LINE: Large-scale Information Network Embedding

Jian Tang[1], Meng Qu[2]*, Mingzhe Wang[2], Ming Zhang[2], Jun Yan[1], Qiaozhu Mei[3]
[1]Microsoft Research Asia, {jiatang, junyan}@microsoft.com
[2]School of EECS, Peking University, {mnqu, wangmingzhe, mzhang_cs}@pku.edu.cn
[3]School of Information, University of Michigan, qmei@umich.edu

ABSTRACT

This paper studies the problem of embedding very large information networks into low-dimensional vector spaces, which is useful in many tasks such as visualization, node classification, and link prediction. Most existing graph embedding methods do not scale for real world information networks which usually contain millions of nodes. In this paper, we propose a novel network embedding method called the "LINE," which is suitable for arbitrary types of information networks: undirected, directed, and/or weighted. The method optimizes a carefully designed objective function that preserves both the local and global network structures. An edge-sampling algorithm is proposed that addresses the limitation of the classical stochastic gradient descent and improves both the effectiveness and the efficiency of the inference. Empirical experiments prove the effectiveness of the LINE on a variety of real-world information networks, including language networks, social networks, and citation networks. The algorithm is very efficient, which is able to learn the embedding of a network with millions of vertices and billions of edges in a few hours on a typical single machine. The source code of the LINE is available online.[1]

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms, Experimentation

Keywords

information network embedding; scalability; feature learning; dimension reduction

*This work was done when the second author was an intern at Microsoft Research Asia.

[1]`https://github.com/tangjianpku/LINE`

WWW 2015, May 18–22, 2015, Florence, Italy.
ACM 978-1-4503-3469-3/15/05.
http://dx.doi.org/10.1145/2736277.2741093.

1. INTRODUCTION

Information networks are ubiquitous in the real world with examples such as airline networks, publication networks, social and communication networks, and the World Wide Web. The size of these information networks ranges from hundreds of nodes to millions and billions of nodes. Analyzing large information networks has been attracting increasing attention in both academia and industry. This paper studies the problem of embedding information networks into low-dimensional spaces, in which every vertex is represented as a low-dimensional vector. Such a low-dimensional embedding is very useful in a variety of applications such as visualization [21], node classification [3], link prediction [10], and recommendation [23].

Various methods of graph embedding have been proposed in the machine learning literature (e.g., [4, 20, 2]). They generally perform well on smaller networks. The problem becomes much more challenging when a real world information network is concerned, which typically contains millions of nodes and billions of edges. For example, the Twitter followee-follower network contains 175 million active users and around twenty billion edges in 2012 [14]. Most existing graph embedding algorithms do not scale for networks of this size. For example, the time complexity of classical graph embedding algorithms such as MDS [4], IsoMap [20], Laplacian eigenmap [2] are at least quadratic to the number of vertices, which is too expensive for networks with millions of nodes. Although a few very recent studies approach the embedding of large-scale networks, these methods either use an indirect approach that is not designed for networks (e.g., [1]) or lack a clear objective function tailored for network embedding (e.g., [16]). We anticipate that a new model with a carefully designed objective function that preserves properties of the graph and an efficient optimization technique should effectively find the embedding of millions of nodes.

In this paper, we propose such a network embedding model called the "LINE," which is able to scale to very large, arbitrary types of networks: undirected, directed and/or weighted. The model optimizes an objective which preserves both the local and global network structures. Naturally, the local structures are represented by the observed links in the networks, which capture the *first-order* proximity between the vertices. Most existing graph embedding algorithms are designed to preserve this *first-order* proximity, e.g., IsoMap [20] and Laplacian eigenmap [2], even if they do not scale. We observe that in a real-world network many (if not the majority of) legitimate links are actually not observed. In other

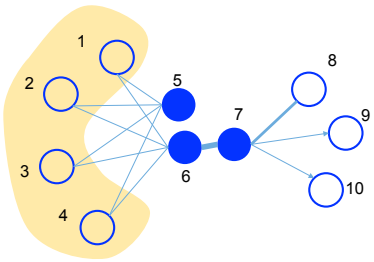


Figure 1: A toy example of information network. Edges can be undirected, directed, and/or weighted. Vertex 6 and 7 should be placed closely in the low-dimensional space as they are connected through a strong tie. Vertex 5 and 6 should also be placed closely as they share similar neighbors.

words, the observed *first-order* proximity in the real world data is not sufficient for preserving the global network structures. As a complement, we explore the *second-order* proximity between the vertices, which is not determined through the observed tie strength but through the shared neighborhood structures of the vertices. The general notion of the *second-order* proximity can be interpreted as nodes with shared neighbors being likely to be similar. Such an intuition can be found in the theories of sociology and linguistics. For example, "the degree of overlap of two people's friendship networks correlates with the strength of ties between them," in a social network [6]; and "You shall know a word by the company it keeps" (Firth, J. R. 1957:11) in text corpora [5]. Indeed, people who share many common friends are likely to share the same interest and become friends, and words that are used together with many similar words are likely to have similar meanings.

Fig. 1 presents an illustrative example. As the weight of the edge between vertex 6 and 7 is large, i.e., 6 and 7 have a high *first-order* proximity, they should be represented closely to each other in the embedded space. On the other hand, though there is no link between vertex 5 and 6, they share many common neighbors, i.e., they have a high *second-order* proximity and therefore should also be represented closely to each other. We expect that the consideration of the *second-order* proximity effectively complements the sparsity of the *first-order* proximity and better preserves the global structure of the network. In this paper, we will present carefully designed objectives that preserve the first-order and the second-order proximities.

Even if a sound objective is found, optimizing it for a very large network is challenging. One approach that attracts attention in recent years is using the stochastic gradient descent for the optimization. However, we show that directly deploying the stochastic gradient descent is problematic for real world information networks. This is because in many networks, edges are weighted and the weights usually present a high variance. Consider a word co-occurrence network, in which the weights (co-occurrences) of word pairs may range from one to hundreds of thousands. These weights of the edges will be multiplied into the gradients, resulting in the explosion of the gradients and thus compromise the performance. To address this, we propose a novel edge-sampling method, which improves both the effectiveness and efficiency of the inference. We sample the edges with the probabilities proportional to their weights, and then treat the sampled edges as binary edges for model updating. With this sampling process, the objective function remains the same and the weights of the edges no longer affect the gradients.

The LINE is very general, which works well for directed or undirected, weighted or unweighted graphs. We evaluate the performance of the LINE with various real-world information networks, including language networks, social networks, and citation networks. The effectiveness of the learned embeddings is evaluated within multiple data mining tasks, including word analogy, text classification, and node classification. The results suggest that the LINE model outperforms other competitive baselines in terms of both effectiveness and efficiency. It is able to learn the embedding of a network with millions of nodes and billions of edges in a few hours on a single machine.

To summarize, we make the following contributions:

- We propose a novel network embedding model called the "LINE," which suits arbitrary types of information networks and easily scales to millions of nodes. It has a carefully designed objective function that preserves both the first-order and second-order proximities.
- We propose an edge-sampling algorithm for optimizing the objective. The algorithm tackles the limitation of the classical stochastic gradient decent and improves the effectiveness and efficiency of the inference.
- We conduct extensive experiments on real-world information networks. Experimental results prove the effectiveness and efficiency of the proposed LINE model.

Organization. The rest of this paper is organized as follows. Section 2 summarizes the related work. Section 3 formally defines the problem of large-scale information network embedding. Section 4 introduces the LINE model in details. Section 5 presents the experimental results. Finally we conclude in Section 6.

2. RELATED WORK

Our work is related to classical methods of graph embedding or dimension reduction in general, such as multi-dimensional scaling (MDS) [4], IsoMap [20], LLE [18] and Laplacian Eigenmap [2]. These approaches typically first construct the affinity graph using the feature vectors of the data points, e.g., the K-nearest neighbor graph of data, and then embed the affinity graph [22] into a low dimensional space. However, these algorithms usually rely on solving the leading eigenvectors of the affinity matrices, the complexity of which is at least quadratic to the number of nodes, making them inefficient to handle large-scale networks.

Among the most recent literature is a technique called graph factorization [1]. It finds the low-dimensional embedding of a large graph through matrix factorization, which is optimized using stochastic gradient descent. This is possible because a graph can be represented as an affinity matrix. However, the objective of matrix factorization is not designed for networks, therefore does not necessarily preserve the global network structure. Intuitively, graph factorization expects nodes with higher first-order proximity are represented closely. Instead, the LINE model uses an objective that is particularly designed for networks, which preserves both the *first-order* and the *second-order* proximities. Practically, the graph factorization method only applies to undirected graphs while the proposed model is applicable for both undirected and directed graphs.

The most recent work related with ours is DeepWalk [16], which deploys a truncated random walk for social network embedding. Although empirically effective, the DeepWalk does not provide a clear objective that articulates what network properties are preserved. Intuitively, DeepWalk expects nodes with higher *second-order* proximity yield similar low-dimensional representations, while the LINE preserves both *first-order* and *second-order* proximities. DeepWalk uses random walks to expand the neighborhood of a vertex, which is analogical to a depth-first search. We use a breadth-first search strategy, which is a more reasonable approach to the *second-order* proximity. Practically, DeepWalk only applies to unweighted networks, while our model is applicable for networks with both weighted and unweighted edges.

In Section 5, we empirically compare the proposed model with these methods using various real world networks.

3. PROBLEM DEFINITION

We formally define the problem of large-scale information network embedding using *first-order* and *second-order* proximities. We first define an information network as follows:

DEFINITION 1. ***(Information Network)*** *An* ***information network*** *is defined as $G = (V, E)$, where V is the set of vertices, each representing a data object and E is the set of edges between the vertices, each representing a relationship between two data objects. Each edge $e \in E$ is an ordered pair $e = (u, v)$ and is associated with a weight $w_{uv} > 0$, which indicates the strength of the relation. If G is undirected, we have $(u, v) \equiv (v, u)$ and $w_{uv} \equiv w_{vu}$; if G is directed, we have $(u, v) \not\equiv (v, u)$ and $w_{uv} \not\equiv w_{vu}$.*

In practice, information networks can be either directed (e.g., citation networks) or undirected (e.g., social network of users in Facebook). The weights of the edges can be either binary or take any real value. Note that while negative edge weights are possible, in this study we only consider non-negative weights. For example, in citation networks and social networks, w_{uv} takes binary values; in co-occurrence networks between different objects, w_{uv} can take any non-negative value. The weights of the edges in some networks may diverge as some objects co-occur many times while others may just co-occur a few times.

Embedding an information network into a low-dimensional space is useful in a variety of applications. To conduct the embedding, the network structures must be preserved. The first intuition is that the local network structure, i.e., the local pairwise proximity between the vertices, must be preserved. We define the local network structures as the *first-order* proximity between the vertices:

DEFINITION 2. ***(First-order Proximity)*** *The* ***first-order*** *proximity in a network is the* ***local*** *pairwise proximity between two vertices. For each pair of vertices linked by an edge (u, v), the weight on that edge, w_{uv}, indicates the first-order proximity between u and v. If no edge is observed between u and v, their* ***first-order*** *proximity is 0.*

The *first-order* proximity usually implies the similarity of two nodes in a real-world network. For example, people who are friends with each other in a social network tend to share similar interests; pages linking to each other in World Wide Web tend to talk about similar topics. Because of this importance, many existing graph embedding algorithms such as IsoMap, LLE, Laplacian eigenmap, and graph factorization have the objective to preserve the *first-order* proximity.

However, in a real world information network, the links observed are only a small proportion, with many others missing [10]. A pair of nodes on a missing link has a zero *first-order* proximity, even though they are intrinsically very similar to each other. Therefore, *first-order* proximity alone is not sufficient for preserving the network structures, and it is important to seek an alternative notion of proximity that addresses the problem of sparsity. A natural intuition is that vertices that share similar neighbors tend to be similar to each other. For example, in social networks, people who share similar friends tend to have similar interests and thus become friends; in word co-occurrence networks, words that always co-occur with the same set of words tend to have similar meanings. We therefore define the *second-order* proximity, which complements the first-order proximity and preserves the network structure.

DEFINITION 3. ***(Second-order Proximity)*** *The* ***second-order*** *proximity between a pair of vertices (u, v) in a network is the similarity between their neighborhood network structures. Mathematically, let $p_u = (w_{u,1}, \ldots, w_{u,|V|})$ denote the first-order proximity of u with all the other vertices, then the* second-order *proximity between u and v is determined by the similarity between p_u and p_v. If no vertex is linked from/to both u and v, the* ***second-order*** *proximity between u and v is 0.*

We investigate both *first-order* and *second-order* proximity for network embedding, which is defined as follows.

DEFINITION 4. ***(Large-scale Information Network Embedding)*** *Given a large network $G = (V, E)$, the problem of* ***Large-scale Information Network Embedding*** *aims to represent each vertex $v \in V$ into a low-dimensional space R^d, i.e., learning a function $f_G : V \to R^d$, where $d \ll |V|$. In the space R^d, both the* ***first-order*** *proximity and the* ***second-order*** *proximity between the vertices are preserved.*

Next, we introduce a large-scale network embedding model that preserves both *first-* and *second-order* proximities.

4. LINE: LARGE-SCALE INFORMATION NETWORK EMBEDDING

A desirable embedding model for real world information networks must satisfy several requirements: first, it must be able to preserve both the *first-order* proximity and the *second-order* proximity between the vertices; second, it must scale for very large networks, say millions of vertices and billions of edges; third, it can deal with networks with arbitrary types of edges: directed, undirected and/or weighted. In this section, we present a novel network embedding model called the "LINE," which satisfies all the three requirements.

4.1 Model Description

We describe the LINE model to preserve the *first-order* proximity and *second-order* proximity separately, and then introduce a simple way to combine the two proximity.

4.1.1 LINE with First-order Proximity

The *first-order* proximity refers to the local pairwise proximity between the vertices in the network. To model the

first-order proximity, for each undirected edge (i, j), we define the joint probability between vertex v_i and v_j as follows:

$$p_1(v_i, v_j) = \frac{1}{1 + \exp(-\vec{u}_i^T \cdot \vec{u}_j)}, \tag{1}$$

where $\vec{u}_i \in R^d$ is the low-dimensional vector representation of vertex v_i. Eqn. (1) defines a distribution $p(\cdot, \cdot)$ over the space $V \times V$, and its empirical probability can be defined as $\hat{p}_1(i, j) = \frac{w_{ij}}{W}$, where $W = \sum_{(i,j)\in E} w_{ij}$. To preserve the first-order proximity, a straightforward way is to minimize the following objective function:

$$O_1 = d(\hat{p}_1(\cdot, \cdot), p_1(\cdot, \cdot)), \tag{2}$$

where $d(\cdot, \cdot)$ is the distance between two distributions. We choose to minimize the KL-divergence of two probability distributions. Replacing $d(\cdot, \cdot)$ with KL-divergence and omitting some constants, we have:

$$O_1 = -\sum_{(i,j)\in E} w_{ij} \log p_1(v_i, v_j), \tag{3}$$

Note that the *first-order* proximity is only applicable for undirected graphs, not for directed graphs. By finding the $\{\vec{u}_i\}_{i=1..|V|}$ that minimize the objective in Eqn. (3), we can represent every vertex in the d-dimensional space.

4.1.2 LINE with Second-order Proximity

The *second-order* proximity is applicable for both directed and undirected graphs. Given a network, without loss of generality, we assume it is directed (an undirected edge can be considered as two directed edges with opposite directions and equal weights). The *second-order* proximity assumes that vertices sharing many connections to other vertices are similar to each other. In this case, each vertex is also treated as a specific "context" and vertices with similar distributions over the "contexts" are assumed to be similar. Therefore, each vertex plays two roles: the vertex itself and a specific "context" of other vertices. We introduce two vectors $\vec{u}_i$ and $\vec{u}'_i$, where $\vec{u}_i$ is the representation of v_i when it is treated as a vertex while $\vec{u}'_i$ is the representation of v_i when it is treated as a specific "context". For each directed edge (i, j), we first define the probability of "context" v_j generated by vertex v_i as:

$$p_2(v_j|v_i) = \frac{\exp(\vec{u}_j'^T \cdot \vec{u}_i)}{\sum_{k=1}^{|V|} \exp(\vec{u}_k'^T \cdot \vec{u}_i)}, \tag{4}$$

where $|V|$ is the number of vertices or "contexts." For each vertex v_i, Eqn. (4) actually defines a conditional distribution $p_2(\cdot|v_i)$ over the contexts, i.e., the entire set of vertices in the network. As mentioned above, the *second-order* proximity assumes that vertices with similar distributions over the contexts are similar to each other. To preserve the *second-order* proximity, we should make the conditional distribution of the contexts $p_2(\cdot|v_i)$ specified by the low-dimensional representation be close to the empirical distribution $\hat{p}_2(\cdot|v_i)$. Therefore, we minimize the following objective function:

$$O_2 = \sum_{i\in V} \lambda_i d(\hat{p}_2(\cdot|v_i), p_2(\cdot|v_i)), \tag{5}$$

where $d(\cdot, \cdot)$ is the distance between two distributions. As the importance of the vertices in the network may be different, we introduce λ_i in the objective function to represent the prestige of vertex i in the network, which can be measured by the degree or estimated through algorithms such as PageRank [15]. The empirical distribution $\hat{p}_2(\cdot|v_i)$ is defined as $\hat{p}_2(v_j|v_i) = \frac{w_{ij}}{d_i}$, where w_{ij} is the weight of the edge (i, j) and d_i is the out-degree of vertex i, i.e. $d_i = \sum_{k\in N(i)} w_{ik}$, where $N(i)$ is the set of out-neighbors of v_i. In this paper, for simplicity we set λ_i as the degree of vertex i, i.e., $\lambda_i = d_i$, and here we also adopt KL-divergence as the distance function. Replacing $d(\cdot, \cdot)$ with KL-divergence, setting $\lambda_i = d_i$ and omitting some constants, we have:

$$O_2 = -\sum_{(i,j)\in E} w_{ij} \log p_2(v_j|v_i). \tag{6}$$

By learning $\{\vec{u}_i\}_{i=1..|V|}$ and $\{\vec{u}'_i\}_{i=1..|V|}$ that minimize this objective, we are able to represent every vertex v_i with a d-dimensional vector $\vec{u}_i$.

4.1.3 Combining first-order and second-order proximities

To embed the networks by preserving both the *first-order* and *second-order* proximity, a simple and effective way we find in practice is to train the LINE model which preserves the *first-order* proximity and *second-order* proximity separately and then concatenate the embeddings trained by the two methods for each vertex. A more principled way to combine the two proximity is to jointly train the objective function (3) and (6), which we leave as future work.

4.2 Model Optimization

Optimizing objective (6) is computationally expensive, which requires the summation over the entire set of vertices when calculating the conditional probability $p_2(\cdot|v_i)$. To address this problem, we adopt the approach of negative sampling proposed in [13], which samples multiple negative edges according to some noisy distribution for each edge (i, j). More specifically, it specifies the following objective function for each edge (i, j):

$$\log \sigma(\vec{u}_j'^T \cdot \vec{u}_i) + \sum_{i=1}^{K} E_{v_n\sim P_n(v)}[\log \sigma(-\vec{u}_n'^T \cdot \vec{u}_i)], \tag{7}$$

where $\sigma(x) = 1/(1 + \exp(-x))$ is the sigmoid function. The first term models the observed edges, the second term models the negative edges drawn from the noise distribution and K is the number of negative edges. We set $P_n(v) \propto {d_v}^{3/4}$, which is proposed in [13] and d_v is the out degree of vertex v.

For the objective function (3), there exists a trivial solution: $u_{ik} = \infty$, for i=1, . . . , $|V|$ and $k = 1, \ldots, d$. To avoid the trivial solution, we can still utilize the negative sampling approach (7) by just changing $\vec{u}_j'^T$ to $\vec{u}_j^T$.

We adopt the asynchronous stochastic gradient algorithm (ASGD) [17] for optimizing Eqn. (7). In each step, the ASGD algorithm samples a mini-batch of edges and then updates the model parameters. If an edge (i, j) is sampled, the gradient w.r.t. the embedding vector $\vec{u}_i$ of vertex i will be calculated as:

$$\frac{\partial O_2}{\partial \vec{u}_i} = w_{ij} \cdot \frac{\partial \log p_2(v_j|v_i)}{\partial \vec{u}_i} \tag{8}$$

Note that the gradient will be multiplied by the weight of the edge. This will become problematic when the weights

of edges have a high variance. For example, in a word co-occurrence network, some words co-occur many times (e.g., tens of thousands) while some words co-occur only a few times. In such networks, the scales of the gradients diverge and it is very hard to find a good learning rate. If we select a large learning rate according to the edges with small weights, the gradients on edges with large weights will explode while the gradients will become too small if we select the learning rate according to the edges with large weights.

4.2.1 Optimization via Edge Sampling

The intuition in solving the above problem is that if the weights of all the edges are equal (e.g., network with binary edges), then there will be no problem of choosing an appropriate learning rate. A simple treatment is thus to unfold a weighted edge into multiple *binary* edges, e.g., an edge with weight w is unfolded into w *binary* edges. This will solve the problem but will significantly increase the memory requirement, especially when the weights of the edges are very large. To resolve this, one can sample from the original edges and treat the sampled edges as *binary* edges, with the sampling probabilities proportional to the original edge weights. With this edge-sampling treatment, the overall objective function remains the same. Therefore, the problem boils down to how to sample the edges according to their weights.

Let $W = (w_1, w_2, \ldots, w_{|E|})$ denote the sequence of the weights of the edges. A simple approach is to calculate the sum of the weights $w_{sum} = \sum_{i=1}^{|E|} w_i$ first, and then to sample a random value within the range of $[0, w_{sum}]$ to see which interval $[\sum_{j=0}^{i-1} w_j, \sum_{j=0}^{i} w_j]$ the random value belongs to. This approach takes $O(|E|)$ time for drawing a sample, which is very time consuming when the number of edges $|E|$ is large. In this paper, we use the alias table method [9] to draw a sample according to the weights of the edges, which takes only $O(1)$ time when repeatedly drawing samples from the same discrete distribution.

Sampling an edge from the alias table takes constant time, i.e. $O(1)$, and optimization with negative sampling takes $O(d(K+1))$ time, where K is the number of negative samples. Therefore, overall each step takes $O(dK)$ time. In practice, we find that the number of steps used for optimization is usually proportional to the number of edges $O(|E|)$. Therefore, the overall time complexity of the LINE is $O(dK|E|)$, which is linear to the number of edges $|E|$, and does not depend on the number of vertices $|V|$. The edge sampling treatment improves the effectiveness of the stochastic gradient descent without compromising the efficiency.

4.3 Discussion

We discuss several practical issues of the LINE model.

Low degree vertices. One practical issue is how to accurately embed vertices with small degrees. As the number of neighbors of such a node is very small, it is very hard to accurately infer its representation, especially with the *second-order* proximity based methods which heavily rely on the number of "contexts." An intuitive solution to this is expanding the neighbors of those vertices by adding higher order neighbors, such as neighbors of neighbors. In this paper, we only consider adding second-order neighbors, i.e., neighbors of neighbors, to each vertex. The weight between vertex i and its second-order neighbor j is measured as

$$w_{ij} = \sum_{k \in N(i)} w_{ik} \frac{w_{kj}}{d_k}. \quad (9)$$

In practice, one can only add a subset of vertices $\{j\}$ which have the largest proximity w_{ij} with the low degree vertex i.

New vertices. Another practical issue is how to find the representation of newly arrived vertices. For a new vertex i, if its connections to the existing vertices are known, we can obtain the empirical distribution $\hat{p}_1(\cdot, v_i)$ and $\hat{p}_2(\cdot|v_i)$ over existing vertices. To obtain the embedding of the new vertex, according to the objective function Eqn. (3) or Eqn. (6), a straightforward way is to minimize either one of the following objective functions

$$-\sum_{j \in N(i)} w_{ji} \log p_1(v_j, v_i), \text{ or } -\sum_{j \in N(i)} w_{ji} \log p_2(v_j|v_i), \quad (10)$$

by updating the embedding of the new vertex and keeping the embeddings of existing vertices. If no connections between the new vertex and existing vertices are observed, we must resort to other information, such as the textual information of the vertices, and we leave it as our future work.

5. EXPERIMENTS

In this section, we empirically evaluate the effectiveness and efficiency of the LINE model. We apply the method to several large-scale real-world networks of different types, including a language network, two social networks, and two citation networks.

5.1 Experiment Setup

Data Sets.

(1) LANGUAGE NETWORK. We construct a word co-occurrence network from the entire set of English WIKIPEDIA pages. Words within every 5-word sliding window are considered to be co-occurring with each other. Words with frequency smaller than 5 are filtered out. (2) SOCIAL NETWORKS. We use two social networks: FLICKR and YOUTUBE[2]. The FLICKR network is denser than the YOUTUBE network (the same network as used in DeepWalk [16]). (3) CITATION NETWORKS. Two types of citation networks are used: an author citation network and a paper citation network. We use the DBLP data set [19][3] to construct the citation networks between authors and between papers. The author citation network records the number of papers written by one author and cited by another author. The detailed statistics of these networks are summarized into Table 1. They represent a variety of information networks: directed and undirected, binary and weighted. Each network contains at least half a million nodes and millions of edges, with the largest network containing around two million nodes and a billion edges.

Compared Algorithms.

We compare the LINE model with several existing graph embedding methods that are able to scale up to very large networks. We do not compare with some classical graph embedding algorithms such as MDS, IsoMap, and Laplacian eigenmap, as they cannot handle networks of this scale.

[2]Available at `http://socialnetworks.mpi-sws.org/data-imc2007.html`

[3]Available at `http://arnetminer.org/citation`

Table 1: Statistics of the real-world information networks.

	Language Network	Social Network		Citation Network	
Name	WIKIPEDIA	FLICKR	YOUTUBE	DBLP(AUTHORCITATION)	DBLP(PAPERCITATION)
Type	undirected,weighted	undirected,binary	undirected,binary	dircted,weighted	directed,binary
|V|	1,985,098	1,715,256	1,138,499	524,061	781,109
|E|	1,000,924,086	22,613,981	2,990,443	20,580,238	4,191,677
Avg. degree	504.22	26.37	5.25	78.54	10.73
#Labels	7	5	47	7	7
#train	70,000	75,958	31,703	20,684	10,398

- Graph factorization (GF) [1]. We compare with the matrix factorization techniques for graph factorization. An information network can be represented as an affinity matrix, and is able to represent each vertex with a low-dimensional vector through matrix factorization. Graph factorization is optimized through stochastic gradient descent and is able to handle large networks. It only applies to undirected networks.

- DeepWalk [16]. DeepWalk is an approach recently proposed for social network embedding, which is only applicable for networks with binary edges. For each vertex, truncated random walks starting from the vertex are used to obtain the contextual information, and therefore only *second-order* proximity is utilized.

- LINE-SGD. This is the LINE model introduced in Section 4.1 that optimizes the objective Eqn. (3) or Eqn. (6) directly with stochastic gradient descent. With this approach, the weights of the edges are directly multiplied into the gradients when the edges are sampled for model updating. There are two variants of this approach: LINE-SGD(1st) and LINE-SGD(2nd), which use *first-* and *second-order* proximity respectively.

- LINE. This is the LINE model optimized through the edge-sampling treatment introduced in Section 4.2. In each stochastic gradient step, an edge is sampled with the probability proportional to its weight and then treated as binary for model updating. There are also two variants: LINE(1st) and LINE(2nd). Like the graph factorization, both LINE(1st) and LINE-SGD(1st) only apply to undirected graphs. LINE(2nd) and LINE-SGD(2nd) apply to both undirected and directed graphs.

- LINE (1st+2nd): To utilize both *first-order* and *second-order* proximity, a simple and effective way is to concatenate the vector representations learned by LINE(1st) and LINE(2nd) into a longer vector. After concatenation, the dimensions should be re-weighted to balance the two representations. In a supervised learning task, the weighting of dimensions can be automatically found based on the training data. In an unsupervised task, however, it is more difficult to set the weights. Therefore we only apply LINE (1st+2nd) to the scenario of supervised tasks.

Parameter Settings.

The mini-batch size of the stochastic gradient descent is set as 1 for all the methods. Similar to [13], the learning rate is set with the starting value $\rho_0 = 0.025$ and $\rho_t = \rho_0(1-t/T)$, where T is the total number of mini-batches or edge samples. For fair comparisons, the dimensionality of the embeddings of the language network is set to 200, as used in word embedding [13]. For other networks, the dimension is set as 128 by default, as used in [16]. Other default settings include: the number of negative samples $K = 5$ for LINE and LINE-SGD; the total number of samples $T = 10$ billion for LINE(1st) and LINE(2nd), $T = 20$ billion for GF; window size $win = 10$, walk length $t = 40$, walks per vertex $\gamma = 40$ for DeepWalk. All the embedding vectors are finally normalized by setting $||\vec{w}||_2 = 1$.

5.2 Quantitative Results

5.2.1 Language Network

We start with the results on the language network, which contains two million nodes and a billion edges. Two applications are used to evaluate the effectiveness of the learned embeddings: word analogy [12] and document classification.

Table 2: Results of word analogy on WIKIPEDIA data.

Algorithm	Semantic (%)	Syntactic (%)	Overall (%)	Running time
GF	61.38	44.08	51.93	2.96h
DeepWalk	50.79	37.70	43.65	16.64h
SkipGram	69.14	57.94	63.02	2.82h
LINE-SGD(1st)	9.72	7.48	8.50	3.83h
LINE-SGD(2nd)	20.42	9.56	14.49	3.94h
LINE(1st)	58.08	49.42	53.35	2.44h
LINE(2nd)	**73.79**	**59.72**	**66.10**	2.55h

Word Analogy. This task is introduced by Mikolov et al. [12]. Given a word pair (a, b) and a word c, the task aims to find a word d, such that the relation between c and d is similar to the relation between a and b, or denoted as: $a : b \rightarrow c :?$. For instance, given a word pair ("China","Beijing") and a word "France," the right answer should be "Paris" because "Beijing" is the capital of "China" just as "Paris" is the capital of "France." Given the word embeddings, this task is solved by finding the word d^* whose embedding is closest to the vector $\vec{u}_b - \vec{u}_a + \vec{u}_c$ in terms of cosine proximity, i.e., $d^* = \text{argmax}_d \cos((\vec{u}_b - \vec{u}_a + \vec{u}_c), \vec{u}_d)$. Two categories of word analogy are used in this task: semantic and syntactic.

Table 2 reports the results of word analogy using the embeddings of words learned on the WIKIPEDIA corpora (SkipGram) or the WIKIPEDIA word network (all other methods). For graph factorization, the weight between each pair of words is defined as the logarithm of the number of co-occurrences, which leads to better performance than the original value of co-occurrences. For DeepWalk, different cutoff thresholds are tried to convert the language network into a binary network, and the best performance is achieved when all the edges are kept in the network. We also compare with the state-of-the-art word embedding model SkipGram [12], which learns the word embeddings directly from the original Wikipedia pages and is also implicitly a matrix

Table 3: Results of Wikipedia page classification on WIKIPEDIA data set.

Metric	Algorithm	10%	20%	30%	40%	50%	60%	70%	80%	90%
Micro-F1	GF	79.63	80.51	80.94	81.18	81.38	81.54	81.63	81.71	81.78
	DeepWalk	78.89	79.92	80.41	80.69	80.92	81.08	81.21	81.35	81.42
	SkipGram	79.84	80.82	81.28	81.57	81.71	81.87	81.98	82.05	82.09
	LINE-SGD(1st)	76.03	77.05	77.57	77.85	78.08	78.25	78.39	78.44	78.49
	LINE-SGD(2nd)	74.68	76.53	77.54	78.18	78.63	78.96	79.19	79.40	79.57
	LINE(1st)	79.67	80.55	80.94	81.24	81.40	81.52	81.61	81.69	81.67
	LINE(2nd)	79.93	80.90	81.31	81.63	81.80	81.91	82.00	82.11	82.17
	LINE(1st+2nd)	**81.04****	**82.08****	**82.58****	**82.93****	**83.16****	**83.37****	**83.52****	**83.63****	**83.74****
Macro-F1	GF	79.49	80.39	80.82	81.08	81.26	81.40	81.52	81.61	81.68
	DeepWalk	78.78	79.78	80.30	80.56	80.82	80.97	81.11	81.24	81.32
	SkipGram	79.74	80.71	81.15	81.46	81.63	81.78	81.88	81.98	82.01
	LINE-SGD(1st)	75.85	76.90	77.40	77.71	77.94	78.12	78.24	78.29	78.36
	LINE-SGD(2nd)	74.70	76.45	77.43	78.09	78.53	78.83	79.08	79.29	79.46
	LINE(1st)	79.54	80.44	80.82	81.13	81.29	81.43	81.51	81.60	81.59
	LINE(2nd)	79.82	80.81	81.22	81.52	81.71	81.82	81.92	82.00	82.07
	LINE(1st+2nd)	**80.94****	**81.99****	**82.49****	**82.83****	**83.07****	**83.29****	**83.42****	**83.55****	**83.66****

Significantly outperforms GF at the: ** 0.01 and * 0.05 level, paired t-test.

factorization approach [8]. The window size is set as 5, the same as used for constructing the language network.

We can see that LINE(2nd) outperforms all other methods, including the graph embedding methods and the SkipGram. This indicates that the *second-order* proximity better captures the word semantics compared to the *first-order* proximity. This is not surprising, as a high *second-order* proximity implies that two words can be replaced in the same context, which is a stronger indicator of similar semantics than first-order co-occurrences. It is intriguing that the LINE(2nd) outperforms the state-of-the-art word embedding model trained on the original corpus. The reason may be that a language network better captures the global structure of word co-occurrences than the original word sequences. Among other methods, both graph factorization and LINE(1st) significantly outperform DeepWalk even if DeepWalk explores second-order proximity. This is because DeepWalk has to ignore the weights (i.e., co-occurrences) of the edges, which is very important in a language network. The performance by the LINE models directly optimized with SGD is much worse, because the weights of the edges in the language network diverge, which range from a single digit to tens of thousands, making the learning process suffer. The LINE optimized by the edge-sampling treatment effectively addresses this problem, and performs very well using either *first-order* or *second-order* proximity.

All the models are run on a single machine with 1T memory, 40 CPU cores at 2.0GHZ using 16 threads. Both the LINE(1st) and LINE(2nd) are quite efficient, which take less than 3 hours to process such a network with 2 million nodes and a billion edges. Both are at least 10% faster than graph factorization, and much more efficient than DeepWalk (five times slower). The reason that LINE-SGDs are slightly slower is that a threshold-cutting technique has to be applied to prevent the gradients from exploding.

Document Classification. Another way to evaluate the quality of the word embeddings is to use the word vectors to compute document representation, which can be evaluated with document classification tasks. To obtain document vectors, we choose a very simple approach, taking the average of the word vector representations in that document. This is because we aim to compare the word embeddings with different approaches instead of finding the best method for document embeddings. The readers can find advanced document embedding approaches in [7]. We download the abstracts of Wikipedia pages from `http://downloads.dbpedia.org/3.9/en/long_abstracts_en.nq.bz2` and the categories of these pages from `http://downloads.dbpedia.org/3.9/en/article_categories_en.nq.bz2`. We choose 7 diverse categories for classification including "Arts," "History," "Human," "Mathematics," "Nature," "Technology," and "Sports." For each category, we randomly select 10,000 articles, and articles belonging to multiple categories are discarded. We randomly sample different percentages of the labeled documents for training and use the rest for evaluation. All document vectors are used to train a one-vs-rest logistic regression classifier using the LibLinear package[4]. We report the classification metrics Micro-F1 and Macro-F1 [11]. The results are averaged over 10 different runs by sampling different training data.

Table 3 reports the results of Wikipedia page classification. Similar conclusion can be made as in the word analogy task. The graph factorization outperforms DeepWalk as DeepWalk ignores the weights of the edges. The LINE-SGDs perform worse due to the divergence of the weights of the edges. The LINE optimized by the edge-sampling treatment performs much better than directly deploying SGD. The LINE(2nd) outperforms LINE(1st) and is slightly better than the graph factorization. Note that with the supervised task, it is feasible to concatenate the embeddings learned with LINE(1st) and LINE(2nd). As a result, the LINE(1st+2nd) method performs significantly better than all other methods. This indicates that the first-order and second-order proximities are complementary to each other.

To provide the readers more insight about the first-order and second-order proximities, Table 4 compares the most similar words to a given word using *first-order* and *second-order* proximity. We can see that by using the contextual proximity, the most similar words returned by the *second-order* proximity are all semantically related words. The most similar words returned by the *first-order* proximity are a mixture of syntactically and semantically related words.

Table 4: Comparison of most similar words using 1st-order and 2nd-order proximity.

Word	Similarity	Top similar words
good	1st	luck bad faith assume nice
	2nd	decent bad excellent lousy reasonable
information	1st	provide provides detailed facts verifiable
	2nd	infomation informaiton informations nonspammy animecons
graph	1st	graphs algebraic finite symmetric topology
	2nd	graphs subgraph matroid hypergraph undirected
learn	1st	teach learned inform educate how
	2nd	learned teach relearn learnt understand

[4] `http://www.csie.ntu.edu.tw/~cjlin/liblinear/`

5.2.2 Social Network

Compared with the language networks, the social networks are much sparser, especially the YOUTUBE network. We evaluate the vertex embeddings through a multi-label classification task that assigns every node into one or more communities. Different percentages of the vertices are randomly sampled for training and the rest are used for evaluation. The results are averaged over 10 different runs.

Flickr Network. Let us first take a look at the results on the FLICKR network. We choose the most popular 5 communities as the categories of the vertices for multi-label classification. Table 5 reports the results. Again, LINE(1st+2nd) significantly outperforms all other methods. LINE(1st) is slightly better than LINE(2nd), which is opposite to the results on the language network. The reasons are two fold: (1) *first-order* proximity is still more important than *second-order* proximity in social network, which indicates strong ties; (2) when the network is too sparse and the average number of neighbors of a node is too small, the *second-order* proximity may become inaccurate. We will further investigate this issue in Section 5.4. LINE(1st) outperforms graph factorization, indicating a better capability of modeling the *first-order* proximity. LINE(2nd) outperforms DeepWalk, indicating a better capability of modeling the *second-order* proximity. By concatenating the representations learned by LINE(1st) and LINE(2nd), the performance further improves, confirming that the two proximities are complementary to each other.

Youtube Network. Table 6 reports the results on YOUTUBE network, which is extremely sparse and the average degree is as low as 5. In most cases with different percentages of training data, LINE(1st) outperforms LINE(2nd), consistent with the results on the FLICKR network. Due to the extreme sparsity, the performance of LINE(2nd) is even inferior to DeepWalk. By combining the representations learned by the LINE with both the *first-* and *second-order* proximity, the performance of LINE outperforms DeepWalk with either 128 or 256 dimension, showing that the two proximities are complementary to each other and able to address the problem of network sparsity.

It is interesting to observe how DeepWalk tackles the network sparsity through truncated random walks, which enrich the neighbors or contexts of each vertex. The random walk approach acts like a depth-first search. Such an approach may quickly alleviate the sparsity of the neighborhood of nodes by bringing in indirect neighbors, but it may also introduce nodes that are long range away. A more reasonable way is to expand the neighborhood of each vertex using a breadth-first search strategy, i.e., recursively adding neighbors of neighbors. To verify this, we expand the neighborhood of the vertices whose degree are less than 1,000 by adding the neighbors of neighbors until the size of the extended neighborhood reaches 1,000 nodes. We find that adding more than 1,000 vertices does not further increase the performance.

The results in the brackets in Table 6 are obtained on this reconstructed network. The performance of GF, LINE(1st) and LINE(2nd) all improves, especially LINE(2nd). In the reconstructed network, the LINE(2nd) outperforms DeepWalk in most cases. We can also see that the performance of LINE(1st+2nd) on the reconstructed network does not improve too much compared with those on the original network. This implies that the combination of *first-order* and *second-order* proximity on the original network has already captured most information and LINE(1st+2nd) approach is a quite effective and efficient way for network embedding, suitable for both dense and sparse networks.

5.2.3 Citation Network

We present the results on two *citation* networks, both of which are directed networks. Both the GF and LINE methods, which use *first-order* proximity, are not applicable for directed networks, and hence we only compare DeepWalk and LINE(2nd). We also evaluate the vertex embeddings through a multi-label classification task. We choose 7 popular conferences including AAAI, CIKM, ICML, KDD, NIPS, SIGIR, and WWW as the classification categories. Authors publishing in the conferences or papers published in the conferences are assumed to belong to the categories corresponding to the conferences.

DBLP(AuthorCitation) Network. Table 7 reports the results on the DBLP(AUTHORCITATION) network. As this network is also very sparse, DeepWalk outperforms LINE(2nd). However, by reconstructing the network through recursively adding neighbors of neighbors for vertices with small degrees (smaller than 500), the performance of LINE(2nd) significantly increases and outperforms DeepWalk. The LINE model directly optimized by stochastic gradient descent, LINE(2nd), does not perform well as expected.

DBLP(PaperCitation) Network. Table 8 reports the results on the DBLP(PAPERCITATION) network. The LINE(2nd) significantly outperforms DeepWalk. This is because the random walk on the paper citation network can only reach papers along the citing path (i.e., older papers) and cannot reach other references. Instead, the LINE(2nd) represents each paper with its references, which is obviously more reasonable. The performance of LINE(2nd) is further improved when the network is reconstructed by enriching the neighbors of vertices with small degrees (smaller than 200).

5.3 Network Layouts

An important application of network embedding is to create meaningful visualizations that layout a network on a two dimensional space. We visualize a co-author network extracted from the DBLP data. We select some conferences from three different research fields: WWW, KDD from "data mining," NIPS, ICML from "machine learning," and CVPR, ICCV from "computer vision." The co-author network is built from the papers published in these conferences. Authors with degree less than 3 are filtered out, and finally the network contains 18,561 authors and 207,074 edges. Laying out this co-author network is very challenging as the three research fields are very close to each other. We first map the co-author network into a low-dimensional space with different embedding approaches and then further map the low-dimensional vectors of the vertices to a 2-D space with the t-SNE package [21]. Fig. 2 compares the visualization results with different embedding approaches. The visualization using graph factorization is not very meaningful, in which the authors belonging to the same communities are not clustered together. The result of DeepWalk is much better. However, many authors belonging to different communities are clustered tightly into the center area, most of which are high degree vertices. This is because DeepWalk uses a random walk based approach to enrich the neighbors of the vertices, which brings in a lot of noise due to the randomness, es-

Table 5: Results of multi-label classification on the FLICKR network.

Metric	Algorithm	10%	20%	30%	40%	50%	60%	70%	80%	90%
Micro-F1	GF	53.23	53.68	53.98	54.14	54.32	54.38	54.43	54.50	54.48
	DeepWalk	60.38	60.77	60.90	61.05	61.13	61.18	61.19	61.29	61.22
	DeepWalk(256dim)	60.41	61.09	61.35	61.52	61.69	61.76	61.80	61.91	61.83
	LINE(1st)	63.27	63.69	63.82	63.92	63.96	64.03	64.06	64.17	64.10
	LINE(2nd)	62.83	63.24	63.34	63.44	63.55	63.55	63.59	63.66	63.69
	LINE(1st+2nd)	**63.20****	**63.97****	**64.25****	**64.39****	**64.53****	**64.55****	**64.61****	**64.75****	**64.74****
Macro-F1	GF	48.66	48.73	48.84	48.91	49.03	49.03	49.07	49.08	49.02
	DeepWalk	58.60	58.93	59.04	59.18	59.26	59.29	59.28	59.39	59.30
	DeepWalk(256dim)	59.00	59.59	59.80	59.94	60.09	60.17	60.18	60.27	60.18
	LINE(1st)	62.14	62.53	62.64	62.74	62.78	62.82	62.86	62.96	62.89
	LINE(2nd)	61.46	61.82	61.92	62.02	62.13	62.12	62.17	62.23	62.25
	LINE(1st+2nd)	**62.23****	**62.95****	**63.20****	**63.35****	**63.48****	**63.48****	**63.55****	**63.69****	**63.68****

Significantly outperforms DeepWalk at the: ** 0.01 and * 0.05 level, paired t-test.

Table 6: Results of multi-label classification on the YOUTUBE network. The results in the brackets are on the reconstructed network, which adds second-order neighbors (i.e., neighbors of neighbors) as neighbors for vertices with a low degree.

Metric	Algorithm	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
Micro-F1	GF	25.43 (24.97)	26.16 (26.48)	26.60 (27.25)	26.91 (27.87)	27.32 (28.31)	27.61 (28.68)	27.88 (29.01)	28.13 (29.21)	28.30 (29.36)	28.51 (29.63)
	DeepWalk	39.68	41.78	42.78	43.55	43.96	44.31	44.61	44.89	45.06	45.23
	DeepWalk(256dim)	39.94	42.17	43.19	44.05	44.47	44.84	45.17	45.43	45.65	45.81
	LINE(1st)	35.43 (36.47)	38.08 (38.87)	39.33 (40.01)	40.21 (40.85)	40.77 (41.33)	41.24 (41.73)	41.53 (42.05)	41.89 (42.34)	42.07 (42.57)	42.21 (42.73)
	LINE(2nd)	32.98 (36.78)	36.70 (40.37)	38.93 (42.10)	40.26 (43.25)	41.08 (43.90)	41.79 (44.44)	42.28 (44.83)	42.70 (45.18)	43.04 (45.50)	43.34 (45.67)
	LINE(1st+2nd)	39.01* (**40.20**)	41.89 (**42.70**)	43.14 (**43.94****)	44.04 (**44.71****)	44.62 (**45.19****)	45.06 (**45.55****)	45.34 (**45.87****)	45.69** (**46.15****)	45.91** (**46.33****)	46.08** (**46.43****)
Macro-F1	GF	7.38 (11.01)	8.44 (13.55)	9.35 (14.93)	9.80 (15.90)	10.38 (16.45)	10.79 (16.93)	11.21 (17.38)	11.55 (17.64)	11.81 (17.80)	12.08 (18.09)
	DeepWalk	28.39	30.96	32.28	33.43	33.92	34.32	34.83	35.27	35.54	35.86
	DeepWalk (256dim)	28.95	31.79	33.16	34.42	34.93	35.44	35.99	36.41	36.78	37.11
	LINE(1st)	28.74 (29.40)	31.24 (31.75)	32.26 (32.74)	33.05 (33.41)	33.30 (33.70)	33.60 (33.99)	33.86 (34.26)	34.18 (34.52)	34.33 (34.77)	34.44 (34.92)
	LINE(2nd)	17.06 (22.18)	21.73 (27.25)	25.28 (29.87)	27.36 (31.88)	28.50 (32.86)	29.59 (33.73)	30.43 (34.50)	31.14 (35.15)	31.81 (35.76)	32.32 (36.19)
	LINE(1st+2nd)	**29.85** (29.24)	31.93 (**33.16****)	33.96 (**35.08****)	35.46** (**36.45****)	36.25** (**37.14****)	36.90** (**37.69****)	37.48** (**38.30****)	38.10** (**38.80****)	38.46** (**39.15****)	38.82** (**39.40****)

Significantly outperforms DeepWalk at the: ** 0.01 and * 0.05 level, paired t-test.

Table 7: Results of multi-label classification on DBLP(AUTHORCITATION) network.

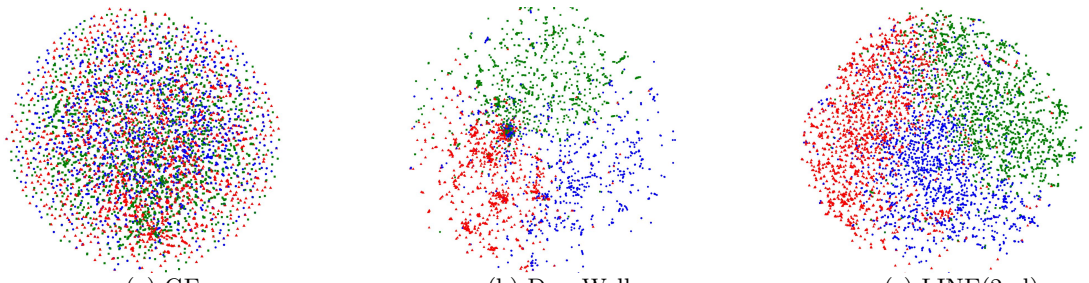
Metric	Algorithm	10%	20%	30%	40%	50%	60%	70%	80%	90%
Micro-F1	DeepWalk	63.98	64.51	64.75	64.81	64.92	64.99	64.99	65.00	64.90
	LINE-SGD(2nd)	56.64	58.95	59.89	60.20	60.44	60.61	60.58	60.73	60.59
	LINE(2nd)	62.49 (**64.69***)	63.30 (**65.47****)	63.63 (**65.85****)	63.77 (**66.04****)	63.84 (**66.19****)	63.94 (**66.25****)	63.96 (**66.30****)	64.00 (**66.12****)	63.77 (**66.05****)
Macro-F1	DeepWalk	63.02	63.60	63.84	63.90	63.98	64.06	64.09	64.11	64.05
	LINE-SGD(2nd)	55.24	57.63	58.56	58.82	59.11	59.27	59.28	59.46	59.37
	LINE(2nd)	61.43 (**63.49***)	62.38 (**64.42****)	62.73 (**64.84****)	62.87 (**65.05****)	62.93 (**65.19****)	63.05 (**65.26****)	63.07 (**65.29****)	63.13 (**65.14****)	62.95 (**65.14****)

Significantly outperforms DeepWalk at the: ** 0.01 and * 0.05 level, paired t-test.

Table 8: Results of multi-label classification on DBLP(PAPERCITATION) network.

Metric	Algorithm	10%	20%	30%	40%	50%	60%	70%	80%	90%
Micro-F1	DeepWalk	52.83	53.80	54.24	54.75	55.07	55.13	55.48	55.42	55.90
	LINE(2nd)	58.42 (**60.10****)	59.58 (**61.06****)	60.29 (**61.46****)	60.78 (**61.73****)	60.94 (**61.85****)	61.20 (**62.10****)	61.39 (**62.21****)	61.39 (**62.25****)	61.79 (**62.80****)
Macro-F1	DeepWalk	43.74	44.85	45.34	45.85	46.20	46.25	46.51	46.36	46.73
	LINE(2nd)	48.74 (**50.22****)	50.10 (**51.41****)	50.84 (**51.92****)	51.31 (**52.20****)	51.61 (**52.40****)	51.77 (**52.59****)	51.94 (**52.78****)	51.89 (**52.70****)	52.16 (**53.02****)

Significantly outperforms DeepWalk at the: ** 0.01 and * 0.05 level, paired t-test.

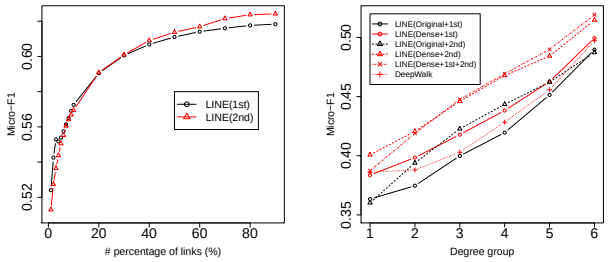


(a) GF (b) DeepWalk (c) LINE(2nd)

Figure 2: Visualization of the co-author network. The authors are mapped to the 2-D space using the t-SNE package with learned embeddings as input. Color of a node indicates the community of the author. Red: "data Mining," blue: "machine learning," green: "computer vision."

pecially for vertices with higher degrees. The LINE(2nd) performs quite well and generates meaningful layout of the network (nodes with same colors are distributed closer).

5.4 Performance w.r.t. Network Sparsity



(a) Sparsity. (b) Degree of vertex.

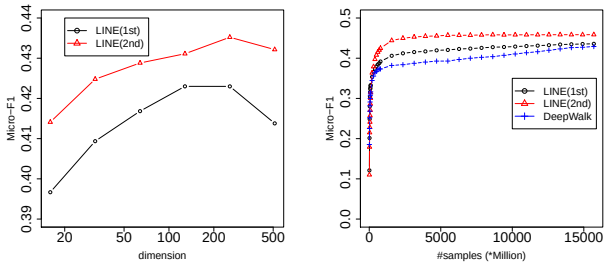
Figure 3: Performance w.r.t. network sparsity.

In this subsection, we formally analyze the performance of the above models w.r.t. the sparsity of networks. We use the social networks as examples. We first investigate how the sparsity of the networks affects the LINE(1st) and LINE(2nd). Fig. 3(a) shows the results w.r.t. the percentage of links on the FLICKR network. We choose FLICKR network as it is much denser than the YOUTUBE network. We randomly select different percentages of links from the original network to construct networks with different levels of sparsity. We can see that in the beginning, when the network is very sparse, the LINE(1st) outperforms LINE(2nd). As we gradually increase the percentage of links, the LINE(2nd) begins to outperform the LINE(1st). This shows that the *second-order* proximity suffers when the network is extremely sparse, and it outperforms *first-order* proximity when there are sufficient nodes in the neighborhood of a node.

Fig. 3(b) shows the performance w.r.t. the degrees of the vertices on both the original and reconstructed YOUTUBE networks. We categorize the vertices into different groups according to their degrees including $(0, 1]$, $[2, 3]$, $[4, 6]$, $[7, 12]$, $[13, 30]$, $[31, +\infty)$, and then evaluate the performance of vertices in different groups. Overall, the performance of different models increases when the degrees of the vertices increase. In the original network, the LINE(2nd) outperforms LINE(1st) except for the *first* group, which confirms that the *second-order* proximity does not work well for nodes with a low degree. In the reconstructed dense network, the performance of the LINE(1st) or LINE(2nd) improves, especially the LINE(2nd) that preserves the *second-order* proximity. We can also see that the LINE(2nd) model on the reconstructed network outperforms DeepWalk in all the groups.

5.5 Parameter Sensitivity

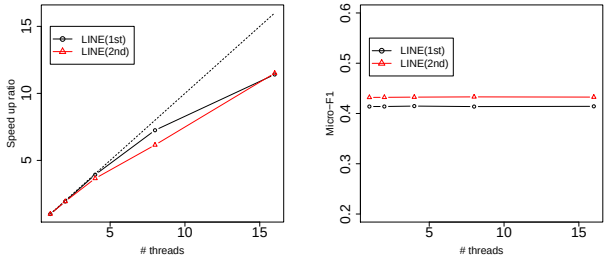
Next, we investigate the performance w.r.t. the parameter dimension d and the converging performance of different models w.r.t the number of samples on the reconstructed YOUTUBE network. Fig. 4(a) reports the performance of the LINE model w.r.t. the dimension d. We can see that the performance of the LINE(1st) or LINE(2nd) drops when the dimension becomes too large. Fig. 4(b) shows the results of the LINE and DeepWalk w.r.t. the number of samples during the optimization. The LINE(2nd) consistently outperforms LINE(1st) and DeepWalk, and both the LINE(1st) and LINE(2nd) converge much faster than DeepWalk.



(a) #Dimension. (b) #Samples.

Figure 4: Sensitivity w.r.t. dimension and samples.

5.6 Scalability



(a) Speed up v.s. #threads. (b) Micro-F1 v.s. #threads.

Figure 5: Performance w.r.t. # threads.

Finally, we investigate the scalability of the LINE model optimized by the edge-sampling treatment and asynchronous stochastic gradient descent, which deploys multiple threads for optimization. Fig. 5(a) shows the speed up w.r.t. the number of threads on the YOUTUBE data set. The speed up is quite close to linear. Fig. 5(b) shows that the classification performance remains stable when using multiple threads for model updating. The two figures together show that the inference algorithm of the LINE model is quite scalable.

6. CONCLUSION

This paper presented a novel network embedding model called the "LINE," which can easily scale up to networks with millions of vertices and billions of edges. It has carefully designed objective functions that preserve both the *first-order* and *second-order* proximities, which are complementary to each other. An efficient and effective edge-sampling method is proposed for model inference, which solved the limitation of stochastic gradient descent on weighted edges without compromising the efficiency. Experimental results on various real-world networks prove the efficiency and effectiveness of LINE. In the future, we plan to investigate higher-order proximity beyond the *first-order* and *second-order* proximities in the network. Besides, we also plan to investigate the embedding of heterogeneous information networks, e.g., vertices with multiple types.

Acknowledgments

The authors thank the three anonymous reviewers for the helpful comments. The co-author Ming Zhang is supported by the National Natural Science Foundation of China (NSFC Grant No. 61472006); Qiaozhu Mei is supported by the National Science Foundation under grant numbers IIS-1054199 and CCF-1048168.

7. REFERENCES

[1] A. Ahmed, N. Shervashidze, S. Narayanamurthy, V. Josifovski, and A. J. Smola. Distributed large-scale natural graph factorization. In *Proceedings of the 22nd international conference on World Wide Web*, pages 37–48. International World Wide Web Conferences Steering Committee, 2013.

[2] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *NIPS*, volume 14, pages 585–591, 2001.

[3] S. Bhagat, G. Cormode, and S. Muthukrishnan. Node classification in social networks. In *Social Network Data Analytics*, pages 115–148. Springer, 2011.

[4] T. F. Cox and M. A. Cox. *Multidimensional scaling*. CRC Press, 2000.

[5] J. R. Firth. A synopsis of linguistic theory, 1930–1955. *In J. R. Firth (Ed.), Studies in linguistic analysis*, pages 1–32.

[6] M. S. Granovetter. The strength of weak ties. *American journal of sociology*, pages 1360–1380, 1973.

[7] Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *Proceedings of The 31st International Conference on Machine Learning*, pages 1188–1196, 2014.

[8] O. Levy and Y. Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems*, pages 2177–2185, 2014.

[9] A. Q. Li, A. Ahmed, S. Ravi, and A. J. Smola. Reducing the sampling complexity of topic models. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 891–900. ACM, 2014.

[10] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007.

[11] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

[12] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[13] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

[14] S. A. Myers, A. Sharma, P. Gupta, and J. Lin. Information network or social network?: the structure of the twitter follow graph. In *Proceedings of the companion publication of the 23rd international conference on World wide web companion*, pages 493–498. International World Wide Web Conferences Steering Committee, 2014.

[15] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. 1999.

[16] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM, 2014.

[17] B. Recht, C. Re, S. Wright, and F. Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 693–701, 2011.

[18] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[19] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su. Arnetminer: extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 990–998. ACM, 2008.

[20] J. B. Tenenbaum, V. De Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[21] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.

[22] S. Yan, D. Xu, B. Zhang, H.-J. Zhang, Q. Yang, and S. Lin. Graph embedding and extensions: a general framework for dimensionality reduction. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(1):40–51, 2007.

[23] X. Yu, X. Ren, Y. Sun, Q. Gu, B. Sturt, U. Khandelwal, B. Norick, and J. Han. Personalized entity recommendation: A heterogeneous information network approach. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 283–292. ACM, 2014.