e \cdot n_{e_i}))m_e \ . \tag{25}$$

To save unnecessary memory space, we do not generate all the connected tie pairs, i.e., $C(G)$, but adopt a simple sampling strategy. First a tie e is drawn from E with the distribution $P_c(f)$ ($P_c(f) \propto deg_{tie}(f)$), and then a tie e' is drawn from $c(e)$ with the uniform distribution. Thus, the sampled tie pair (e, e') is obtained for training with SGD.

4.5.2 Learning the directionality function

As discussed in Sec. 4.1, the directionality function is modeled with a logistic regression LR.

$$d(e) = \sigma(w \cdot m_e + b) \tag{26}$$

After E-Step, the embedding matrix M is obtained. Then the embedding vectors corresponding to directed social ties (ties in E_d) and labels of them are used to form the training data for LR. To initialize the parameters of LR, we set w and b equal to w' and b', respectively. At last, LR is trained on those training data with the L_2 regularization.

Algorithm 1 Learning the directionality function with DeepDirect

Require:
A mixed social network $G = (V, E)$, where $E = E_d \bigcup E_b \bigcup E_u$, the dimension number l, the iteration number τ, and other hyper parameters α, β, λ, γ.
Ensure:
The embedding matrix M, the parameters of LR, i.e., w and b.
1: /*Preprocessing social ties*/
2: **for all** $(u, v) \in E_d$ **do**
3: $E_d = E_d \bigcup \{(v, u)\}$
4: $y_{uv} = 1$, $y_{vu} = 0$
5: **end for**
6: **for all** $(u, v) \in E_u$ **do**
7: Calculate y_{uv}^d with Eq. 14.
8: Generate $t(u, v)$ through randomly selecting at most γ common neighbors of u and v.
9: **end for**
10: /*E-Step*/
11: Initialize M, N, w' and b'.
12: **repeat**
13: Sample a tie e from E according to P_c, and randomly select a tie e' which is connected to e.
14: Sample λ negative ties from E according to P_n.
15: Update the embedding matrix M with Eq. 23.
16: Update the connection matrix N with Eq. 24 and Eq. 25.
17: Update w' and b' with Eq. 22 and Eq. 21.
18: **until** Reach $\tau|C(G)|$ iterations
19: /*D-Step*/
20: $w = w'$, $b = b'$
21: Learning w and b through training the logistic regression model LR.
22: **return** M, w and b.

4.6 Algorithm analysis

In this subsection, the complete algorithm and the analysis of its time complexity are presented.

Algorithm 1 shows the details of the algorithm which learns the directionality function with **DeepDirect**.

The input of the algorithm is the given mixed social network G, and all the hyper parameters. The output of this algorithm is the embedding matrix M and the learned parameters w as well as b of LR. Thus, for a given tie e, the value of the directionality function $d(e)$ can be calculated with Eq. 26.

In the algorithm, we first preprocess parts of the social ties in the network. For each directed social tie $(u, v) \in E_d$, a tie (v, u) is added into E_d, and their labels are recorded (Lines 2 – 5). For each undirected social tie $(u, v) \in E_u$, we calculate one of its pseudo-labels y_{uv}^d and generate the set $t(u, v)$ for it, which consists of up to γ randomly selected common neighbors of u and v (Lines 6 – 9).

Next, the embedding matrix is learned. The embedding matrix M, the connection matrix N, and the parameters w' and b' are initialized (Line 11). We repeat $\tau|C(G)|$ iterations, where $|C(G)|$ is the number of all the connected tie pairs. In each iteration, we sample a connected tie pair (e, e') with the strategy proposed in Sec. 4.5, draw λ negative ties for (e, e'), and update M, N, w' and b' with the update rules in Eqs. 21 –25 (Lines 13 – 17).

Then, D-Step is executed as discussed in Sec. 4.5.2. Parameters w and b are initialized with the values of w' and b', respectively (Line 20). After that, LR is trained with labeled ties (Line 21).

At last, the embedding matrix M and the parameters w as well as b are returned (Line 22).

Now we discuss the time complexity of Algorithm 1. The time cost of this algorithm consists of three parts, i.e., the time cost of preprocessing, the time cost of E-Step and that of D-Step. Thus, we have

$$T = T_{pre} + T_E + T_D. \tag{27}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2877748, IEEE Transactions on Knowledge and Data Engineering

Obviously, the preprocessing step costs $O(|E_d| + \gamma|E_u|)$ time. The time cost of E-Step can be seen as $t \times \tau C(G)$, where t is the time cost of one iteration. It can be found that $t = O(\lambda \times l)$. Moreover, because the social networks are sparse, we have $C(G) = C \times |E|$, where C is a constant number. Thus, $T_E = O(\lambda \times l \times C \times |E|) = O(|E|)$. As for T_D, it is known that the time cost of training the logistic regression is $O(|E_d|)$.

Finally, the time complexity of Algorithm 1 is $O(|E|)$, which means the runtime of our algorithm should be linear to the number of social ties in a given mixed social network.

5 Applications

In this section, two applications respectively corresponding to the two real requirements discussed in Sec. 1 are discussed.

5.1 Direction discovery on undirected ties

The first application is to discover the directions of undirected ties in the set E_u with the directionality function. For an undirected social tie (u, v), its direction can be predicted by comparing the two directionality function values w.r.t. it:

$$\text{Direction of } (u,v): \begin{cases} u \to v & \text{if } d(u,v) \geq d(v,u) \\ v \to u & \text{if } d(v,u) > d(u,v) \end{cases}. \tag{28}$$

5.2 Direction quantification on bidirectional ties

Another application of the directionality function is to quantize the directionality of bidirectional ties. For a bidirectional social tie, we argue that the two directions of this tie are not always equal, i.e., one of the directions may be stronger. Thus, how to quantize these two directions of a bidirectional tie is worth studying. Our directionality function can help with the direction quantification on bidirectional social ties.

The adjacency matrix is a widely used tool to represent the topology of a network. For a bidirectional tie (u, v), it corresponds to two cells A_{uv} and A_{vu} (supposing the adjacency matrix is A) with value '1'. We think that $d(u, v)$ and $d(v, u)$ are better than the value of '1' to quantitatively describe this social tie, if the directionality function d has been learned. Thus, through replacing the values of the cells corresponding to bidirectional ties with the directionality function values, we obtain a new matrix, called the directionality adjacency matrix. The directionality adjacency matrix is the result of direction quantification on bidirectional ties, and it can help with a lot of tasks based on the adjacency matrix, e.g., link prediction in directed social networks.

6 Experiments

In this section, the experimental results conducted on five real-world data sets are reported. Our methods are compared with some baselines in learning the directionality function. However, it is hard to judge whether a learned directionality function is good. Thus, the performance of the two tasks introduced in Sec. 5, which are based on the directionality function, is measured to infer the effect of the learned directionality functions. Sec. 6.1, shows our experimental settings, including the data sets and the baseline methods. In Sec. 6.2, experiments about the task of direction discovery on undirected ties are conducted. We compare our methods with baselines to show that **DeepDirect** outperforms all the other methods, find out the effectiveness of the supervision and the directionality patterns in the **DeepDirect** model, and study the parameter sensitivity of **DeepDirect**. Sec. 6.3 shows how to leverage the directionality function to improve the analysis tasks on bidirectional social ties. At last, the scalability of **DeepDirect** is studied in Sec. 6.4.

TABLE 2
Data sets

Data sets	Nodes	Ties
Twitter	65,044	526,296
LiveJournal	80,000	1,894,724
Epinions	75,879	508,837
Slashdot	77,360	905,468
Tencent	75,000	705,864

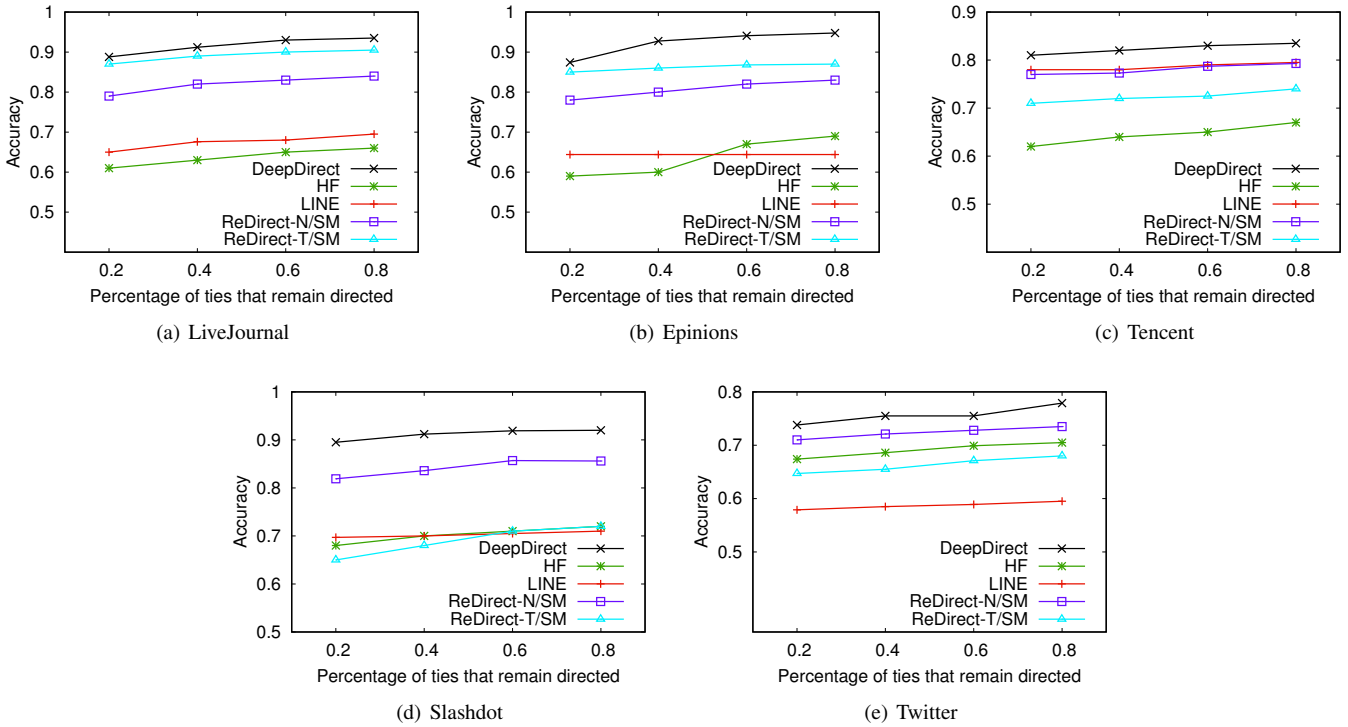
6.1 Experimental Settings

In our experiments, data sets are collected from five real social networks: **Twitter**, **LiveJournal**, **Epinions**, **Slashdot** and **Tencent** [10]. Since these networks are all large and sparse, we sample sub-networks from them through breadth-first traversal, and let each subnetwork contain $65,000 - 80,000$ nodes, which follows the preprocessing of data sets in [10]. The details of these subnetworks are shown in Table 2.

The following methods are employed to learn the directionality function.

- **LINE** [17]: It is a state-of-the-art graph embedding method, which maps the nodes of a network to low-dimensional vectors considering both the first-order and the second-order proximities of nodes. We first generate vectors for each individual with **LINE**. Then, for each social tie, the two vectors corresponding to the source node and the target node are concatenated as its feature vector.
- **HF**: It employs the handcrafted features proposed in Sec. 3, i.e., node degree features, node centrality features and directed triad count features, to form feature vectors for social ties.
- **DeepDirect**: It is the directionality learning model with edge-based network embedding we proposed in Sec. 4, which maps all the social ties into a low-dimensional space considering the topology information of network, the labeled data and the directionality patterns.
- **ReDirect-N/sm** [10]: This is the semi-supervised variant of **ReDirect-N** [10], which can make use of labeled data. It represents each node i with two latent vectors, denoted as h_i and h'_i, and takes the inner product of h_i and h'_j as the directionality value of the social tie (i, j). **ReDirect-N/sm** leverages four directionality patterns to propagate the label information across the whole network to learn the latent vectors for each nodes.
- **ReDirect-T/sm** [10]: Similar to **ReDirect-N/sm**, this is the semi supervised variant of **ReDirect-T**, which is tie-centroid and different from the node-centroid method **ReDirect-N**. **ReDirect-T/sm** starts with known directionality values of the labeled data as well as randomly initialized directionality values of other social ties. It updates the directionality value for each unlabeled social tie through the directionality values of its neighbors based on the four directionality patterns in iterations. When it converges, the directionality value for each social tie is learned.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2877748, IEEE Transactions on Knowledge and Data Engineering



(a) LiveJournal (b) Epinions (c) Tencent

(d) Slashdot (e) Twitter

Fig. 3. Accuracy of direction discovery on five data sets. **DeepDirect** outperforms all the other methods on all the data sets no matter how many directed social ties are employed.

The first three methods, i.e., a baseline method **LINE** and our proposed methods **HF** and **DeepDirect**, generate feature vectors for social ties in different ways and learn the directionality with the logistic regression model based on the features. As for the last two methods, i.e., **Redirect-N/sm** and **Redirect-T/sm**, they do not learn the directionality function of the given network, but directly learn the directionality value of each social ties in the network.

For our proposed **DeepDirect**, we set the negative sample number $\lambda = 5$, the iteration number $\tau = 10$ and the dimension number $l = 128$. As for the hyper parameters α and β, which balances the effect of the three loss functions in E-Step, we use the grid search with cross-validation to determine the optimal values. For **LINE**, we set the dimension number $l = 64$, which is the half of that in **DeepDirect**, since we need to concatenate the embedding vectors of source/target nodes to present an edge. For **ReDirect-N/sm**, the dimension number Z is set to 40, which is the optimal considering the trade-off between time cost and performance. Other parameters of baselines are all set following the corresponding papers.

6.2 Direction Discovery on Undirected Ties

In this subsection, plenty of experiments about direction discovery on undirected ties are conducted.

In our data sets, there are directed social ties and bidirectional ties, but no undirected ones. Therefore, we hide the directions of a part of directed social ties randomly to generate mixed social networks. These social ties are regarded as undirected ties in E_u, and the remaining directed social ties form the set E_d. The five directionality function learning methods introduced above are employed to learn the directionality functions for networks, and the learned functions are used to predict the directions of undirected ties in E_u through the approach mentioned in Sec. 5.1. We measure the performance of a method based on the prediction accuracy, i.e., the fraction of ties in the entire set E_u whose directions are predicted correctly.

First, our methods are compared with all the baselines in Sec. 6.2.1, and the results show that **DeepDirect** outperforms all the other methods. Next, for the **DeepDirect** model, the effectiveness of the supervision on labeled data is discussed in Sec. 6.2.2, and the effectiveness of directionality patterns is studied in Sec. 6.2.3. Then, some experiments shows the parameter sensitivity of **DeepDirect** in Sec. 6.2.4. At last, the visualization of embedding results of **DeepDirect** compared with **LINE** is demonstrated in Sec. 6.2.5, which shows that embedding vectors generated by **DeepDirect** are more discriminative for directionality of social ties.

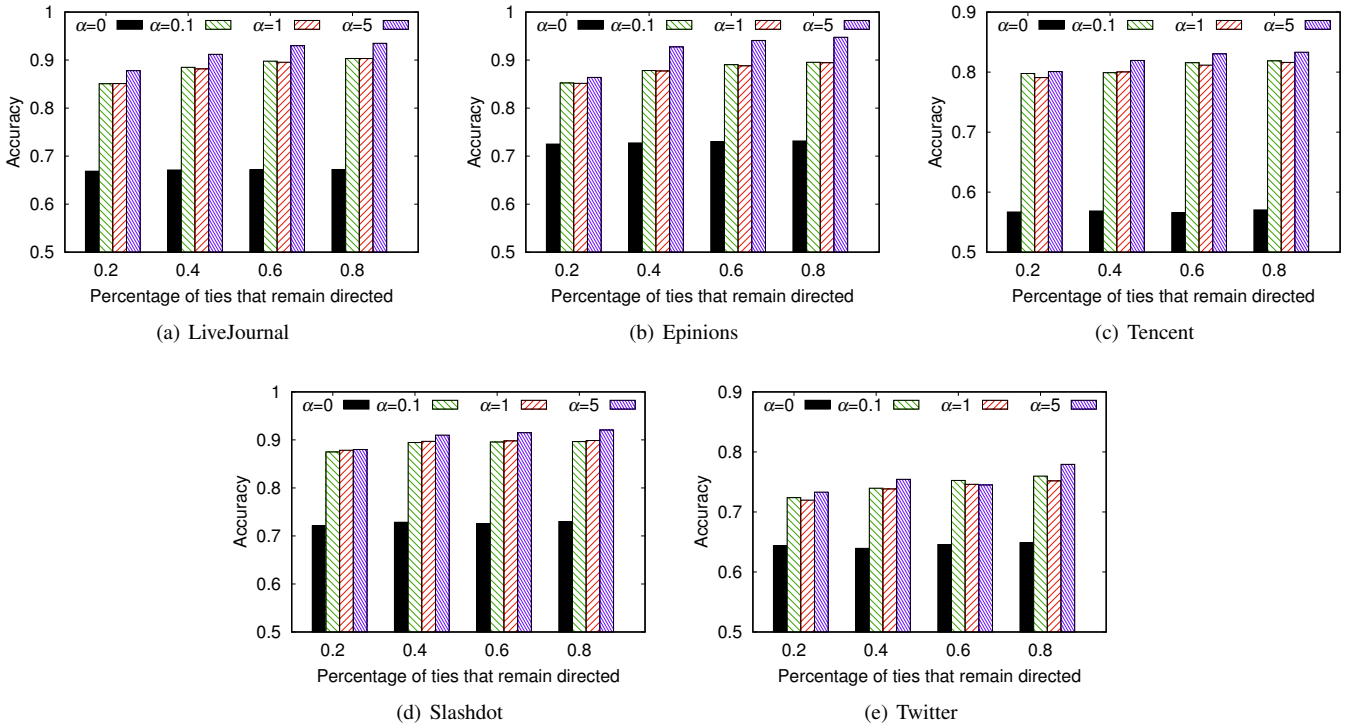
6.2.1 *Comparisons among all the methods.*

In this part, we aim to find out the effect of different directionality function learning algorithms on the task of direction discovery. The experiments are conducted on all the data sets with different percentages of ties that remain directed, i.e., $|E_d|/(|E_d| + |E_u|)$.

Fig. 3 demonstrates the experimental results. It can be seen that **DeepDirect** outperforms all the other methods on all the data sets no matter how many directed social ties are employed. This means **DeepDirect** learns the directionality function well, and the learned directionality function can be used to discover the directions of undirected ties effectively. **Redirect-N/sm** and **Redirect-T/sm** are in the second tier in the experiments. **LINE** and **HF** are the worst on almost all the data sets among all the methods.

HF cannot achieve good performance because of the way it generates feature vectors. It extracts features for each social tie through calculating some statistical values of this tie, such as degrees and centralities. However, this approach ignores the con-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2877748, IEEE Transactions on Knowledge and Data Engineering



(a) LiveJournal (b) Epinions (c) Tencent

(d) Slashdot (e) Twitter

Fig. 4. Effectiveness of labeled data in E-Step. The results show that introducing the labeled data and considering the prediction loss on them is really effective for the E-Step of **DeepDirect**.

nection between social ties, which means the topology information of the network is not well utilized. The feature vectors extracted by **HF** contain little topology information, and thus **HF** cannot learn a good directionality function of a given network.

LINE embeds the networks into low-dimensional spaces with topology information preserved, which is similar to the E-Step of **DeepDirect**, but it has worse performance in this experiment compared with **DeepDirect**. The difference between them is that LINE is node-based while **DeepDirect** is edge-based. **LINE** learns the embedding vectors for all the nodes through preserving the first-order and the second-order proximities among nodes. Thus, given a social tie (u, v), we need to concatenate the embedding vectors corresponding to u and v to form the embedding vector for (u, v), since there are no other proper approaches. Under this approach, the embedding features of ties (u, v) and (u, w) would have the same first half, and that is how **LINE** models the relation of connected social tie pairs in an indirect way. However, the approach we proposed in Sec. 4.2, which is based on the conditional probability that a tie is the connected tie of another tie, models this relation better and more directly. Moreover, besides the topology information, the E-Step of **DeepDirect** also utilizes the label information and the directionality patterns to learn the edge embedding vectors. Therefore, **DeepDirect** achieves much better performance than **LINE** in experiments.

As for **ReDirect-N/sm** and **ReDirect-T/sm**, we can see that the relationship of their performance is not stable on different data sets, which is pointed out in [10]. Specifically, on LiveJournal and Epinions, **ReDirect-T/sm** outperforms **ReDirect-N/sm**, while on the other three data sets, **Redirect-N/sm** performs better. However, **DeepDirect** always outperforms both of them on all the data sets. They are both the semi-supervised variant of a unsupervised method, and thus, they are based on the four directionality patterns obtained from observation, and the label information is only considered as the supplement. Thus, if a social network is not very consistent with those patterns, the performance of **ReDirect-N/sm** and **ReDirect-T/sm** will be poor. Unlike them, **DeepDirect** is designed for the supervised task, i.e., TDL, and the directionality patterns are only supplements when there are few labeled data. Even the network is not consistent with the patterns, **DeepDirect** still could learn the directionality function well through embedding the network with topology information and label information. That is why **DeepDirect** always outperforms **ReDirect-N/sm** and **ReDirect-T/sm**.

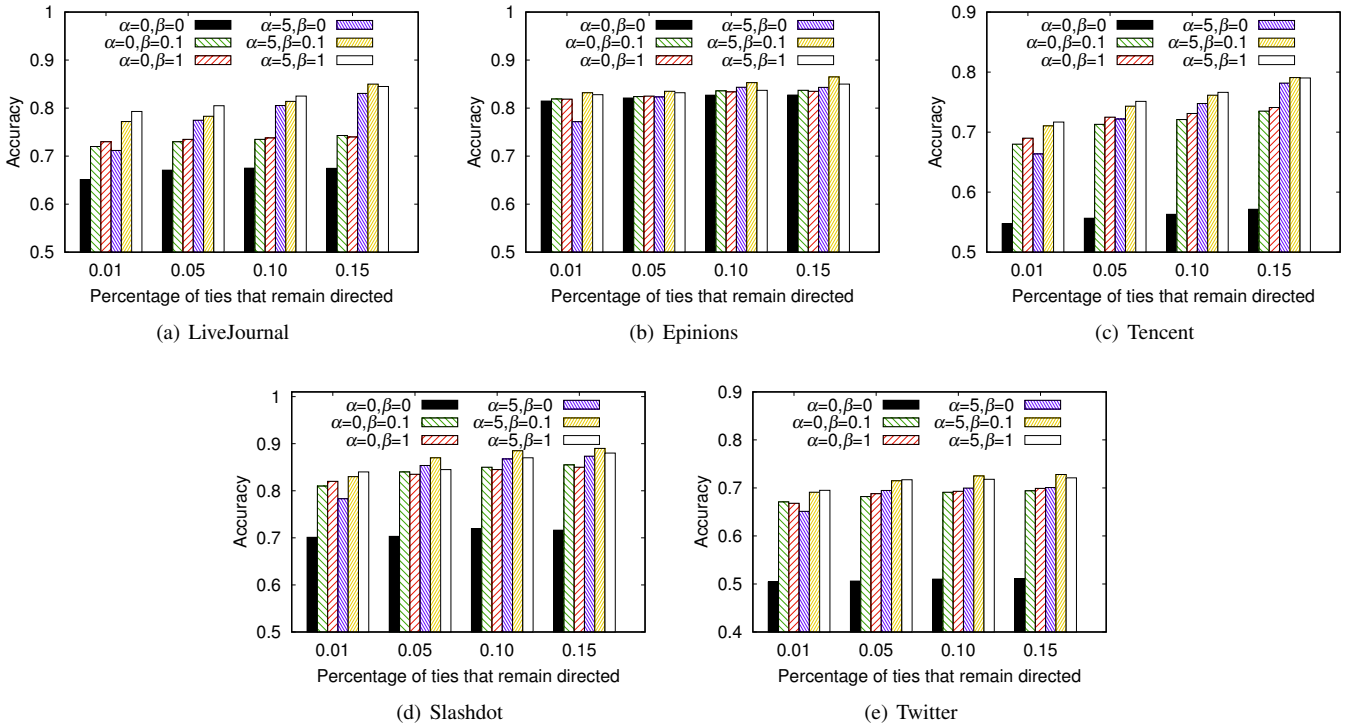
The success of **DeepDirect** means the embedding result of the E-Step is effective. Therefore, the effectiveness of the loss functions used in the E-Step and the parameter sensitivity will be studied in the following.

6.2.2 *Effectiveness of labeled data in E-Step*

First, the effectiveness of the loss function L_{label} is discussed in this part. As we know, L_{label} represents the loss of predicting the directions on directed ties and the parameter α adjusts the contribution of L_{label} to the total loss L (Eq. 18). In this experiment, we set $\beta = 0$ to remove the effect of $L_{pattern}$, and adjust the value of α to test how it affects the performance of **DeepDirect**. This experiment is conducted on the five data sets with different percentage of ties that remain directed, i.e., labeled ties.

The result is shown in Fig. 4. It is obvious that the performance of **DeepDirect** when $\alpha > 0$ is always better than that when $\alpha = 0$, e.g., the accuracy on Tencent is less than 0.6 when $\alpha = 0$ and it significantly grows to about 0.8 when we set $\alpha = 0.1$. This means introducing the labeled data and considering the prediction loss on them is really effective for the E-Step of **DeepDirect**. Besides, it can be found that in most cases 5 seems the optimal

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2877748, IEEE Transactions on Knowledge and Data Engineering



(a) LiveJournal (b) Epinions (c) Tencent

(d) Slashdot (e) Twitter

Fig. 5. Effectiveness of directionality patterns in E-Step. The results demonstrate that utilizing directionality patterns when there are few labeled data is effective to improve the performance of **DeepDirect**

value for α compared with 1 and 0.1, which suggests a higher weight for L_{label} may bring the increase of the performance of **DeepDirect**. However, it is worth noting that the parameter α should be carefully increased, because the partial derivative of w' b' and m_e will all grow with α according to Eqs. 21 – 23. If we assign α a large value, we will have to employ a small learning rate to prevent the potential loss explosion, which means a long convergence time.

6.2.3 Effectiveness of directionality patterns in E-Step

Next, the effectiveness of the directionality patterns, i.e., $L_{pattern}$ in E-Step, is demonstrated through experimental results. Because the directionality patterns are utilized as the supplement when there are few labeled data, i.e., directed social ties, in the network, we set the percentage of ties that remain directed under 15% in this experiment for all the data sets. We set six groups of values for α and β in this experiment. The first three groups ($\alpha = 0$ $\beta = 0$, $\alpha = 0$ $\beta = 0.1$, and $\alpha = 0$ $\beta = 1$) are used to demonstrate the effectiveness of directionality patterns when L_{label} is not used. The last three groups ($\alpha = 5$ $\beta = 0$, $\alpha = 5$ $\beta = 0.1$, and $\alpha = 5$ $\beta = 1$) are used to study the effectiveness of directionality patterns with L_{label} being considered.

As shown in Fig. 5, it is obvious that introducing $L_{pattern}$ always makes the performance better in this experiment, no matter whether L_{label} is used. Moreover, the improvement caused by $L_{pattern}$ is more significant when there are fewer ties that remain directed. For all the data sets, the best performances are achieved when both L_{label} and $L_{pattern}$ contribute to E-Step, i.e. $\alpha > 0$ and $\beta > 0$.

There is an interesting fact on Epinions that introducing labeled data in E-Step (i.e., $\alpha = 5, \beta = 0$) decreases the accuracy when the percentage of ties that remain directed is 0.01. In our opinion, without L_{label} and $L_{pattern}$ **DeepDirect** can perform well in this case. But introducing L_{label} with few labeled ties causes overfitting, and thus, the performance of **DeepDirect** decreases. Then the overfitting is eliminated when we set $\beta > 0$, because some pseudo-labels are generated as supplements of real labels. Therefore, this kind of overfitting can be avoided when **DeepDirect** works correctly (both $\alpha > 0$ and $\beta > 0$).

As a conclusion, utilizing directionality patterns when there are few labeled data is effective to improve the performance of **DeepDirect**.

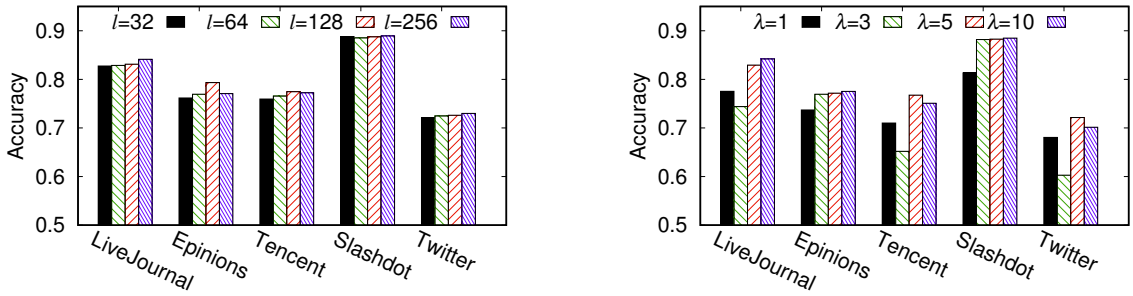
6.2.4 Parameter sensitivity

Here, the parameter sensitivity of **DeepDirect** is studied. Actually, the effect of parameters α and β are shown in Sec. 6.2.2 and Sec. 6.2.3, and thus only the number of dimensions l and the number of negative samples λ are discussed in this part. The experiment is conducted on all the five data sets with 20% directed social ties.

The results in Fig. 6(a) demonstrate that when l increases the performance gets better in most cases. However, this increment is not very significantly, and it is worth noting that the time complexity of E-Step is almost linear with l. Therefore, we should balance the performance and the time cost when choosing l, and it seems that 128 is the optimal value for l in our task through the experimental results.

Fig. 6(b) shows the variation of the accuracy of **DeepDirect** with the change of λ. It is obvious that when $\lambda = 5$ or $\lambda = 10$, **DeepDirect** achieves a better performance than that when $\lambda = 1$. However, the case when $\lambda = 3$ is more complicated. On Epinions and Slashdot, the performance when $\lambda = 3$ is almost the same as that when $\lambda = 5$ or $\lambda = 10$, while on the other three data sets, it is even worse than that when $\lambda = 1$. For a similar reason with l,

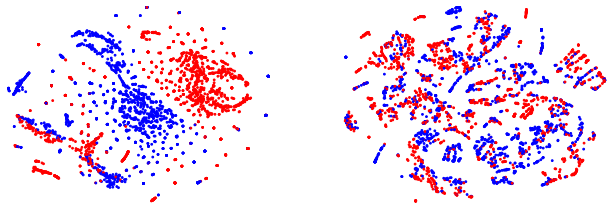
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2877748, IEEE Transactions on Knowledge and Data Engineering



(a) l, the number of dimensions of embedding vectors

(b) λ, the number of negative samples

Fig. 6. The effect of parameters l and λ.



(a) DeepDirect

(b) LINE

Fig. 7. Visualization of embedding results on a sub-network of Slashdot. The visualization result with **DeepDirect** is much more meaningful and discriminative than that with **LINE**.

5 is the optimal value for λ considering both the performance and the time cost.

6.2.5 Visualization of embedding results

In Sec. 6.2.1, we have discussed that **DeepDirect** embeds the social ties of a given network much better than **LINE**, because it directly models the connection between social ties. In this part, a network extracted from the data set Slashdot is visualized to show the difference of the embedding results of **DeepDiect** and **LINE**.

The nodes with top 1% degrees of Slashdot are selected, and the social ties among them are kept to form a new network. In this new network, we hide the directionality of 90% of the directed social ties and treat them as undirected ties, and thus we obtain a mixed social network. Both **DeepDirect** and **LINE** are employed to embed this mixed social network, and the embedding vectors for the undirected ties are obtained. Then these embedding vectors are transformed to 2D points through t-SNE [21], and these 2D points are visualized according to the actual directions of corresponding social ties. For a social tie (u, v) whose direction is hidden during embedding, the 2D point corresponding to it is marked in red if u is the actual source node, and is marked in blue if v is the actual source node.

Fig. 7 compares the visualization results of **DeepDirect** and **LINE**. It can be seen that the points of different colors in Fig. 7(a) are separable and those in Fig. 7(b) seem totally mixed, which means the visualization with **DeepDirect** is much more meaningful and discriminative than that with **LINE**. The visualization results intuitively show that the embedding vectors for social ties obtained through **DeepDirect** are more appropriate for our direction discovery task.

6.3 Direction Quantification on Bidirectional Ties

As discussed in Sec. 5.2, the direction information of bidirectional social ties can be quantitatively described with the directionality function. Through replacing the values of the cells corresponding to bidirectional ties with the directionality function values, the directionality adjacency matrix can be obtained. The directionality adjacency matrix could benefit many tasks which are based on the adjacency matrix. Here, we take a popular predictive task, link prediction, as an example to show this benefit.

The goal of link prediction is to predict whether there will be a newly formed social tie between two given individuals. A simple but effective method is based on the Jaccard Coefficient:

$$f_{Jaccard}(u \to v) = \frac{sum(A_{i,:} \cdot A_{:,j})}{sum(A_{i,:}) + sum(A_{:,j})} \tag{29}$$

where A represents the adjacency matrix. Individual pairs with higher Jaccard Coefficient are believed to have a higher chance to be friends.

For a data set G, all the individuals and 80% of social ties are extracted to form a new network G'. When conducting link prediction experiments on G' through Jaccard Coefficient, we test on all the 2-hop neighbors in G'. Specifically, those connected in G are considered as positive samples while others are regarded as negative ones.

In this experiment, three data sets are used: **LiveJournal**, **Epinions** and **Slashdot**, because over 50% social ties in them are bidirectional. We build the directionality adjacency matrices with directionality functions learned by the five algorithms, and compare them together with the original adjacency matrix in the link prediction task based on Jaccard Coefficient. The performance of link prediction is measured with Area Under the ROC Curve (AUC), and the results are shown in Fig. 8.

From the result, we can conclude that: First, the performance of link prediction is improved after we employ the directionality adjacency matrices, which means our idea that quantizing the bidirectional ties through the directionality function is effective; Second, **DeepDirect** outperforms the others in this task, just like the task in Sec. 6.2.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2877748, IEEE Transactions on Knowledge and Data Engineering

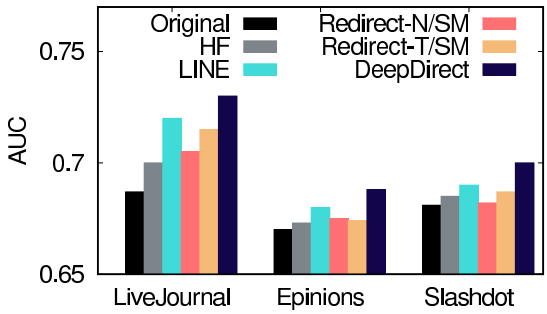


Fig. 8. AUC of link prediction on three data sets.

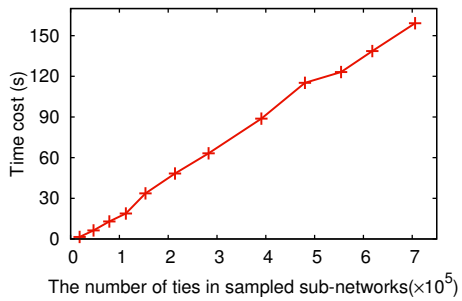


Fig. 9. Scalability of **DeepDirect**.

6.4 Scalability

At last, this subsection studies the scalability of **DeepDirect**. In Sec. 4.6, we have analyzed that the time complexity of **DeepDirect** is linear with the number of social ties in the network. In this experiment, we sample sub-networks from Tencent with different number of social ties through a BFS process, run **DeepDirect** on these sub-networks, and record the time cost.

As shown in Fig. 9, the time cost of **DeepDirect** is always linear with the number of social ties in sub-networks. Therefore, **DeepDirect** has a good scalability.

7 Related Work

In this section, we survey some lines of study that are related to our work.

Attributes of social ties. The study on attributes of social ties has attracted much attention in recent years. Social ties are classified into two categories, i.e., strong ones and weak ones in [3] and [4], while the strength of social ties is measured in a quantitative way in [22], [23] and [24]. There are also some work ([5], [6] and [7]) which study the tie sign prediction problem on signed social networks, e.g., Epinions, whose users can express trust or distrust of others. However, the directionality of social ties is studied rarely. The most related work to ours is [10], which considers that the directions of social ties are existing but hidden in undirected social networks, and proposes an unsupervised framework, ReDirect, to recover the directions in undirected networks.

Deep learning. Deep learning is a representation-learning method with multiple levels of representation [25]. The most popular deep models are the convolution neuron network (CNN) [26] and the recurrent neural network (RNN) . CNNs are proven extremely effective in processing images [27], audio, and video, while RNNs have an advantage on sequential data such as text and speech. In this paper, our embedding method borrows the idea of the skip-gram model [20], a famous deep model for word embedding, to learn the representation of social ties.

Graph embedding. There exist some classical graph embedding approaches based on dimensionality reduction of Laplacian or the adjacency matrices, such as LLE [28], IsoMAP [29] and Laplacian Eigenmaps [30]. However, these methods suffer from heavy computation cost and performance drawbacks.

Almost all the recent graph embedding methods utilize deep learning techniques, because deep learning is very good at representation learning and graph embedding can be seen as the graph representation learning. There are two major categories of recent graph embedding methods based on deep learning.

The first category, e.g., SAE [13], DNR [14], and SDNE [31], uses the deep autoencoder to transform the raw vectors for nodes (which can be columns of adjacency matrix, Laplacian matrix, modularity matrix, etc.) to low-dimensional embedding vectors non-linearly, and the model is optimized through minimizing the reconstruction error w.r.t. embedding vectors.

The second category of graph embedding methods, such as DeepWalk [15], Node2vec [16], LINE [17] and GraRep [32], borrows the idea of word2vec [20], which is a famous deep learning method to solve the word embedding problem in natural language processing area. It considers the neighbor relations between nodes as the co-occurrence relations between words, defines a conditional probability (similar to word2vec) to model the neighbor relations, and estimates the embedding vectors through maximizing the log-likelihood of the whole graph.

Besides, Xu *et al*. [33] focus on the coupled heterogeneous network consisting of two different but related sub-networks connected by some inter-network edges. They propose a method named embedding of embedding (EOE) for jointly embedding the coupled network.

However, all the existing graph embedding models are node-based, which means they map the nodes to low-dimensional vectors. Our proposed model, **DeepDirect**, learns the edge representations.

8 Conclusion and Future Work

This paper studies the directionality information of social ties in directed networks. Firstly, a novel problem named TDL is defined, which aims to learn the directionality function of a social network. Next, two methods are proposed to solve the TDL problem. The first is a simple but effective method which learns the directionality function based on handcrafted features. In the second method, named **DeepDirect**, we first learn the embedding of the network, and then learn the directionality functions based on the embedding vectors. Then, we introduce how to apply the directionality function in two real applications, which are direction discovery on undirected ties and direction quantification on bidirectional ones. Finally, comprehensive experiments are conducted to evaluate the effect of our methods through the performance of the two applications. The experimental results show that our methods are effective and promising.

This paper does not only propose an effective method for social tie directionality learning, but also shows a general way for analysis tasks on social ties. For a prediction task on social ties, we can first learn the edge-based embedding of the network, and then use off-the-shelf vector-based machine learning methods to solve the problem. To learn the the edge-based embedding of the network, it is effective to combine the network topological

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2877748, IEEE Transactions on Knowledge and Data Engineering

information, labeled data, and a priori knowledge, just like what we do in **DeepDirect**.

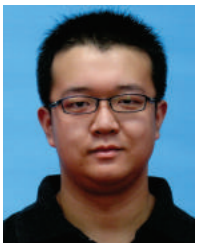
There are many directions of the future work. Among them is to leverage transfer learning to improve the performance on networks with few labeled data. Also, we can try to use a deep neural network in D-Step to learn a non-linear directionality function. Moreover, since now the undirected ties are regarded as directed ties with hidden direction, we can study the possibility that an undirected tie is actually bidirectional and analyze its directionality of two directions.

ACKNOWLEDGMENTS

This work is supported in part by the National Key Research and Development Program of China (No. 2017YFC0820402), the Intelligent Manufacturing Comprehensive Standardization and New Pattern Application Project of Ministry of Industry and Information Technology (Experimental validation of key technical standards for trusted services in industrial Internet), and the China National Arts Fund (No. 20164129).

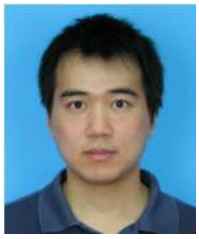
REFERENCES

[1] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," in *CIKM*, 2003, pp. 556–559.
[2] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3-5, pp. 75–174, 2010.
[3] E. Gilbert and K. Karahalios, "Predicting tie strength with social media," in *CHI*, 2009, pp. 211–220.
[4] I. Kahanda and J. Neville, "Using transactional information to predict link strength in online social networks." *ICWSM*, vol. 9, pp. 74–81, 2009.
[5] J. Leskovec, D. Huttenlocher, and J. Kleinberg, "Predicting positive and negative links in online social networks," in *WWW*, 2010, pp. 641–650.
[6] S.-H. Yang, A. J. Smola, B. Long, H. Zha, and Y. Chang, "Friend or frenemy?: predicting signed ties in social networks," in *SIGIR*, 2012, pp. 555–564.
[7] J. Ye, H. Cheng, Z. Zhu, and M. Chen, "Predicting positive and negative links in signed social networks by transfer learning," in *WWW*, 2013, pp. 1477–1488.
[8] W. Denis, V. Heidi, E. Steven and W. Michel, "The Strong, the Weak, and the Unbalanced: The Link Between Tie Strength and Cyberaggression on a Social Network Site," *Social Science Computer Review*, vol. 33, no. 3, pp. 315-342, 2015.
[9] J. Zhang, X. Kong and P. S. Yu, "Predicting Social Links for New Users across Aligned Heterogeneous Social Networks," in *ICDM*, 2013, pp. 1289-1294.
[10] J. Zhang, C. Wang, J. Wang, J. X. Yu, J. Chen, and C. Wang, "Inferring directions of undirected social ties," *TKDE*, vol. 28, no. 12, pp. 3276–3292, 2016.
[11] S. Yan, D. Xu, B. Zhang, H. Zhang, Q. Yang, and S. Lin, "Graph embedding and extensions: A general framework for dimensionality reduction," *TPAMI*, vol. 29, no. 1, pp. 40–51, 2007.
[12] P. Cui, X. Wang, J. Pei, and W. Zhu, "A Survey on Network Embedding," *CoRR*, Nov., 2017.
[13] F. Tian, B. Gao, Q. Cui, E. Chen, and T. Liu, "Learning deep representations for graph clustering," in *AAAI*, 2014, pp. 1293–1299.
[14] L. Yang, X. Cao, D. He, C. Wang, X. Wang, and W. Zhang, "Modularity based community detection with deep learning," in *IJCAI*, 2016, pp. 2252–2258.
[15] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: online learning of social representations," in *SIGKDD*, 2014, pp. 701–710.
[16] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *SIGKDD*, 2016, pp. 855–864.
[17] 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.
[18] H. Cai, V. W. Zheng, and K. C. Chang, "A Comprehensive Survey of Graph Embedding: Problems, Techniques and Applications, " *CoRR*, Feb. 2018.
[19] F. Harary and R. Z. Norman. "Some properties of line digraphs," *Rendiconti del Circolo Matematico di Palermo* vol. 9, no. 2, pp. 161–168, 1960.
[20] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013, pp. 3111–3119.
[21] L. V. D. Maaten and G. Hinton, "Viualizing data using t-sne," *JMLR*, vol. 9, no. 2605, pp. 2579–2605, 2008.
[22] C. A. Yeung and T. Iwata, "Strength of social influence in trust networks in product review sites," in *WSDM*, 2011, pp. 495–504.
[23] R. Xiang, J. Neville, and M. Rogati, "Modeling relationship strength in online social networks," in *WWW*, 2010, pp. 981–990.
[24] J. Zhuang, T. Mei, S. C. Hoi, X.-S. Hua, and S. Li, "Modeling social strength in social media community via kernel-based learning," in *MM*, 2011, pp. 113–122.
[25] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[26] B. B. Le Cun, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *NIPS*, 1990.
[27] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012, pp. 1097–1105.
[28] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.
[29] J. B. Tenenbaum, V. d. Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.
[30] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *NIPS*, 2001, pp. 585–591.
[31] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *SIGKDD*, 2016, pp. 1225–1234.
[32] S. Cao, W. Lu, and Q. Xu, "Grarep: Learning graph representations with global structural information," in *CIKM*, 2015, pp. 891–900.
[33] L. Xu, X. Wei, J. Cao, and P. S. Yu, "Embedding of embedding (EOE): joint embedding for coupled heterogeneous networks," in *WSDM*, 2017, pp. 741–749.
[34] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins. "Propagation of trust and distrust," in *WWW*, 2004, pp 403412.



Changping Wang received his B.S. degree in Computer Software from Tsinghua University, in 2012. He is currently a Ph.D. candidate in the School of Software, Tsinghua University. His major research interests include social network analysis, data mining and database applications.

Chaokun Wang received the B.Eng. degree in computer science and technology, the M.Sc. degree in computational mathematics and the Ph.D. degree in computer software and theory in 1997, 2000 and 2005, respectively, from Harbin Institute of Technology, China. He joined the faculty of School of Software at Tsinghua University in February 2006, where currently he is a Tenured Associate Professor. His current research interest includes social network analysis, graph data management, and music computing.



Zheng Wang received the B.E. degree in computer science and the M.E. degree in signal processing from Beijing Jiaotong University, in 2009 and 2011, respectively. He is currently working toward the Ph.D. degree in the School of Software at Tsinghua University. His research interests include social network analysis, causality and artificial intelligence.

Xiaojun Ye received the BS degree in mechanical engineering from Northwestern Polytechnical University, Xian, China, in 1987 and the Ph.D. degree in information engineering from INSA Lyon, France, in 1994. Currently, he is a professor at School of Software, Tsinghua University, Beijing, China. His research interests include cloud data management, data security and privacy, and database system testing.

Jeffrey Xu Yu received the BE, ME, and the PhD degrees in computer science, from the University of Tsukuba, Japan, in 1985, 1987, and 1990, respectively. He is currently a professor at the Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong. His major research interests include graph mining, graph database, keyword search, and query processing and optimization. He is a senior member of the IEEE, a member of the IEEE Computer Society, and a member of ACM.

Bin Wang is currently an associate professor at School of Software, Tsinghua University, China. He received his Ph.D. in Computer Science from Tsinghua University in 2005. He was a research assistant at Department of Computer Science, Hong Kong University, and had postdoctoral research training at ISA/ALICE Research Group, INRIA-LORIA, France. His current research interests include deep learning, geometry processing, computer vision, image and video processing.