Edge2vec: Edge-based Social Network Embedding

CHANGPING WANG, Tsinghua University
CHAOKUN WANG, Tsinghua University
ZHENG WANG, Tsinghua University
XIAOJUN YE, Tsinghua University
PHILIP S. YU, University of Illinois at Chicago

Graph embedding, also known as network embedding and network representation learning, is a useful technique which helps researchers analyze information networks through embedding a network into a low-dimensional space. However, existing graph embedding methods are all node-based, which means they can just directly map the nodes of a network to low-dimensional vectors while the edges could only be mapped to vectors indirectly. One important reason is the computational cost, because the number of edges is always far greater than the number of nodes. In this paper, considering an important property of social networks, i.e., the network is sparse, and hence the average degree of nodes is bounded, we propose an edge-based graph embedding (**edge2vec**) method to map the edges in social networks directly to low-dimensional vectors. **Edge2vec** takes both the local and the global structure information of edges into consideration to preserve structure information of embedded edges as much as possible. To achieve this goal, **edge2vec** first ingeniously combines the deep autoencoder and Skip-gram model through a well-designed deep neural network. The experimental results on different data sets show **edge2vec** benefits from the direct mapping in preserving the structure information of edges.

CCS Concepts: • **Information systems** → **Data mining**; • **Theory of computation** → **Social networks**; • **Computer systems organization** → **Neural networks**.

Additional Key Words and Phrases: Graph Embedding, Deep Autoencoder, Skip-gram, Edge2vec

ACM Reference Format:
Changping Wang, Chaokun Wang, Zheng Wang, Xiaojun Ye, and Philip S. Yu. 2020. Edge2vec: Edge-based Social Network Embedding. *ACM Trans. Knowl. Discov. Data.* 1, 1, Article 1 (January 2020), 24 pages. https://doi.org/10.1145/3391298

1 INTRODUCTION

With the development of information technology, more and more data are stored in information networks, e.g., online social networks, protein-protein interaction networks and citation networks. Analyzing these information networks has been attracting increasing attention from both academia and industry. Because most information networks are large and have millions of nodes, one useful

Chaokun Wang is supported in part by the National Natural Science Foundation of China (No. 61872207) and Kwai Inc. Xiaojun Ye is supported by the National Key R&D Program of China (No. 2019QY1402). Philip S. Yu is supported in part by NSF under grants (III-1526499, III-1763325, III-1909323, and CNS-1930941).
Authors' addresses: Changping Wang and Zheng Wang, School of software, Tsinghua University; emails: {wang-cp12, zheng-wang13}@mails.tsinghua.edu.cn; Chaokun Wang (corresponding author) and Xiaojun Ye, School of software, Tsinghua University; emails: {chaokun, yexj}@tsinghua.edu.cn; Philip S. Yu, Department of Computer Science, University of Illinois at Chicago; email: psyu@uic.edu.

1556-4681/2020/1-ART1 $15.00
https://doi.org/10.1145/3391298

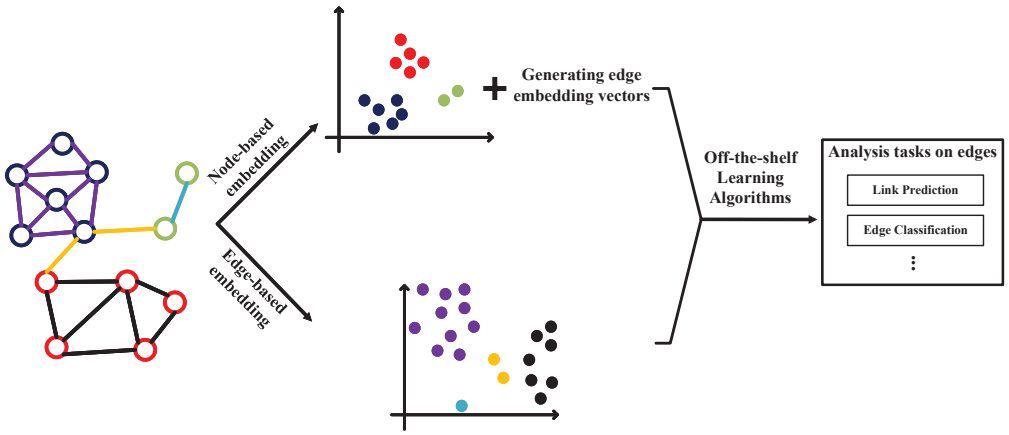


Fig. 1. The difference between node-based and edge-based graph embedding methods when dealing with analysis tasks on edges.

technique called graph embedding [2, 30], which is also known as network embedding and network representation learning, is proposed to help us analyze large information networks. A graph embedding method embeds a network into a low-dimensional space through mapping each node in the network to a low-dimensional vector. Thus, the embedded network can be processed in follow-up analysis tasks, e.g., link prediction [22] and node clustering [32], through vector-based machine learning algorithms.

However, existing graph embedding methods are all node-based, which means the edges in the network cannot be mapped to low-dimensional vectors directly. There are some indirect methods to represent edges through existing graph embedding methods. For instance, node2vec [9] uses the Hadamard product of the two vectors which denote the source node and the target node of an edge to represent the edge. We argue that representing edges in a direct approach is very important, because an edge representation vector generated from the embedding vectors of its endpoints cannot preserve the complete properties of this edge. Thus, the analysis tasks on edges, such as link prediction, cannot achieve the best performance based on these indirect approaches.

Figure 1 shows the difference between node-based and edge-based graph embedding methods when dealing with analysis tasks on edges. The node-based methods focus on the nodes and their corresponding properties, e.g. the node proximity [30]. The embedding result can be used for analysis tasks on edges through generating edge embedding vectors. However, edge-based graph embedding methods can directly embed edges in the network by focusing on the properties w.r.t. edges (e.g. the edge proximity from the aspect of structure information).

The drawback of an edge-based graph embedding method may be its computational cost, since the number of edges is far greater than that of nodes in a general network. However, in social networks, one very important category of information networks, the average degree of nodes is bounded [32]. That means the computational costs of edge-based embedding methods on these networks are acceptable. Moreover, there are a lot of studies on the attribute of edges in social networks, which are also called social ties. For example, Gilbert et. al. [8] classify the social ties to strong ones and weak ones, Ye et. al .[43] study tie sign prediction problem on signed social networks, and Zhang et. al. [46] propose a framework to recover the directions in undirected social

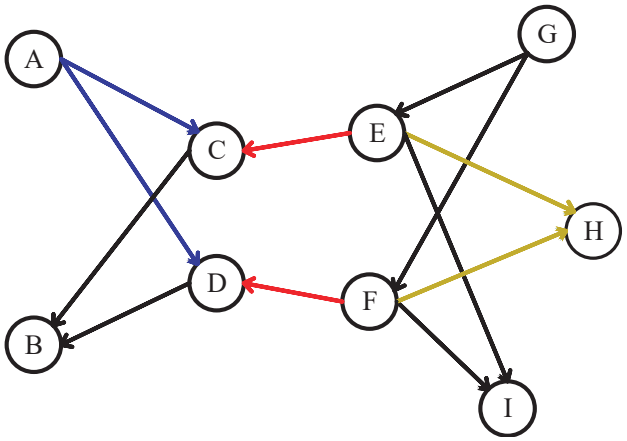


Fig. 2. Edges (A, C) and (A, D), and edges (E, H) and (F, H) both have similar local structure information. Edges (E, C) and (F, D) have similar global structure information.

networks. Thus, an edge-based graph embedding method for social networks is necessary and feasible.

In this paper, we propose the first edge-based social network embedding method, which is called **edge2vec**. Similar to existing graph embedding methods, **edge2vec** considers only the structure information of a social network, which makes it easy to apply to different social networks with various content information. **Edge2vec** is designed to preserve both the local and the global structure information of edges. As shown in Figure 2, edges (A, C) and (A, D) share the same source node A, and thus we think the two edges must be similar to each other from the aspect of local structure information. For the similar reason, edges (E, H) and (F, H) also have the similar local structures. Edges (E, C) and (F, D) share neither source node nor target node, which means their local structures are totally different. However, E and F have similar neighborhood structures, i.e. they both have four neighbors and share three of them (G, H and I), and so do C and D. Thus, edges (E, C) and (F, D) play similar roles in the network, and we think they should be similar from the aspect of global structure information of the network. To achieve the goal of preserving both the local and the global structure information, **edge2vec** first ingeniously combines the characteristics of deep autoencoder [29] and Skip-gram model [26] through a well-designed deep neural network.

It is worth noting that running a node-based network embedding method on the line graph [12] is a trivial way to obtain the embedding vectors of edges in the original graph. The line graph of a graph G is another graph, denoted as $L(G)$, that represents the adjacencies between edges of G. The nodes in $L(G)$ corresponds to the edges in G, and two edges in $L(G)$ are adjacent if and only if the corresponding edges in G are connected (in directed graphs, two edges like (u, v) and (v, w) are connected). However, it is hard to capture the global structure information of the original graph based on the line graph. For instance, as we discussed before, the edges (E, C) and (F, D) in Figure 2 have similar global structure information. But if we construct the line graph, we can find that the two nodes corresponding to (E, C) and (F, D) are neither adjacent themselves nor connected to similar neighbors, which means existing node-based embedding methods cannot consider these two nodes of line graph similar to each other. Thus, simply running the node-based embedding methods on the line graph cannot help us preserve the global structure information.

The main contributions of this paper are summarized as follows:

- We propose a novel graph embedding problem which aims to embed a social network into a low-dimensional space through mapping all the edges in the network to low-dimensional vectors. The embedding vectors apply to all the analysis tasks on the edges of social networks.
- We propose **edge2vec** to address the edge-based social network embedding problem. It employs a well-designed deep neural network with multiple deep autoencoders, to embed social networks with both the global and the local edge proximity preserved.
- We conduct experiments to compare **edge2vec** with state-of-the-art node-based graph embedding methods. The experimental results show that **edge2vec** can preserve more structure information of edges than the node-based graph embedding methods.

The rest of this paper is organized as follows. The related work is discussed in Section 2. In Section 3, we present the definition of our problem, i.e. edge-based social network embedding. We propose our edge-based embedding model, **edge2vec**, in Section 4. The experimental results about **edge2vec** are demonstrated in Section 5. At last, Section 6 concludes this paper.

2 RELATED WORK

In this section, we survey lines of related work.

2.1 Node-based graph embedding methods

There exist some classical graph embedding methods based on dimensionality reduction of Laplacian or the adjacency matrices, such as LLE [28], IsoMAP [31] and Laplacian Eigenmaps [2]. However, they suffer from heavy computational costs and performance drawbacks.

Some recent studies utilize the deep learning technique to perform dimensionality reduction. SAE [32] employs a stacked sparse autoencoder to transform the adjacency matrix into a low-dimensional space. DNR [41] also utilizes a stacked autoencoder, but it uses the modularity matrix as the input. A Siamese network consisting of two deep autoencoders is used in SDNE [34] to preserve both the first-order proximity and the second-order proximity of nodes.

Meanwhile, many graph embedding methods with the Skip-gram model, which is designed for word representation [26], are proposed recently. They treat the nodes in a network as words in a document, and adopt different approaches to capture the local "context" relationship between nodes. DeepWalk [27] employs the truncated random walks to obtain the context information of nodes. Node2vec [9] improves DeepWalk through replacing the truncated random walks with a new search strategy which combines the property of BFS and DFS. LINE [30] considers the neighborhood of a node as its context. GraRep [5] extends LINE by considering the indirect neighbors, but it suffers from the length of embedding vectors.

Also, there exist some studies based-on matrix factorization. In [40], Yang *et al.* prove that DeepWalk, which is based on the Skip-gram model, is actually equivalent to matrix factorization and propose TADW, which incorporates text features into network representation learning under the framework of matrix factorization. M-NMF [35] is a novel modularized nonnegative matrix factorization model to incorporate the community structure into network embedding.

The variational autoencoder (VAE) [15] is a popular generative model to learn the latent representations. Some recent studies [10, 16] are proposed to learn latent representations for graphs based on VAE. They leverage GCN [17] as encoders, and thus they can naturally incorporate node features.

The above are all unsupervised graph embedding methods. However, the unsupervised embedding vectors lack the ability of discrimination when applied to supervised learning tasks, such as node classification. Discriminative Deep Random Walk (DDRW) [20] and max-margin DeepWalk (MMDW) [33] are supervised models which jointly optimize the classifier and the embedding

model. They both aim to learn the embedding vectors that well capture the topological structure and meanwhile are discriminative for the node classification tasks. RSDNE [36] is also a supervised method. It approximately guarantees intra-class similarity and inter-class dissimilarity, obtaining favorable performance in both the general and zero-shot graph embedding problems.

Xu *et al.* [39] focus on the coupled heterogeneous network, which consists of two different but related sub-networks connected by inter-network edges. They propose a method named embedding of embedding (EOE) for jointly embedding the coupled network, and the experimental result shows the embedding result learned by EOE is better than that obtained through embedding the sub-networks separately with baseline embedding methods.

Recently there is a study focusing on the edge representation [1]. However, the way in which that study represents edges is far different from ours. It is actually an end-to-end link prediction model based on the node-based embedding. It learns embedding vectors for nodes and an edge function for edges. The edge function maps a node pair to a real value to present an edge. That work is still a node-based network embedding method, and the edge function it learns does not apply to tasks except link prediction. Therefore, it cannot solve our problem which is to learn the representation vectors w.r.t. edges for general propose.

All the existing graph embedding models learn the representations only for nodes. Our proposed method, **edge2vec**, first learns the edge representation vectors by ingeniously combining the characteristics of the deep autoencoder and the Skip-gram model through a well-designed deep neural network.

2.2 Analysis tasks on edges in social networks

The problem of link prediction [22] is a famous task related to edges in social networks. To solve this problem, the traditional methods utilize local neighbor structure to measure the node proximity [22]. There are also some different methods, such as random walk methods [21, 47], kernel methods [18], matrix factorization methods [7, 25], content-based methods [11, 24], and attribute-based methods [45]. In [9], Grover and Leskovec employ the graph embedding method for the link prediction task.

Furthermore, a large and popular kinds of analysis tasks focusing on the natural attributes of edges in social networks, i.e. social ties. In [8, 13], social ties are classified to strong ones and weak ones, while some other studies [38, 44, 48] measure the strength of social ties in a quantitative way. There are also some work, [19, 42, 43], which study the tie sign prediction problem on signed social networks, e.g. Epinions, whose users can express trust or distrust of others. A recent and novel study about social ties is about the directionality [46], 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.

In this paper, we choose the tasks of link prediction, social tie direction prediction and social tie sign prediction to show the advantages of *edge2vec*.

2.3 Knowledge graph embedding

Knowledge graphs are directed graphs whose nodes correspond to entities and edges represent the rich relations between entities. Knowledge graph embedding methods model the knowledge graphs through embedding the entities into a low-dimensional space [4], just like what network embedding methods do. Inspired by [26], TransE [3] represents the relations as the translation in the embedding space: If a triple (h, r, t) holds, where h and t are the head entity and the tail entity w.r.t. the relationship r, the embedding of t should be close to the embedding of h plus the embedding of r. TransE cannot model 1-to-N, N-to-1, and N-to-N relations, and some follow-up methods (TransH [37] and TransR [23]) are proposed to address this issue.

Although these knowledge graph embedding approaches can map the edges (relations) in a knowledge graph to low-dimensional vectors, they do not apply to our edge-based network embedding problem. The reason is that edges are of different types in knowledge graphs, and each type of edges corresponds to a single embedding vector through knowledge graph embedding methods. We want to learn different representations for different edges, but in social networks, edges are of only one or a few types (e.g. friends or foes). Thus, existing knowledge graph embedding methods cannot be used in our problem.

3 PROBLEM DEFINITION

In this section, we give some useful concepts as well as the formal description of the edge-based social network embedding problem.

Definition 1 (Social network). *A social network is denoted as a directed graph $G = (V, E)$, where V is the node set and E is the edge set. Each node $v \in V$ represents an individual. Each edge $e \in E$ is an ordered node pair $e = (s, t)$ $(s, t \in V)$, which represents a social tie from s to t, and s and t are called the source node and the target node of e, respectively.*

In the above definition, we use a directed graph to represent the structure of the social network. For directed social networks, this approach is natural. Moreover, for an undirected social network, it is also easy to represent it with a directed graph through mapping each undirected social tie with two opposite directed edges. Then, we define the local and the global proximity between edges as follows.

Definition 2 (Local edge proximity). *For two edges $e_1 = (s_1, t_1)$ and $e_2 = (s_2, t_2)$, if $t_1 = t_2$ or $s_1 = s_2$, the local edge proximity of e_1 to e_2 is 1, otherwise it is 0.*

The local edge proximity between two edges is calculated depending on the relationship of source nodes and target nodes of the two edges. We believe that two edges which share the common source node or the common target node should be similar to each other from the view of local structure information of a network.

As we described above, for an undirected social network, we represent each of its undirected edges through two directed edges. Thus, if there are two undirected edges $e_1 = (s, t)$ and $e_2 = (s, r)$, we can represent them as four directed edges: $e_{11} = (s, t)$, $e_{12} = (t, s)$, $e_{21} = (s, r)$ and $e_{22} = (r, s)$. For edge pairs (e_{11}, e_{21}) and (e_{12}, e_{22}), the local edge proximity between the two edges of each pair is 1. However, the local edge proximity between the two edges of any other edge pair taken out of these four directed edges, e.g., the edge pair (e_{12}, e_{21}), is 0.

Definition 3 (Global edge proximity). *For an edge $e = (s, t)$, we use a neighborhood vector $n_e = (w_{s1}, ..., w_{s|V|}, w_{t1}, ..., w_{t|V|})$ to represent its neighborhood structure, where $w_{ij} \in [0, 1]$ denotes the neighborhood relationship between nodes i and j, and can be defined in different ways. The global edge proximity between two edges e_1 and e_2 is defined as the similarity of n_{e_1} and n_{e_2}.*

If two edges have similar neighborhood vectors, we consider they play similar roles in communicating nodes in the network, and they should be similar from the global view of the network structure. The detailed discussion about the neighborhood relationship as well as the similarity metric for neighborhood vectors is presented in Section 4.2. Last, the definition of edge-based social network embedding is as follow.

Definition 4 (Edge-based social network embedding problem). *Given a social network $G = (V, E)$, the aim of **edge-based social network embedding** is to represent each edge $e \in E$ in a low-dimensional space R^d $(d \ll |E|)$. The local and the global proximity between all the edges should be preserved as much as possible after mapping all the edges into R^d.*

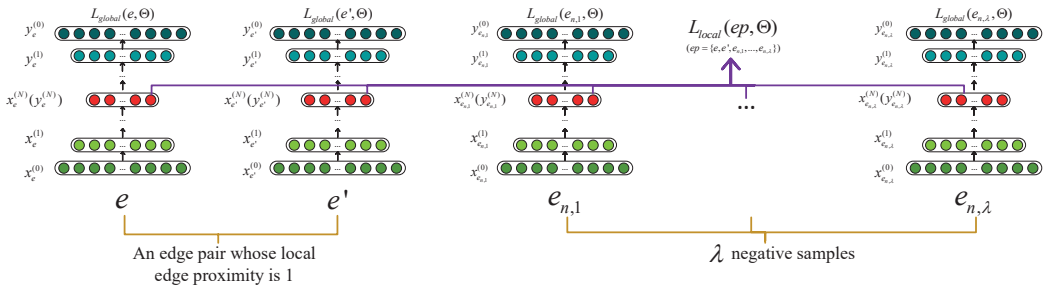


Fig. 3. The framework of edge2vec.

4 METHOD

In this section, we present how **edge2vec** embeds a social network with both the local and the global proximity preserved. First, we present the framework of **edge2vec**, i.e. a deep neural network consisting of multiple deep autoencoders which share the same structure and parameters. Next, we preserve the global edge proximity between edges through minimizing the reconstruction error of neighborhood vectors with the deep autoencoder. Then, we preserve the local edge proximity by maximizing the "co-occur" probability of two edges sharing the same source/target node under the skip-gram model. After that, we propose the element-wise joint loss for the deep neural network of **edge2vec**, and discuss how to embed a social network with **edge2vec**. Last, we analyze the time and space complexity of **edge2vec**.

4.1 The framework of edge2vec

The framework of **edge2vec** is shown in Figure 3. We employ a deep neural network consisting of multiple deep autoencoders which share the same structure and parameter Θ.

For each deep autoencoder, its input is the *neighborhood vector* of a given edge. Meanwhile, it has two outputs. The first output is the *representation vector* in the middle of the the deep autoencoder, which is the embedding vector for the given edge. The second one is the *reconstructed vector* in the top of it.

The input of the complete deep neural network is a group of edges $ep = (e, e', e_{n,1}, ...e_{n,\lambda})$ and their corresponding neighborhood vectors. In ep, e and e' is a pair of edges between which the local edge proximity is 1, and $e_{n,1}, e_{n,2}, ..., e_{n,\lambda}$ are λ negative samples, which means the local edge proximity between e and each of them is 0.

First, for each edge, we assign it to a deep autoencoder in the neural network, and define the loss L_{global}, which is a variant of the reconstruction error for ordinary deep autoencoders, based on the *reconstructed vector* and the *neighborhood vector* for this edge to ensure the *representation vector* can preserve the property of the *neighborhood vector*.

Next, we define the loss L_{local} on the embedding vectors of all the edges in the group. The definition of L_{local}, inspired by the Skip-gram model [26], tries to make the embedding result by the neural network preserve the local edge proximity, i.e. we would like the proximity of e to e' after embedding is close and that of e to $e_{n,i} (i \in 1, 2, ..., \lambda)$ is far.

Then, we obtain a joint loss function for the input edge group through combining L_{global} and L_{local}. By minimizing this joint loss, we learn the shared parameter Θ for the deep autoencoders.

Finally, each edge in the network can be embedded to a vector through the deep autoencoder with parameter Θ.

Notation	Meaning
w_{ij}	neighborhood relationship between nodes i, j
A_{ij}	adjacency relationship between nodes i, j
N	depth of the deep autoencoder
$x_e^{(n)}$	representation vector for edge e in the n-th layer
$y_e^{(n)}$	reconstructed vector for edge e in the n-th layer
$x_e^{(0)}$	input vector for edge e
$y_e^{(0)}$	final reconstructed vector for edge e
$x_e^{(N)}, y_e^{(N)}, \vec{e}$	embedding vector for e
$W^{(n)}, b^{(n)}$	encoder parameters in the n-th layer
$M^{(n)}, d^{(n)}$	decoder parameters in the n-th layer
Θ	deep autoencoder parameter consisting of $\{W^{(n)}, b^{(n)}, M^{(n)}, d^{(n)}\}_{n\in1,2,\ldots,N}$
k	distance of the furthest indirect neighbor taken into consideration
β	decaying factor for indirect neighbor
p	penalty factor for non-zero values when calculating reconstruction error
λ	the number of negative samples
α	combination factor
(e, e')	an edge pair between which the local edge proximity is 1
$\{e_{n,i}\}_{i=1}^{\lambda}$	negative samples w.r.t. (e, e')
m	the number of examples for training

Table 1. Notations

Before introducing the details of **edge2vec** in the next subsections, we define some of the notations which are used later in Table 1.

4.2 Preserving the global edge proximity

We first discuss how to capture the existing global edge proximity in a social network. In Definition 3, the global edge proximity between two edges is the similarity of their neighborhood vectors, which contains neighborhood relationship between the source/target node of the edge and each node in the graph. Thus, calculating the neighborhood relationship between two arbitrary nodes is the key step to capture the global edge proximity between the two edges of each edge pair.

4.2.1 Neighborhood relationship. Some existing graph embedding methods [30, 32, 34] use the adjacency relationship, i.e.,

$$w_{ij} = A_{ij} = \begin{cases} 1 & if\ (i,j) \in E \\ 0 & otherwise \end{cases}. \tag{1}$$

This approach cannot represent the indirect neighborhood relationship between two nodes, and thus it cannot capture the global edge proximity well.

GraRep [5] considers the k-step neighbor relationship between nodes. It proposes the k-step probability transition matrix A^k, where A_{ij}^k exactly refers to the transition probability from node i to node j between which the transition consists of exactly k steps. Given the maximum step length K, each A^k $(k \leq K)$ is used separately, and K embedding vectors corresponding to the same node are concatenated to form the final vector, which means the length of the final embedding vectors is far longer than that of other embedding methods. Thus, this approach is not appropriate, either.

ALGORITHM 1: Calculating the k-step adjacent matrix w^k.

Input: The social network $G = (V, E)$ with the adjacent matrix A, and the step length k ($k \leq 2$).
Output: The k-step adjacent matrix w^k.

```
w^1 ← A;
for n from 2 to k do
    w^n ← w^{n-1};
    for each (i, m) that w^{n-1}_{im} ≠ 0 do
        for each j that w^1_{mj} ≠ 0 do
            w^n_{ij} ← max(w^n_{ij}, w^{n-1}_{im} × w^1_{mj} × β);
        end
    end
end
Output w^k.
```

In **edge2vec**, we employ a k-step adjacency relationship to describe the neighborhood relationship of two nodes

$$\begin{aligned} w_{ij}^1 &= A_{ij}, \\ w_{ij}^k &= \max\left(\max_m(w_{im}^{k-1} \times w_{mj}^1 \times \beta), w_{ij}^{k-1}\right), \end{aligned} \tag{2}$$

where β is the decaying factor for each step. w^k is a k-step adjacency matrix, where w_{ij}^k reflects the strength of neighborhood relationship in k steps between nodes i and j. If there exists a path from i to j within k steps, w_{ij}^k will be larger than 0. The shorter the path is, the larger w_{ij}^k will be. We do not need k separate values to represent the neighbor relationship between i and j like [5]. Thus, this approach considers indirect neighborhood relationship as well as avoids the increase of the length of embedding vectors. The computation of w^k is shown in Algorithm 1. The worst time complexity of this algorithm is $O(n^3)$ when G is a complete graph. However, as we mentioned in the introduction, the average degree of a social network is bounded. Thus, the practical time cost of this algorithm is far less than the worst case.

We have introduced how to calculate the neighborhood relationship for two nodes. Now we are able to generate the neighborhood vector for each edge. According to Definition 3, the global proximity between two edges is defined as the similarity between their neighborhood vectors. However, there exist various similarity metrics for vectors. If we want to preserve the global proximity under various vector similarity metrics, we must preserve the characteristics of neighborhood vectors as much as possible. Inspired by [32, 34], we use a deep autoencoder to transform neighborhood vectors to low-dimensional representation vectors. We believe that it can preserve the characteristics of the neighborhood vector for each edge well after transforming.

4.2.2 Deep autoencoder. A deep autoencoder [29] contains multiple nonlinear transformations in both the encoder and the decoder. As shown in Figure 3, there are multiple deep autoencoders in **edge2vec**, which all share the same structure and the parameters. We take the first of them as an example. Its input is $x_e^{(0)}$ which represents the neighborhood vector of the given edge e.

$$x_e^{(0)} = n_e = (w_{s1}^k, ..., w_{s|V|}^k, w_{t1}^k, ..., w_{tN}^k) \tag{3}$$

where $(s, t) = e$. The encoder is as follows:

$$x_e^{(n)} = \sigma\left(W^{(n)} x_e^{(n-1)} + b^{(n)}\right),\ n = 1, 2, ..., N\ , \tag{4}$$

where $W^{(n)}$ and $b^{(n)}$ are the weights and bias of the n-th layer in the encoder, σ is the sigmoid function, and N is the depth of the deep autoencoder.

The vector $x_e^{(N)}$ is the representation vector for the neighborhood vector of e, and is also the embedding vector for e.

Let $y_e^{(N)} = x_e^{(N)}$. The decoder is as follows:

$$y_e^{(n-1)} = \sigma\big(M^{(n)} y_e^{(n)} + d^{(n)}\big),\ n = 1, 2, ..., N\ , \tag{5}$$

where $M^{(n)}$ and $d^{(n)}$ are the weights and bias of the n-th layer in the decoder. The vector $y_e^{(0)}$ is the reconstructed vector based on the representation vector $x_e^{(N)}$.

The loss function of this deep autoencoder is defined as follows:

$$L_{global}(e, \Theta) = \|(y_e^{(0)} - x_e^{(0)}) \circ I_e\|_2^2\ , \tag{6}$$

where $\circ$ is the Hadamard product and I_e is the indicator vector for $x_e^{(0)}$ [34]. Let $I_e = \{I_{e,i}\}_{i=1}^{2|V|}$, $x_e^{(0)} = \{x_{e,i}^{(0)}\}_{i=1}^{2|V|}$, and I_e is defined as follows:

$$I_{e,i} = \begin{cases} p & if\ x_{e,i}^{(0)} > 0 \\ 1 & otherwise \end{cases}\ , \tag{7}$$

where $p > 1$ is the penalty factor.

As we know, social networks are often very sparse, and thus the neighborhood vectors for edges are sparse too. Obviously, we care about the non-zero values more than the zeros, and thus we would like to have representation vectors which are able to reconstruct as many as possible non-zero values in the neighborhood vectors. Through introducing the indicator vector, we add penalty on the reconstruction error of the non-zero values in Eq. 6.

Through minimizing the loss defined in Eq. 6, we can learn the parameters of the deep autoencoder that can ensure the property of input neighborhood vectors preserved. Since the neighborhood vectors can represent the global structure information well, the global edge proximity must be preserved when we embed the social network with this trained deep autoencoder.

4.3 Preserving the local edge proximity

Now we present how to preserve the local edge proximity in this subsection.

4.3.1 Skip-gram. If the local edge proximity between edge e and e' is 1 but not 0, they must share either the same source node or the same target node. We employ the Skip-gram [26] to model this local structure property in networks like [5, 9]. We consider e as the "target edge" and e' as the "context edge", and then we aim to maximize the probability that e generates e':

$$p(e'|e) = \frac{exp(\vec{e} \cdot \vec{e'})}{\sum_{i=1}^{|E|} exp(\vec{e} \cdot \vec{e}_i)} \tag{8}$$

where $\vec{e}$ refers the embedding vector for edge e. Eq. 8 is defined under the assumption of "symmetry in feature space" [9], which means the context vector and the target vector of an edge are both the same as the embedding vector.

However, the denominator of Eq. 8 is hard to compute. We adopt the Negative sampling method [26] to approximate the objective function as follows:

$$o(e, e') = \log \sigma(\vec{e} \cdot \vec{e'}) + \sum_{i=1}^{\lambda} E_{e_i \sim P_n(e)}\big(\log \sigma(-\vec{e} \cdot \vec{e}_i)\big) \tag{9}$$

which takes place of $\log p(e'|e)$. In Eq. 9, $P_n(e)$ is the noise distribution, and λ is the number of negative samples.

4.3.2 Deep autoencoders with Skip-gram. In the previous subsection, we introduce how to learn the parameters of a single deep autoencoder to embed edges with global edge proximity preserved. Now we concatenate multiple deep autoencoders which share the same structure and parameters (as shown in Figure 3), and discuss how to learn the shared parameters that can ensure the local edge proximity preserved.

For a given edge pair (e, e') bwtween which the local edge proximity is 1, we draw λ negative samples, i.e., $(e_{n,1}, e_{n,2}, ..e_{n,\lambda})$. We generate an input example, denoted as ep, for the complete deep neural network in **edge2vec**, which consists of the neighborhood vectors corresponding to e, e' and the λ negative samples. The loss function for preserving the local edge proximity is defined based on Eq. 9 as follows:

$$L_{local}(ep, \Theta) = -\log \sigma\left(x_e^{(N)} \cdot x_{e'}^{(N)}\right) - \sum_{i=1}^{\lambda} \left(\log \sigma\left(- x_e^{(N)} \cdot x_{e_{n,i}}^{(N)}\right)\right) \quad , \tag{10}$$

where we use the opposite form of the objective function in Eq. 9 to change the optimization goal from maximizing $o(e, e')$ to minimizing $L_{local}(ep, \Theta)$. By minimizing this local loss function, we can learn the shared parameters of deep autoencoders to preserve the local edge proximity.

4.4 Embedding social networks with edge2vec

Through combining the global loss in Eq. 6 for all the deep autoencoders with the local loss in Eq. 10, the example-wise joint loss is defined as:

$$L(ep, \Theta) = \alpha \sum_{c \in ep} L_{global}(c, \Theta) + (1 - \alpha) L_{local}(ep, \Theta) \quad , \tag{11}$$

where α is the combination factor to balance the two parts of the joint loss.

To optimize the **edge2vec** model, we use the stochastic gradient descent to minimize the example-wise joint loss L defined in Eq. 11, and thus we need to calculate the partial derivative of L w.r.t. the parameter Θ.

First, the partial derivative of the joint loss L is shown as follows:

$$\frac{\partial L(ep, \Theta)}{\partial \Theta} = \alpha \sum_{c \in ep} \frac{\partial L_{global}(c, \Theta)}{\partial \Theta} + (1 - \alpha) \frac{\partial L_{local}(ep, \Theta)}{\partial \Theta} \quad . \tag{12}$$

Next, we study the term corresponding to the global loss, i.e. $\frac{\partial L_{global}(c,\Theta)}{\partial \Theta}$ in Eq. 12. According to Eq. 6, it can be calculated as follows:

$$\begin{aligned} \frac{\partial L_{global}(c, \Theta)}{\partial \Theta} &= \frac{\partial \| \left(y_c^{(0)} - x_c^{(0)}\right) \circ I_c \|_2^2}{\partial y_c^{(0)}} \cdot \frac{\partial y_c^{(0)}}{\partial \Theta} \\ &= 2 I_c \circ I_c \circ \left(y_c^{(0)} - x_c^{(0)}\right) \cdot \frac{\partial y_c^{(0)}}{\partial \Theta} \end{aligned} \tag{13}$$

where $x_c^{(0)}$ and $y_c^{(0)}$ are the input vector and reconstructed vector for the deep autoencoder w.r.t. the edge c, and I_c is the indicator vector for $x_c^{(0)}$. Thus, $\frac{\partial y_c^{(0)}}{\partial \Theta}$ is the partial derivative of the reconstructed vector for the deep autoencoder w.r.t. the autoencoder parameter Θ. According to the encoding and decoding process defined in Eq. 4 and Eq. 5, $\frac{\partial y_c^{(0)}}{\partial \Theta}$ is easy to be calculated based on the back propagation method. Therefore, we can get $\frac{\partial L_{global}(c,\Theta)}{\partial \Theta}$.

ALGORITHM 2: Embedding a social network with edge2vec.

Input: The social network $G = (V, E)$, the structure of the deep autoencoder, the hyper parameters $\alpha, \beta, k, p, \lambda, m$.

Output: The embedding vectors for all $e \in E$.

```
Generate the neighborhood vector (x_e^(0)) for each e ∈ E;
Pre-train the deep autoencoder layer by layer through treating it as a stacked autoencoder to obtain a
  good initial value of the deep autoencoder parameter Θ;
Sample m edge pairs whose local edge proximity is 1 uniformly to form a set EP';
for each edge pair (e, e') in EP' do
    Draw λ negative edges, so that the local edge proximity between e and each negative edge is 0;
    Generate an input example with (e, e') and the λ negative edges;
end
Combine examples into a set EP;
repeat
    for each example ep in EP do
        for each edge e in ep do
            Based on Θ, calculate the embedding vector x_e^(N) and the reconstructed vector y_e^(0) for e
              according to Eq. 4 and Eq. 5;
        end
        Calculate the joint loss L with Eq. 11;
        Update Θ through gradient decent and back propagation based on the partial derivate in Eq. 12;
    end
until convergence;
for each edge e in E do
    Transform e to embedding vector e⃗ = x_e^(N) with the learned Θ.
end
Output {e⃗}_{e∈E}.
```

Then, we focus on the term $\frac{\partial L_{local}(ep,\Theta)}{\partial \Theta}$ in Eq. 12, which is corresponding to the local loss. Based on Eq. 10, this term can be calculated as follows:

$$\begin{aligned}
\frac{\partial L_{local}(ep,\Theta)}{\partial \Theta} =& \frac{\partial\left(-\log\sigma\left(x_e^{(N)}\cdot x_{e'}^{(N)}\right)-\sum_{i=1}^{\lambda}\left(\log\sigma\left(-x_e^{(N)}\cdot x_{e_{n,i}}^{(N)}\right)\right)\right)}{\partial\Theta} \\
=& \left(x_{e'}^{(N)}\left(\sigma\left(x_e^{(N)}\cdot x_{e'}^{(N)}\right)-1\right)+\sum_{i=1}^{\lambda}x_{e_{n,i}}^{(N)}\left(\sigma\left(x_e^{(N)}\cdot x_{e_{n,i}}^{(N)}\right)\right)\right)\cdot\frac{\partial x_e^{(N)}}{\partial\Theta} \\
&+\left(x_e^{(N)}\left(\sigma\left(x_e^{(N)}\cdot x_{e'}^{(N)}\right)-1\right)\right)\cdot\frac{\partial x_{e'}^{(N)}}{\partial\Theta} \\
&+\sum_{i=1}^{\lambda}\left(x_e^{(N)}\left(\sigma\left(x_e^{(N)}\cdot x_{e_{n,i}}^{(N)}\right)\right)\cdot\frac{\partial x_{e_{n,i}}^{(N)}}{\partial\Theta}\right)
\end{aligned} \tag{14}$$

where $\frac{\partial x_c^{(N)}}{\partial\Theta}(c = e, e', e_{n,1}, \ldots e_{n,\lambda})$ is the partial derivative of the embedding vector for the edge c w.r.t. the deep autoencoder parameter Θ. Similar to the calculation of $\frac{\partial y_c^{(0)}}{\partial\Theta}$ in Eq. 13, $\frac{\partial x_c^{(N)}}{\partial\Theta}$ can be

obtained through the back propagation on the deep autoencoder. Hence, we can calculate the term $\frac{\partial L_{local}(ep,\Theta)}{\partial \Theta}$ in Eq. 12.

Finally, combining Eq. 13 and Eq 14, we obtain the partial derivative of the joint loss in Eq. 12 w.r.t. the deep autoencoder parameter Θ.

Algorithm 2 shows the pseudo-code of embedding a given social network with **edge2vec**. We first generate the neighborhood vector for each edge (Line 1). Next, we use a greedy layer-wise pre-training process to find a good initial value of Θ (Line 2). Then, we sample m edge pairs (Line 3), which is similar to the random walk process in DeepWalk and the edge sampling in LINE, and generate examples based on the edge pairs with negative sampling method (Lines 4 – 8). After that, we train the **edge2vec** model through stochastic gradient descent and back propagation with the joint loss defined in Eq. 11 (Lines 9 – 17). At last, we transform each edge to its embedding vector with the learned Θ, and then we finish the edge-based network embedding (Lines 18 – 21).

4.5 Complexity analysis

In this subsection, the time and space complexity of **edge2vec** as well as some other graph embedding methods (DeepWalk [27], Node2vec [9], LINE [30] and SDNE [34]) are presented.

For DeepWalk and Node2vec, they are both based on random walks, and the time complexity of training is $O(\gamma twd|V|)$, where γ is the iteration number, t is the maximum length of walks, w is the window size of skip-gram, d is the dimension of embedding, and $|V|$ is the number of nodes. LINE does not need random walks, but leverages all the edges during the training process. Thus, its time complexity is $O(\tau d|E|)$, where τ is the iteration number and $|E|$ is the number of edges. The time cost of SDNE is $O(\tau h|V||E|)$, where h is the size of the first hidden layer. As for our **edge2vec**, the training time cost is $O(\tau hm|V|)$, where m is the number of sampled edge pairs. Since both SDNE and **edge2vec** employ a deep neural network, they can easily speed up through training with GPUs.

The memory cost of these embedding methods mainly depends on the number of parameters. Thus, the space complexity of DeepWalk, Node2vec and LINE is $O(d|V|)$, and that of SDNE and **edge2vec** is $O(h|V|)$.

5 EXPERIMENTS

In this section, we evaluate the effect of **edge2vec**. We first introduce the data sets and baseline embedding methods in Section 5.1. Next, we conduct experiments on three analysis tasks for edges, i.e. link prediction, social tie direction prediction and social tie sign prediction, to compare **edge2vec** with the baselines in Sections 5.2 – 5.4. Then, in Section 5.5, we analyze the edge proximity preservation quantitatively. At last, we demonstrate the parameter sensitivity of **edge2vec** in Section 5.6.

5.1 Experimental settings

We use six data sets collected from real-world social networks.

- Epinions [46]. This data set is extracted from the general consumer review site Epinions.com as a who-trust-whom social network. The trust information is not considered in the data set.
- LiveJournal [46]. It is extracted from the LiveJournal online social network.
- Slashdot [46]. Slashdot allows users to tag each other as friends or foes. However, in this data set, only the friend links are considered while the foe links are ignored.
- Tencent [46]. It is extracted from Tencent Weibo, a microblogging website in China.

Data sets	# Nodes	# Edges	Autoencoder Structure
Epinions	3,248	126,385	6,496 - 1,024 - 128
LiveJournal	5,192	115,358	10,384 - 1,024 - 128
Slashdot	10,428	344,862	20,856 - 4,096 - 512 - 128
Tencent	101,460	510,364	202,920 - 4,096 - 512 - 128
Epinions-sign	7,982	130,357	15,964 - 2,048 - 128
Wikipedia	7,125	107,080	14,250 - 2,048 - 128

Table 2. The statistics of the data sets.

- Epinions-sign [19]. This is also extracted from Epinions.com. Different from the first data set, Epinions, this data set considers the trust and distrust between individuals, i.e. , this is a signed network.
- Wikipedia [19]. This network corresponds to votes cast by Wikipedia users in elections for promoting individuals to the role of admin. A signed edge from one user to another indicates a positive or negative vote by the former on the promotion of the latter.

Notice that the above six data sets are all unweighted and directed social networks, and the first four are unsigned while the last two are signed networks. The unsigned networks are used in the experiments in Section 5.2 and Section 5.3, while the signed networks are used in the experiments in Section 5.4 and Section 5.5, and experiments in Section 5.6 are conducted on all the networks. The statistics of the networks we used in the experiments are shown in Table 2.

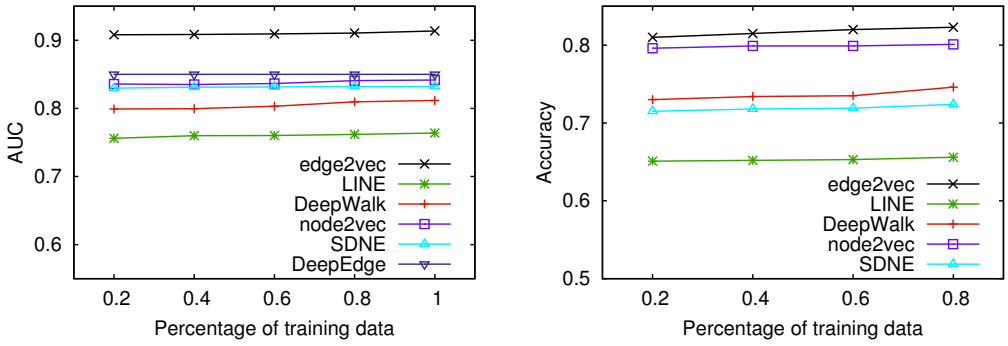
Four state-of-the-art graph embedding methods are used as baselines in the experiments.

- DeepWalk [27]: It is based on the Skip-gram model, and employs truncated random walks to obtain the context information of nodes.
- Node2vec [9]: This method improves the DeepWalk by replacing the random walk with a new search strategy which combines the properties of BFS and DFS.
- LINE [30]: It defines a loss function based on the first-order and the second-order proximity between nodes to learn graph representations on large-scale information networks.
- SDNE [34]: This approach employs a Siamese network consisting of two deep autoencoders, such that both the first-order proximity and the second-order proximity of nodes can be preserved.

We only compare edge2vec with existing network embedding methods but do not consider the state-of-the-art methods for any specific tasks. As far as we know, this baseline setting is adopted in all the network embedding studies [9, 27, 30]. Network embedding methods provide the capability to deal with lots of analysis tasks on networks. We value the generality of network embedding methods, and a good network embedding method should outperform other embedding methods in many tasks.

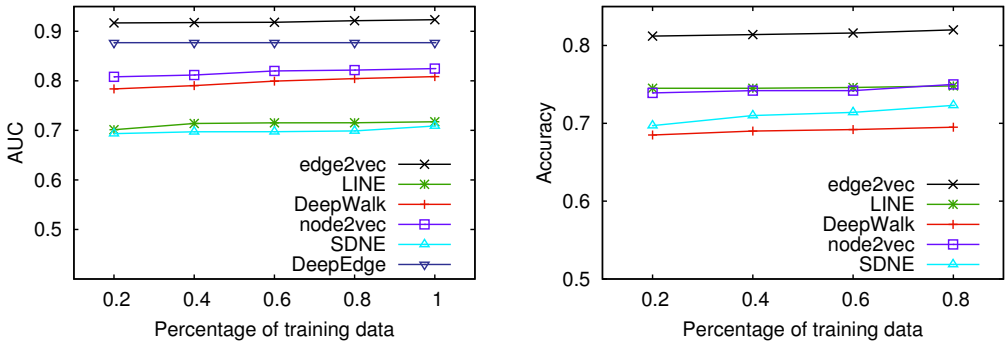
We obtain the source code of the first three methods from their corresponding projects on Github. We implement SDNE by ourselves, since the source code is not publicly available.

These embedding methods are all node-based, and thus we need to generate vectors for edges according to the embedding vectors of its source node and target node. The approaches in [9] are not appropriate here for the directed social networks, because these approaches will generate the same vector for edge (s, t) and (t, s) given the embedding vectors of t and s. Thus, we employ a simple but effective approach which concatenates the embedding vectors of t and s to generate the vector representation of edge (t, s).



(a) Link prediction on Epinions. (b) Social tie direction prediction on Epinions.

Fig. 4. Link prediction and social tie direction prediction on Epinions.



(a) Link prediction on LiveJournal. (b) Social tie direction prediction on LiveJournal.

Fig. 5. Link prediction and social tie direction prediction on LiveJournal.

We set the embedding dimension of the four baselines as 64 and that of **edge2vec** as 128, so that we can ensure the vector representations generated by all these methods are as of the same length. The number of samples for DeepWalk, LINE and node2vec are set to the same [9]. Other hyper-parameters of baselines are set according to the corresponding papers, e.g., p and q for node2vec are learned by a grid search over $\{0.25, 0.5, 1, 2, 4\}$. For **edge2vec**, we set $\beta = 0.5, p = 10, \lambda = 5$ through experience and previous work [30, 34]. The optimal values of α and k is discovered by a grid search with cross-validation on the training set, and we will discuss them in Section 5.6.

The structures of the deep autoencoder in **edge2vec** for different data sets are shown in Table 2. For example, 6,496-1,024-128 means a two-layer autoencoder, the input and output numbers of the first layer are 6,496 and 1,024, and those of the second layer are 1,024 and 128. It is worth noting that the input number for the first layer is always twice of the node number, since the length of neighborhood vectors is twice of the node number.

5.2 Link prediction

Our first task is link prediction, a classical problem for social networks. The aim of link prediction problem is to predict whether a given edge which does not exist will appear in the future.

We use a similar preprocessing step in [9], which first employs the graph embedding method in link prediction task. We first remove 20% edges from the original network randomly while the connectivity of the residual network is ensured. The edges in the residual network are treated as positive samples in the training set, and the removal edges are treated as positive samples in the test set. We sample node pairs that are not connected in the residual network as negative training samples, and node pairs that are not connected in the origin network as negative test samples. Negative samples in the training/test set are both with the same number of the positive ones in the corresponding set.

In this task, we introduce a specific baseline.

- DeepEdge [1]: It is a node-based embedding method trained for link prediction task. It learns embedding vectors for nodes and an edge function for edges. The edge function maps a node pair to a real value to present an edge.

This baseline is not used in following tasks because its edge function is trained for link prediction specifically. We obtain its source code from Github.

For **edge2vec** as well as baselines except for DeepEdge, we first learn the embedding vectors for edges, and then train a logistic regression classifier with different percentages of the training set. For DeepEdge, we treat it as an end-to-end link prediction model as in [1], and thus it uses all the samples of the training set. The performances of different methods in this task is measured with the Area Under the ROC Curve (AUC) metric on the test set.

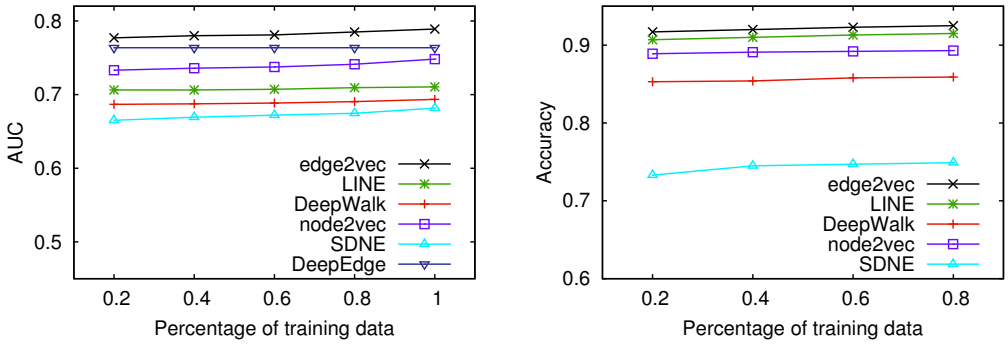
Please note that we only compare **edge2vec** with embedding-based link prediction methods, since the experimental results in [1, 9] show that the link prediction methods based on heuristic scores always achieve worse performance than embedding-based link prediction methods.

The results are shown in Figures 4(a), 5(a), 6(a) and 7(a) (DeepEdge can be only conducted with 100% training set, so we draw a line with the result corresponding to 100% percentage in each figure to show its performance). We can observe that **edge2vec** outperforms all the baselines in all the unsigned networks significantly. This is because it can preserve both the global and the local proximity of edges well, and baselines only preserve the proximity of nodes. DeepEdge is always the second-best, since it learns an edge function specified for link prediction task. As we discussed, it represents edges through the node vectors and the edge function. The results show that our strategy that represents edges with vectors is more effective. Node2vec outperforms DeepWalk, due to its improvement on the search strategy.

There is an interesting fact that the performance of all the methods, except for DeepEdge which is only conducted with 100% training set, does not get better significantly with the increase of training data. We know that this experimental task consists of two parts: The unsupervised part, i.e., the embedding model, learns how to transform edges to vectors from the network of 80% of the edges, where the supervised part learns to classify edges based on the remaining 20%. It is obvious that the proportion of training data used in the supervised part could not affect the unsupervised part, and thus we find that the final performance is mainly determined by the unsupervised part.

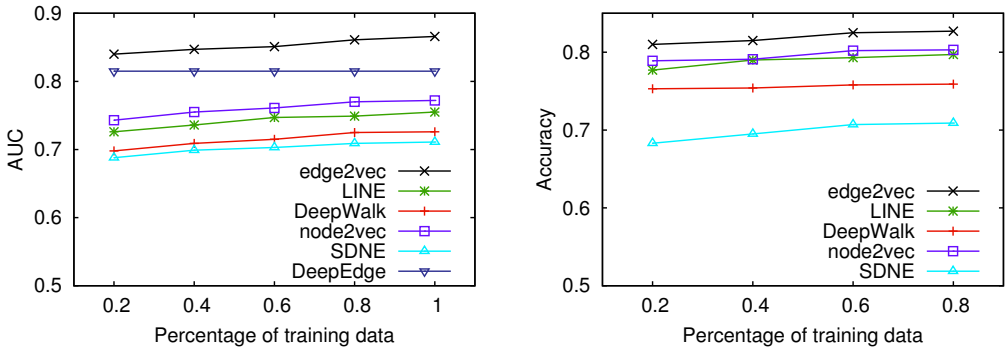
5.3 Social tie direction prediction

The study on the social tie direction is novel and interesting [46]. Zhang et. al. argue that there exists directionality information hidden in undirected social networks, and propose a method to recover the directions of social ties in a given undirected social networks. In our experiments, we modify this problem to a supervised learning problem. Supposing the directions of a certain number



(a) Link prediction on Slashdot. (b) Social tie direction prediction on Slashdot.

Fig. 6. Link prediction and social tie direction prediction on Slashdot.



(a) Link prediction on Tencent. (b) Social tie direction prediction on Tencent.

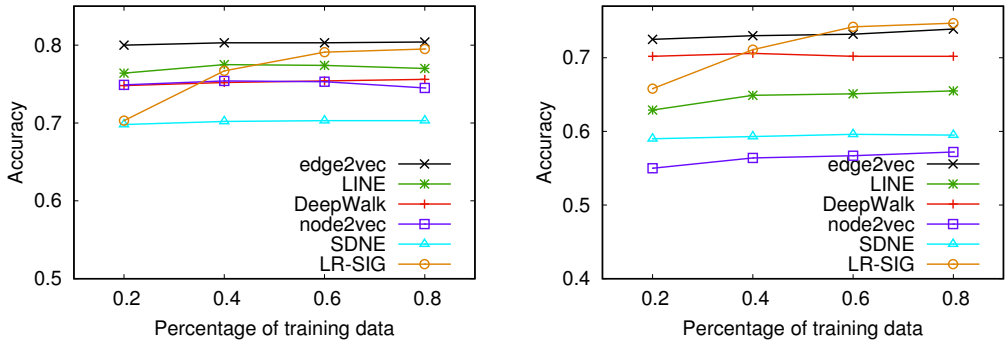
Fig. 7. Link prediction and social tie direction prediction on Tencent.

of social ties are unknown in a directed social network, we aim to predict the directions of them with the network structure information.

First, we hide the directions of the social network to generate an undirected network, and train the graph embedding models on the undirected network. Next, we select a various amount of edges with their directionality information as training data, treat the remaining edges as the test set, and transform them into the low-dimensional space through the graph embedding models. Last, we train a logistic regression classifier on the training set, and measure its performance on the test set with the metric of accuracy.

Figures 4(b), 5(b), 6(b) and 7(b) show the experimental result. **Edge2vec** is also the best embedding algorithm in this task. The relationship between the performance of DeepWalk and node2vec remains the same as that in the link prediction task. However, the performance of LINE is unstable and varies on different data sets. We think the reason is the different sampling approach used by LINE and DeepWalk/node2vec.

We can also observe that the proportion of training data matters little for the similar reason in the link prediction experiment.



(a) Epinions-sign (b) Wikipedia

Fig. 8. Social tie sign prediction.

5.4 Social tie sign prediction

Social tie sign prediction problem [19] is one of the studies on signed social networks. Different from normal (unsigned) social networks, social ties in signed social network can be either positive or negative. Usually, a positive tie means trust or approval, while a negative one indicates dislike or disapproval.

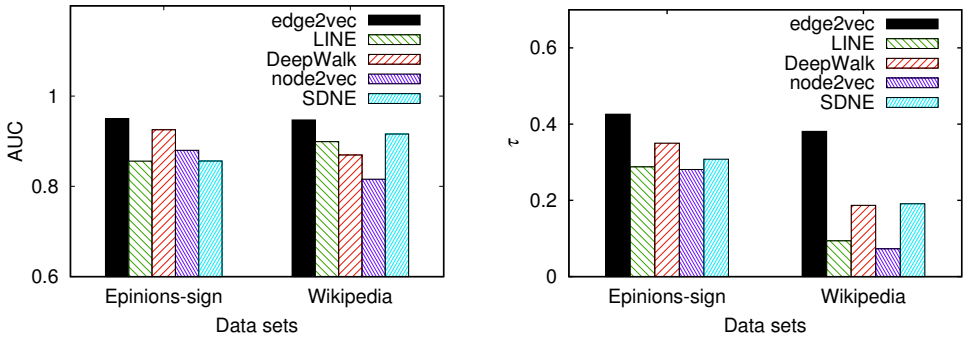
Like the experiments in previous subsections, we first embed the networks with different approaches and obtain the representation vectors for all the edges. Next, we follow the step in [43] to create a balanced data set for each signed network, since the number of positive edges is much higher than that of negative edges in each network. We randomly sample a positive edge for each negative edge so that the number of positive and negative edges is equal. Then we train a logistic regression classifier on each data set and measure the embedding result based the prediction accuracy.

In this task, a non-embedding state-of-the-art sign prediction method is introduced as a baseline.

- LR-SIG [6]: It is a global method which utilizes a low rank model of signed networks to solve the sign prediction problem. By using a gradient based matrix factorization approach and a sigmoid loss function, it completes the adjacency matrix efficiently with high-precision.

Figure 8 shows the experimental results on Epinions-sign and Wikipedia. **Edge2vec** outperforms all the embedding baselines, which is the same as the results in previous subsections. However, node2vec performs worse than DeepWalk in this experiment. For the non-embedding baseline LR-SIG, **Edge2vec** is always better than it on Epinions-sign. On Wikipedia, **Edge2vec** outperforms LR-SIG when percentage of training data is small (0.2 and 0.4), and is slightly worse than it when the percentage gets larger (0.6 and 0.8). These results demonstrate that **Edge2vec** is very good at the task of sign prediction.

We can find that in this experiment, for all the embedding methods the accuracy of classification is still relatively stable as the amount of training data increases, just like the experiments in the previous two subsections. Thus, in all the analysis tasks we study in this paper, the quality of embedding methods is the key factor. It implies that when we want to analyze the edges of a given social network (including link prediction and edge classification) through edge embeddings, the help of good embedding methods (e.g. **edge2vec**) is much greater than that of more labeled data.



(a) Local proximity (b) Global proximity

Fig. 9. Edge proximity preservation.

The result of the above three analysis tasks shows that the edge-based graph embedding method is really necessary for the tasks on edges in social networks. For node-based graph embedding vectors, the properties of an edge cannot be described completely only by the embedding vectors of the source/target node.

5.5 Edge proximity preservation

In this subsection, we quantitatively study how much of the global/local edge proximity is preserved after embedding with different approaches. The experiments in this subsection are conducted on the two signed data sets, i.e., Epinions-sign and Wikipedia.

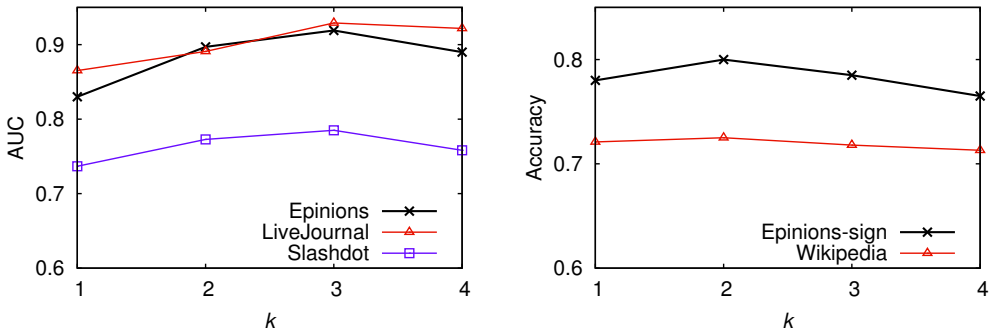
5.5.1 The local edge proximity preservation. We first look at the local edge proximity. According to Definition 2, the local edge proximity between two edges is 1 if they share common source/target node, otherwise it is 0. For both of the two networks, we first sample 10,000 positive edge pairs (between which the local edge proximity is 1) and 10,000 negative edge pairs (between which the local edge proximity is 0). Then we embed the two networks with different embedding approaches and obtain the embedding vectors for all the edges in the 20,000 edge pairs. Next, for each edge pair, we define the proximity between the two edges after embedding as the cosine similarity of their embedding vectors:

$$proximity(e_a, e_b) = cosine\text{-}similarity(\vec{e_a}, \vec{e_b}) = \frac{\vec{e_a} \cdot \vec{e_b}}{|\vec{e_a}| \cdot |\vec{e_b}|} \quad . \tag{15}$$

Finally, we calculate the AUC metric on all sampled edge pairs with their labels (true local edge proximity, i.e., 0 or 1) and their scores (proximity defined on the cosine similarity of edge embeddings). A higher AUC value means the corresponding embedding method preserves the local edge proximity better.

Figure 9(a) demonstrates that the AUC values corresponding to our method **edge2vec** are the highest on both Epinions-sign and Wikipedia. However, the other four methods perform not as well in this experiment. Thus, we think the result verifies the advantage of **edge2vec** to preserve the local edge proximity.

5.5.2 The global edge proximity preservation. For the global edge proximity, we also sample 10,000 edge pairs for each data sets, but without any limitations. For each edge pair, we calculate its global edge proximity according to Definition 3 with 2-step neighborhood vector and cosine similarity,



(a) The effect of k w.r.t. the link prediction task. (b) The effect of k w.r.t. the social tie sign prediction task.

Fig. 10. The effect of parameter k.

and we calculate its proximity after embedding with Eq. 15. Then we get two ranks w.r.t. the edge pairs according to their global edge proximity and proximity after embedding, respectively. We use Kendall rank correlation coefficient [14] τ to measure the similarity of the two ranks:

$$\tau = \frac{n_c - n_d}{n(n-1)/2} \quad , \tag{16}$$

where n_c is the number of concordant pairs, n_d is the number of discordant pairs, and n is the number of elements in the two ranks. Concordant (discordant) pair is the ordered pair of elements, which has the same (opposite) order in both ranks. The value of τ is between -1 and 1, and the higher τ is, the more similar the two ranks are. We would like the coefficient τ of the two ranks of the edge pairs be high, which means the global edge proximity is preserved well.

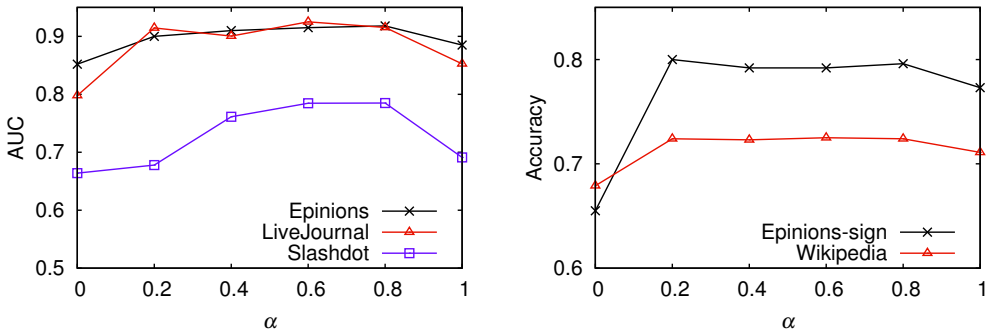
The experimental result shown in Figure 9(b) shows that the value of τ w.r.t. **edge2vec** is significantly higher than those w.r.t. other embedding approaches, especially on the Wikipedia data set (the τ value w.r.t. **edge2vec** is more than 0.4 while the τ values w.r.t. some baselines are below 0.1).

We can see that **edge2vec** can preserve both the local and global edge proximity better than the baseline approaches. We think the essential reason is that **edge2vec** is designed for preserving the edge proximity with the help of its deep neural network, but others are proposed for embedding nodes and do not consider the edge proximity preservation at all.

5.6 Parameter sensitivity

In this subsection, we discuss the effect of two important parameters in **edge2vec**, i.e., k and α. We evaluate how these two parameters affect the results for link prediction and social tie sign prediction tasks.

5.6.1 The maximum length corresponding to calculating neighborhood relationships. k is the maximum length of steps when we calculate the neighborhood vectors, which are able to capture the global structure information of a given edge. A larger k value means that more global structure information is captured by the neighborhood vectors. Figure 10(a) shows that in the link prediction task on the four unsigned data sets, the performance of **edge2vec** gets better with the increase of the k value when $k <= 3$, and when the value of k increases to 4, its performance becomes worse. The results of the social tie sign prediction task on the two signed network (Figure 10(b)) are



(a) The effect of α w.r.t. the link prediction task. (b) The effect of α w.r.t. the social tie sign prediction task.

Fig. 11. The effect of parameter α.

similar with a small difference, i.e. the inflection point appears when $k = 2$. It is because that when the value of k is too large, the sparsity of neighborhood vectors is broken and some interference information is introduced.

5.6.2 The weights of local and global proximity. The parameter $\alpha \in [0, 1]$ balances the weights of the global and the local proximity in Eq. 11. When $\alpha = 0$, only the local proximity is considered. The global proximity affects more on the embedding result with the increase of α. When the α value reaches 1, **edge2vec** totally relies on the global proximity, and it equals to an autoencoder model without negative sampling. Figure 11 shows how the value of α affects the performance. When $\alpha = 0$ or 1, the result is relatively poor. If both the global proximity and the local proximity are taken into consideration, especially when $\alpha = 0.6$ or 0.8 in Figure 11(a) and $\alpha = 0.2$ in Figure 11(b), we can see the performance is much better. It demonstrates that both the global proximity and the local proximity in **edge2vec** are effective.

6 CONCLUSION AND FUTURE WORK

From the actual demand of social network analysis, we propose a novel graph embedding method called **edge2vec**. It can embed a social network into a low-dimensional space through mapping the edges in the network to representation vectors. We combine the deep autoencoder model and the Skip-gram model together in our proposed method. **edge2vec** is able to preserve the global and the local structure information of a social network. The experimental results show that the representation vectors for edges obtained from **edge2vec** are much better than those obtained from the existing node-based embedding methods in an indirect way.

This paper is the first step of edge-based graph embedding study, and we can see many directions for future work. (1) One is to incorporate content information into **edge2vec** to improve the embedding result. Since the content information varies on different social networks, a general approach needs to be designed carefully. (2) Like most existing graph embedding methods, **edge2vec** is an unsupervised model. Thus, in supervised learning tasks for edges, e.g. social tie sign prediction, the label information is only used in the classification period but not in the embedding period. We can study the supervised variant of **edge2vec**, which is able to utilize the label information of edges when embedding the network. (3) As the time complexity of **edge2vec** is high duo to the deep autoencoder, we will try to improve it to scale at large networks with local/global edge proximity preserved. (4) Network analysis for a specific task (e.g. link prediction) based on edge embedding is

promising. We will study how to embed edges to get the best performance in a specific task but not for general purposes.

REFERENCES

[1] Sami Abu-El-Haija, Bryan Perozzi, and Rami Al-Rfou. 2017. Learning Edge Representations via Low-Rank Asymmetric Projections. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017*. 1787–1796. https://doi.org/10.1145/3132847.3132959

[2] Mikhail Belkin and Partha Niyogi. 2002. Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering. In *Advances in Neural Information Processing Systems 14*, T. G. Dietterich, S. Becker, and Z. Ghahramani (Eds.). MIT Press, 585–591. http://papers.nips.cc/paper/1961-laplacian-eigenmaps-and-spectral-techniques-for-embedding-and-clustering.pdf

[3] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 2787–2795. http://papers.nips.cc/paper/5071-translating-embeddings-for-modeling-multi-relational-data.pdf

[4] Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning Structured Embeddings of Knowledge Bases. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011*. http://www.aaai.org/ocs/index.php/AAAI/AAAI11/paper/view/3659

[5] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. GraRep: Learning Graph Representations with Global Structural Information. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM 2015, Melbourne, VIC, Australia, October 19 - 23, 2015*. 891–900. https://doi.org/10.1145/2806416.2806512

[6] Kai-Yang Chiang, Cho-Jui Hsieh, Nagarajan Natarajan, Inderjit S. Dhillon, and Ambuj Tewari. 2014. Prediction and clustering in signed networks: a local to global perspective. *Journal of Machine Learning Research* 15, 1 (2014), 1177–1213. http://dl.acm.org/citation.cfm?id=2638573

[7] Daniel M. Dunlavy, Tamara G. Kolda, and Evrim Acar. 2011. Temporal Link Prediction Using Matrix and Tensor Factorizations. *ACM Trans. Knowl. Discov. Data* 5, 2, Article 10 (Feb. 2011), 27 pages. https://doi.org/10.1145/1921632.1921636

[8] Eric Gilbert and Karrie Karahalios. 2009. Predicting tie strength with social media. In *Proceedings of the 27th International Conference on Human Factors in Computing Systems, CHI 2009, Boston, MA, USA, April 4-9, 2009*. 211–220. https://doi.org/10.1145/1518701.1518736

[9] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. 855–864. https://doi.org/10.1145/2939672.2939754

[10] Aditya Grover, Aaron Zweig, and Stefano Ermon. 2019. Graphite: Iterative Generative Modeling of Graphs. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA (Proceedings of Machine Learning Research)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.), Vol. 97. PMLR, 2434–2444. http://proceedings.mlr.press/v97/grover19a.html

[11] John Hannon, Mike Bennett, and Barry Smyth. 2010. Recommending twitter users to follow using content and collaborative filtering approaches. In *Proceedings of the 2010 ACM Conference on Recommender Systems, RecSys 2010, Barcelona, Spain, September 26-30, 2010*. 199–206. https://doi.org/10.1145/1864708.1864746

[12] Frank Harary and Robert Z Norman. 1960. Some properties of line digraphs. *Rendiconti del Circolo Matematico di Palermo* 9, 2 (1960), 161–168.

[13] Indika Kahanda and Jennifer Neville. 2009. Using Transactional Information to Predict Link Strength in Online Social Networks. In *Proceedings of the Third International Conference on Weblogs and Social Media, ICWSM 2009, San Jose, California, USA, May 17-20, 2009*. http://aaai.org/ocs/index.php/ICWSM/09/paper/view/213

[14] Maurice G Kendall. 1938. A new measure of rank correlation. *Biometrika* 30, 1/2 (1938), 81–93.

[15] Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

[16] Thomas N Kipf and Max Welling. 2016. Variational Graph Auto-Encoders. *NIPS Workshop on Bayesian Deep Learning* (2016).

[17] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

[18] Risi Kondor and John D. Lafferty. 2002. Diffusion Kernels on Graphs and Other Discrete Input Spaces. In *Machine Learning, Proceedings of the Nineteenth International Conference (ICML 2002), University of New South Wales, Sydney, Australia, July 8-12, 2002*. 315–322.

[19] Jure Leskovec, Daniel P. Huttenlocher, and Jon M. Kleinberg. 2010. Predicting positive and negative links in online social networks. In *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*. 641–650. https://doi.org/10.1145/1772690.1772756

[20] Juzheng Li, Jun Zhu, and Bo Zhang. 2016. Discriminative Deep Random Walk for Network Classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. 1004–1013. http://aclweb.org/anthology/P/P16/P16-1095.pdf

[21] Rong-Hua Li, Jeffrey Xu Yu, and Jianquan Liu. 2011. Link prediction: the power of maximal entropy random walk. In *Proceedings of the 20th ACM Conference on Information and Knowledge Management, CIKM 2011, Glasgow, United Kingdom, October 24-28, 2011*. 1147–1156. https://doi.org/10.1145/2063576.2063741

[22] David Liben-Nowell and Jon M. Kleinberg. 2003. The link prediction problem for social networks. In *Proceedings of the 2003 ACM CIKM International Conference on Information and Knowledge Management, New Orleans, Louisiana, USA, November 2-8, 2003*. 556–559. https://doi.org/10.1145/956863.956972

[23] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*. 2181–2187. http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9571

[24] Masoud Makrehchi. 2011. Social link recommendation by learning hidden topics. In *Proceedings of the 2011 ACM Conference on Recommender Systems, RecSys 2011, Chicago, IL, USA, October 23-27, 2011*. 189–196. https://doi.org/10.1145/2043932.2043968

[25] Aditya Krishna Menon and Charles Elkan. 2011. Link Prediction via Matrix Factorization. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2011, Athens, Greece, September 5-9, 2011, Proceedings, Part II*. 437–452. https://doi.org/10.1007/978-3-642-23783-6_28

[26] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 3111–3119. http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf

[27] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: online learning of social representations. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*. 701–710. https://doi.org/10.1145/2623330.2623732

[28] Sam T. Roweis and Lawrence K. Saul. 2000. Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science* 290, 5500 (2000), 2323–2326. https://doi.org/10.1126/science.290.5500.2323 arXiv:http://science.sciencemag.org/content/290/5500/2323.full.pdf

[29] Ruslan Salakhutdinov and Geoffrey E. Hinton. 2009. Semantic hashing. *Int. J. Approx. Reasoning* 50, 7 (2009), 969–978. https://doi.org/10.1016/j.ijar.2008.11.006

[30] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale Information Network Embedding. In *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015*. 1067–1077. https://doi.org/10.1145/2736277.2741093

[31] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. 2000. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science* 290, 5500 (2000), 2319–2323. https://doi.org/10.1126/science.290.5500.2319 arXiv:http://science.sciencemag.org/content/290/5500/2319.full.pdf

[32] Fei Tian, Bin Gao, Qing Cui, Enhong Chen, and Tie-Yan Liu. 2014. Learning Deep Representations for Graph Clustering. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*. 1293–1299. http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8527

[33] Cunchao Tu, Weicheng Zhang, Zhiyuan Liu, and Maosong Sun. 2016. Max-Margin DeepWalk: Discriminative Learning of Network Representation. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*. 3889–3895. http://www.ijcai.org/Abstract/16/547

[34] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural Deep Network Embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. 1225–1234. https://doi.org/10.1145/2939672.2939753

[35] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. 2017. Community Preserving Network Embedding. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*. 203–209. http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14589

[36] Zheng Wang, Xiaojun Ye, Chaokun Wang, Yuexin Wu, Changping Wang, and Kaiwen Liang. 2018. RSDNE: Exploring Relaxed Similarity and Dissimilarity from Completely-Imbalanced Labels for Network Embedding. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, February 2-7, 2018, New Orleans, Louisiana, USA*. 475–482. https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16062

[37] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge Graph Embedding by Translating on Hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec*

City, Québec, Canada. 1112–1119. http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8531

[38] Rongjing Xiang, Jennifer Neville, and Monica Rogati. 2010. Modeling relationship strength in online social networks. In *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*. 981–990. https://doi.org/10.1145/1772690.1772790

[39] Linchuan Xu, Xiaokai Wei, Jiannong Cao, and Philip S. Yu. 2017. Embedding of Embedding (EOE): Joint Embedding for Coupled Heterogeneous Networks. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM 2017, Cambridge, United Kingdom, February 6-10, 2017*. 741–749. http://dl.acm.org/citation.cfm?id=3018723

[40] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y. Chang. 2015. Network Representation Learning with Rich Text Information. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*. 2111–2117. http://ijcai.org/Abstract/15/299

[41] Liang Yang, Xiaochun Cao, Dongxiao He, Chuan Wang, Xiao Wang, and Weixiong Zhang. 2016. Modularity Based Community Detection with Deep Learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*. 2252–2258. http://www.ijcai.org/Abstract/16/321

[42] Shuang-Hong Yang, Alexander J. Smola, Bo Long, Hongyuan Zha, and Yi Chang. 2012. Friend or frenemy?: predicting signed ties in social networks. In *The 35th International ACM SIGIR conference on research and development in Information Retrieval, SIGIR '12, Portland, OR, USA, August 12-16, 2012*. 555–564. https://doi.org/10.1145/2348283.2348359

[43] Jihang Ye, Hong Cheng, Zhe Zhu, and Minghua Chen. 2013. Predicting positive and negative links in signed social networks by transfer learning. In *Proceedings of the 22nd international conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 1477–1488.

[44] Ching-man Au Yeung and Tomoharu Iwata. 2011. Strength of social influence in trust networks in product review sites. In *Proceedings of the Forth International Conference on Web Search and Web Data Mining, WSDM 2011, Hong Kong, China, February 9-12, 2011*. 495–504. https://doi.org/10.1145/1935826.1935899

[45] Zhijun Yin, Manish Gupta, Tim Weninger, and Jiawei Han. 2010. LINKREC: a unified framework for link recommendation with user attributes and graph structure. In *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*. 1211–1212. https://doi.org/10.1145/1772690.1772879

[46] Jun Zhang, Chaokun Wang, Jianmin Wang, Jeffrey Xu Yu, Jun Chen, and Changping Wang. 2016. Inferring Directions of Undirected Social Ties. *IEEE Trans. Knowl. Data Eng.* 28, 12 (2016), 3276–3292. https://doi.org/10.1109/TKDE.2016.2605081

[47] Jun Zhang, Chaokun Wang, Philip S. Yu, and Jianmin Wang. 2013. Learning latent friendship propagation networks with interest awareness for link prediction. In *The 36th International ACM SIGIR conference on research and development in Information Retrieval, SIGIR '13, Dublin, Ireland - July 28 - August 01, 2013*. 63–72. https://doi.org/10.1145/2484028.2484029

[48] Jinfeng Zhuang, Tao Mei, Steven C. H. Hoi, Xian-Sheng Hua, and Shipeng Li. 2011. Modeling social strength in social media community via kernel-based learning. In *Proceedings of the 19th International Conference on Multimedia 2011, Scottsdale, AZ, USA, November 28 - December 1, 2011*. 113–122. https://doi.org/10.1145/2072298.2072315