GraRep: Learning Graph Representations with Global Structural Information

Shaosheng Cao
Xidian University
shelsoncao@gmail.com

Wei Lu
Singapore University of Technology and Design
luwei@sutd.edu.sg

Qiongkai Xu
IBM Research - China
Australian National University
xuqkai@cn.ibm.com

ABSTRACT

In this paper, we present GraRep, a novel model for learning vertex representations of weighted graphs. This model learns low dimensional vectors to represent vertices appearing in a graph and, unlike existing work, integrates global structural information of the graph into the learning process. We also formally analyze the connections between our work and several previous research efforts, including the DeepWalk model of Perozzi *et al.* [20] as well as the skip-gram model with negative sampling of Mikolov *et al.* [18]

We conduct experiments on a language network, a social network as well as a citation network and show that our learned global representations can be effectively used as features in tasks such as clustering, classification and visualization. Empirical results demonstrate that our representation significantly outperforms other state-of-the-art methods in such tasks.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning; H.2.8 [**Database Management**]: Database Applications - Data Mining

General Terms

Algorithms, Experimentation

Keywords

Graph Representation, Matrix Factorization, Feature Learning, Dimension Reduction

1. INTRODUCTION

In many real-world problems, information is often organized using graphs. For example, in social network research, classification of users into meaningful social groups based on social graphs can lead to many useful practical applications such as user search, targeted advertising and recommendations. Therefore, it is essential to accurately learn useful information from the graphs. One strategy is to learn the *graph representations* of a graph: each vertex of the graph is represented with a low-dimensional vector in which meaningful semantic, relational and structural information conveyed by the graph can be accurately captured.

Recently, there has been a surge of interest in learning graph representations from data. For example, DeepWalk [20], one recent model, transforms a graph structure into a sample collection of linear sequences consisting of vertices using uniform sampling (which is also called *truncated random walk*). The skip-gram model [18], originally designed for learning word representations from linear sequences, can also be used to learn the representations of vertices from such samples. Although this method is empirically effective, it is not well understood what is the exact loss function defined over the graph involved in their learning process.

In this work, we first present an explicit loss function of the skip-gram model defined over the graph. We show that essentially we can use the skip-gram model to capture the k-step ($k = 1, 2, 3, \dots$) relationship between each vertex and its k-step neighbors in the graph with different values of k. One limitation of the skip-gram model is it projects all such k-step relational information into a common subspace. We argue that such a simple treatment can lead to potential issues. The above limitation is overcome in our proposed model through the preservation of different k-step relational information in distinct subspaces.

Another recently proposed work is LINE [25], which has a loss function to capture both 1-step and 2-step local relational information. To capture certain complex relations in such local information, they also learn non-linear transformations from such data. While their model can not be easily extended to capture k-step (with $k > 2$) relational information for learning their graph representation, one important strategy used to enhance the effectiveness of their model is to consider higher-order neighbors for vertices with small degrees. This strategy implicitly captures certain k-step information into their model to some extent. We believe k-step relational information between different vertices, with different values of k, reveals the useful global structural information associated with the graph, and it is essential to explicitly take full advantage of this when learning a good graph representation.

In this paper, we propose GraRep, a novel model for learning graph representations for knowledge management. The model captures the different k-step relational information with different values of k amongst vertices from the graph directly by manipulating different global transition matrices

CIKM'15, October 19–23, 2015, Melbourne, Australia.

DOI: http://dx.doi.org/10.1145/2806416.2806512.

defined over the graph, without involving slow and complex sampling processes. Unlike existing work, our model defines different loss functions for capturing the different k-step local relational information (*i.e.*, a different k). We optimize each model with matrix factorization techniques, and construct the global representations for each vertex by combining different representations learned from different models. Such learned global representations can be used as features for further processing.

We give a formal treatment of this model, showing the connections between our model and several previous models. We also demonstrate the empirical effectiveness of the learned representations in solving several real-world problems. Specifically, we conducted experiments on a language network clustering task, a social network multi-label classification task, as well as a citation network visualization task. In all such tasks, GraRep outperforms other graph representation methods, and is trivially parallelizable.

Our contributions are as follows:

- We introduce a novel model to learn latent representations of vertices on graphs, which can capture global structural information associated with the graph.
- We provide from a probabilistic prospective an understanding of the uniform sampling method used in DeepWalk for learning graph representations, which translates a graph structure into linear sequences. Furthermore, we explicitly define their loss function over graphs and extend it to support weighted graphs.
- We formally analyze the deficiency associated with the skip-gram model with negative sampling. Our model defines a more accurate loss function that allows nonlinear combinations of different local relational information to be integrated.

The organization of this paper is described below. Section 2 discusses related work. Section 3 proposes our loss function and states the optimization method using matrix factorization. Section 4 presents the overall algorithm. Section 5 gives a mathematical explanation to elucidate the rationality of the proposed work and shows its connection to previous work. Section 6 discusses the evaluation data and introduces baseline algorithms. Section 7 presents experiments as well as analysis on the parameter sensitivity. Finally, we conclude in Section 8.

2. RELATED WORK

2.1 Linear Sequence Representation Methods

Natural language corpora, consisting of streams of words, can be regarded as special graph structures, that is, linear chains. Currently, there are two mainstream methods for learning word representations: neural embedding methods and matrix factorization based approaches.

Neural embedding methods employ a fixed slide window capturing context words of current word. Models like skip-gram [18] are proposed, which provide an efficient approach to learning word representations. While these methods may yield good performances on some tasks, they can poorly capture useful information since they use separate local context windows, instead of global co-occurrence counts [19]. On the other hand, the family of matrix factorization methods can utilize global statistics [5]. Previous work include Latent Semantic Analysis (LSA) [15], which decomposes term-document matrix and yields latent semantic representations. Lund et al. [17] put forward Hyperspace Analogue to Language (HAL), factorizing a word-word co-occurrence counts matrix to generate word representations. Levy et al. [4] presented matrix factorization over shifted positive Pointwise Mutual Information (PMI) matrix for learning word representations and showed that the Skip-Gram model with Negative Sampling (SGNS) can be regarded as a model that implicitly such a matrix [16].

2.2 Graph Representation Approaches

There exist several classical approaches to learning low dimensional graph representations, such as multidimensional scaling (MDS) [8], IsoMap [28], LLE [21], and Laplacian Eigenmaps [3]. Recently, Tang et al. [27] presented methods for learning latent representational vectors of the graphs which can then be applied to social network classification. Ahmed et al. [1] proposed a *graph factorization* method, which used stochastic gradient descent to optimize matrices from large graphs. Perozzi et al. [20] presented an approach, which transformed graph structure into several linear vertex sequences by using a truncated random walk algorithm and generated vertex representations by using skip-gram model. This is considered as an equally weighted linear combination of k-step information. Tang et al. [25] later proposed a large-scale information network embedding, which optimizes a loss function where both 1-step and 2-step relational information can be captured in the learning process.

3. GRAREP MODEL

In this section, we define our task and present our loss function for our task, which is then optimized with the matrix factorization method.

3.1 Graphs and Their Representations

Definition 1. (Graph) A *graph* is defined as $G = (V, E)$. $V = \{v_1, v_2, \ldots, v_n\}$ is the set of *vertices* with each V indicating one object while $E = \{e_{i,j}\}$ is the set of *edges* with each E indicating the relationship between two vertices. A *path* is a sequence of edges which connect a sequence of vertices.

We first define *adjacency matrix* S for a graph. For an unweighted graph, we have $S_{i,j} = 1$ if and only if there exists an edge from v_i to v_j, and $S_{i,j} = 0$ otherwise. For a weighted graph, $S_{i,j}$ is a real number called the weight of the edge $e_{i,j}$, which indicates the significance of edge. Although weights can be negative, we only consider non-negative weights in this paper. For notational convenience, we also use w and c to denote vertices throughout this paper.

The following diagonal matrix D is known as the *degree matrix* for a graph with adjacency matrix S:

$$D_{ij} = \begin{cases} \sum_p S_{ip}, & \text{if } i = j \\ 0, & \text{if } i \neq j \end{cases}$$

Assume we would like to capture the transitions from one vertex to another, and assume in the adjacency matrix $S_{i,j}$ is proportional to the transition probability from v_i to v_j, we can define the following *(1-step) probability transition matrix*

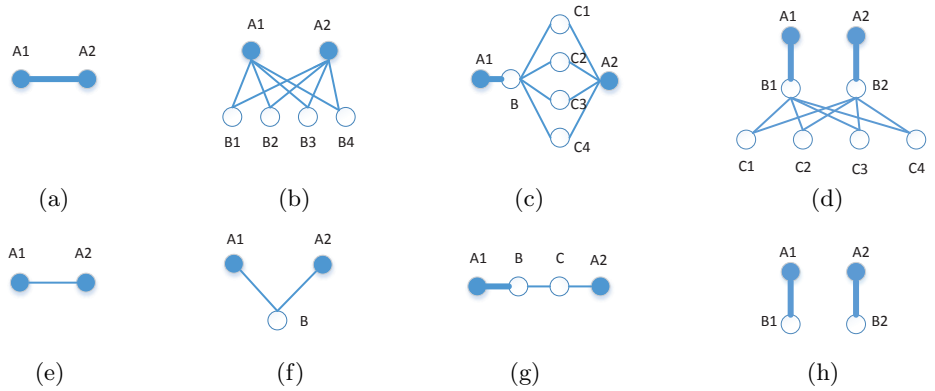


Figure 1: The importance of capturing different k-step information in the graph representations. Here we give examples for $k = 1, 2, 3, 4$.

A:

$$A = D^{-1}S$$

where $A_{i,j}$ is the probability of a transition from v_i to vertex v_j within one step. It can be observed that the A matrix can be regarded as a re-scaled S matrix whose rows are normalized.

Definition 2. (Graph Representations with Global Structural Information) Given a *graph* G, the task of *Learning Graph Representations with Global Structural Information* aims to learn a global representation matrix $W \in \mathbb{R}^{|V| \times d}$ for the complete graph, whose i-th row W_i is a d-dimensional vector representing the vertex v_i in the graph G where the global structural information of the graph can be captured in such vectors.

In this paper, global structural information serves two functions: 1) the capture of long distance relationship between two different vertices and 2) the consideration of distinct connections in terms of different transitional steps. This would be further illustrated later.

As we have discussed earlier, we believe the k-step (with varying k) relational information from the graph needs to be captured when constructing such global graph representations. To validate this point, Figure 1 gives some illustrative examples showing the importance of k-step (for k=1,2,3,4) relational information that needs to be captured between two vertices A_1 and A_2. In the figure, a thick line indicates a strong relation between two vertices, while a thin line indicates a weaker relation. Here, (a) and (e) show the importance of capturing the simple 1-step information between the two vertices which are directly connected to each other, where one has a stronger relation and the other has a weaker relation. In (b) and (f), 2-step information is shown, where in (b) both vertices share many common neighbors, and in (f) only one neighbor is shared between them. Clearly, 2-step information is important in capturing how strong the connection between the two vertices is – the more common neighbors they share, the stronger the relation between them is. In (c) and (g), the importance of 3-step information is illustrated. Specifically, in (g), despite the strong relation between A_1 and B, the relation between A_1 and A_2 can be weakened due to the two weaker edges connecting B and C, as well as C and A_2. In contrast, in (c), the relation between A_1 and A_2 remains strong because of the large number of common neighbors between B and A_2 which strengthened their relation. Clearly such 3-step information is essential to be captured when learning a good graph representation with global structural information. Similarly, the 4-step information can also be crucial in revealing the global structural properties of the graph, as illustrated in (d) and (h). Here, in (d), the relation between A_1 and A_2 is clearly strong, while in (h) the two vertices are unrelated since there does not exist a path from one vertex to the other. In the absence of 4-step relational information, such important distinctions can not be properly captured.

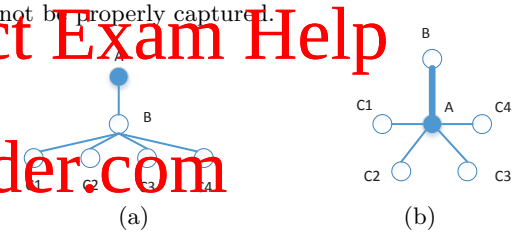


Figure 2: The importance of maintaining different k-step information separately in the graph representations.

We also additionally argue that it is essential to treat different k-step information differently when learning graph representations. We present one simple graph in (a) of Figure 2. Let us focus on learning the representation for the vertex A in the graph. We can see that A receives two types of information when learning its representation: 1-step information from B, as well as 2-step information from all C vertices. We note that if we do not distinguish these two different types of information, we can construct an alternative graph as shown in (b) of Figure 2, where A receives exactly the same information as (a), but has a completely different structure.

In this paper, we propose a novel framework for learning accurate graph representations, integrating various k-step information which together captures the global structural information associated with the graph.

3.2 Loss Function On Graph

We discuss our loss function used for learning the graph representations with global structural information in this section. Assume we are now given a graph with a collection of vertices and edges. Consider a vertex w together with another vertex c. In order for us to learn global representations to capture their relations, we need to understand how strongly these two vertices are connected to each other.

Let us begin with a few questions. Does there exist a path from w to c? If so, if we randomly sample a path starting with w, how likely is it for us to reach c (possibly within a fixed number of steps)? To answer such questions, we first use the term $p_k(c|w)$ to denote the probability for a transition from w to c in exactly k steps. We already know what are the 1-step transition probabilities, to compute the k-step transition probabilities we introduce the following *k-step probability transition matrix*

$$A^k = \underbrace{A \cdots A}_{k}$$

We can observe that $A^k_{i,j}$ exactly refers to the transition probability from vertex i to vertex j where the transition consists of exactly k step(s). This directly leads to:

$$p_k(c|w) = A^k_{w,c}$$

where $A^k_{w,c}$ is the element from w-th row and c-th column of the matrix A^k.

Let us consider a particular k first. Given a graph G, consider the collection of all paths consisting of k steps that can be sampled from the graph which start with w and end with c (here we call w "current vertex", and c "context vertex"). Our objective aims to maximize: 1) the probability that these pairs come from the graph, and 2) the probability that all other pairs do not come from the graph.

Motivated by the skip-gram model by Mikolov et al. [18], we employ *noise contrastive estimation* (NCE), which is proposed by Gutmann et al. [11], to define our objective function. Following a similar discussion presented in [16], we first introduce our k-step loss function defined over the complete graph as follows:

$$L_k = \sum_{w \in V} L_k(w)$$

where

$$L_k(w) = \left(\sum_{c \in V} p_k(c|w) \log \sigma(\vec{w} \cdot \vec{c}) \right) + \lambda \mathbb{E}_{c' \sim p_k(V)} [\log \sigma(-\vec{w} \cdot \vec{c'})]$$

Here $p_k(c|w)$ describes the k-step relationship between w and c (the k-step transition probability from w to c), $\sigma(\cdot)$ is sigmoid function defined as $\sigma(x) = (1 + e^{-x})^{-1}$, λ is a hyper-parameter indicating the number of negative samples, and $p_k(V)$ is the distribution over the vertices in the graph. The term $\mathbb{E}_{c' \sim p_k(V)}[\cdot]$ is the expectation when c' follows the distribution $p_k(V)$, where c' is an instance obtained from *negative sampling*. This term can be explicitly expressed as:

$$\begin{aligned} &\mathbb{E}_{c' \sim p_k(V)}[\log \sigma(-\vec{w} \cdot \vec{c'})] \\ &= p_k(c) \cdot \log \sigma(-\vec{w} \cdot \vec{c}) + \sum_{c' \in V \setminus \{c\}} p_k(c') \cdot \log \sigma(-\vec{w} \cdot \vec{c'}) \end{aligned}$$

This leads to a local loss defined over a specific (w,c):

$$L_k(w,c) = p_k(c|w) \cdot \log \sigma(\vec{w} \cdot \vec{c}) + \lambda \cdot p_k(c) \cdot \log \sigma(-\vec{w} \cdot \vec{c})$$

In this work, we set a maximal length for each path we consider. In other words, we assume $1 \le k \le K$. In fact, when k is large enough, the transition probabilities converge to certain fixed values. The distribution $p_k(c)$ can be computed as follows:

$$p_k(c) = \sum_{w'} q(w') p_k(c|w') = \frac{1}{N} \sum_{w'} A^k_{w',c}$$

Note that here N is the number of vertices in graph G, and $q(w')$ is the probability of selecting w' as the first vertex in the path, which we assume follow a uniform distribution, *i.e.*, $q(w') = 1/N$. This leads to:

$$L_k(w,c) = A^k_{w,c} \cdot \log \sigma(\vec{w} \cdot \vec{c}) + \frac{\lambda}{N} \sum_{w'} A^k_{w',c} \cdot \log \sigma(-\vec{w} \cdot \vec{c})$$

Following [16], we define $e = \vec{w} \cdot \vec{c}$, and setting $\frac{\partial L_k}{\partial e} = 0$. This yields the following:

$$\vec{w} \cdot \vec{c} = \log \left(\frac{A^k_{w,c}}{\sum_{w'} A^k_{w',c}} \right) - \log(\beta)$$

where $\beta = \lambda / N$.

This concludes that we essentially need to factorize the matrix Y into two matrices W and C, where each row of W and each row of C consists of a vector representation for the vertex w and c respectively, and the entries of Y are:

$$Y^k_{i,j} = W^k_i \cdot C^k_j = \log \left(\frac{A^k_{i,j}}{\sum_t A^k_{t,j}} \right) - \log(\beta)$$

Now we have defined our loss function and showed that optimizing the proposed loss essentially involves a matrix factorization problem.

3.3 Optimization with Matrix Factorization

Following the work of Levy et al. [16], to reduce noise, we replace all negative entries in Y^k with 0. This gives us a positive k-step log probabilistic matrix X^k, where

$$X^k_{ij} = \max(Y^k_{i,j}, 0)$$

While various techniques for matrix factorization exist, in this work we focus on the popular singular value decomposition (SVD) method due to its simplicity. SVD has been shown successful in several matrix factorization tasks [9, 14], and is regarded as one of the important methods that can be used for dimensionality reduction. It was also used in [16].

For the matrix X^k, SVD factorizes it as:

$$X^k = U^k \Sigma^k (V^k)^T$$

where U and V are orthonormal matrices and Σ is a diagonal matrix consisting of an ordered list of singular values.

We can approximate the original matrix X^k with X^k_d:

$$X^k \approx X^k_d = U^k_d \Sigma^k_d (V^k_d)^T$$

where Σ^k_d is the matrix composed by the top d singular values, and U^k_d and V^k_d are first d columns of U^k and V^k, respectively (which are the first d eigenvector of XX^T and X^TX respectively).

This way, we can factorize our matrix X^k as:

$$X^k \approx X^k_d = W^k C^k$$

where

$$W^k = U^k_d (\Sigma^k_d)^{\frac{1}{2}}, \qquad C^k = (\Sigma^k_d)^{\frac{1}{2}} {V^k_d}^T$$

The resulting W^k gives representations of current vertices as its column vectors, and C^k gives the representations of context vertices as its column vectors [6, 5, 30]. The final matrix W^k is returned from the algorithm as the low-d representations of the vertices which capture k-step global structural information in the graph.

In our algorithm we consider all k-step transitions with all $k = 1, 2, \ldots, K$, where K is a pre-selected constant. In our algorithm, we integrate all such k-step information when learning our graph representation, as to be discussed next.

Note that here we are essentially finding a projection from the row space of X^k to the row space of W^k with a lower rank. Thus alternative approaches other than the popular SVD can also be exploited. Examples include incremental SVD [22], independent component analysis (ICA) [7, 13], and deep neural networks [12]. Our focus in this work is on the novel model for learning graph representations, so we do not pursue any alternative methods. In fact, if alternative method such as sparse auto-encoder is used at this step, it becomes relatively harder to justify whether the empirical effectiveness of our representations is due to our novel model, or comes from any non-linearity introduced in this dimensionality reduction step. To maintain the consistency with Levy et al. [16], we only employed SVD in this work.

4. ALGORITHM

We detail our learning algorithm in this section. In general, graph representations are extracted for other applications as features, such as classification and clustering. An effective way to encode k-step representation in practice is to concatenate the k-step representation as a global feature for each vertex, since each different step representation reflects different local information. Table 1 shows the overall algorithm, and we explain the essential steps of our algorithm here.

Step 1. Get k-step transition probability matrix A^k for each $k = 1, 2, \ldots, K$.

As shown in Section 3, given a graph S, we can calculate the k-step transition probability matrix A^k through the product of inverse of degree matrix D and adjacent matrix S. For a weighted graph, S is a real matrix, while for an unweighted graph, S is a binary matrix. Our algorithm is applicable to both cases.

Step 2. Get each k-step representation

We get k-step log probability matrix X^k, then subtract each entry by $\log(\beta)$, and replace the negative entries by zeros. After that, we construct the representational vectors as rows of W^k, where we introduce a solution to factorize the positive log probability matrix X^k using SVD. Finally, we get all k-step representations for each vertex on the graph.

Step 3. Concatenate all k-step representations

We concatenate all k-step representations to form a global representation, which can be used in other tasks as features.

5. SKIP-GRAM MODEL AS A SPECIAL CASE OF GRAREP

GraRep aims to learn representations for graphs where we optimize the loss function based on matrix factorization. On the other hand, SGNS has been shown to be successful in handling linear structures such as natural language sentences. Is there any intrinsic relationship between them? In this section, we provide a view of SGNS as a special case of the GraRep model.

5.1 Explicit Loss of Skip-gram Model on Graph

SGNS aims at representing words in linear sequences, so we need to translate a graph structure into linear structures. Deepwalk reveals an effective way with uniform sampling

Table 1: Overall Algorithm

GraRep Algorithm
Input
Adjacency matrix S on graph
Maximum transition step K
Log shifted factor β
Dimension of representation vector d
1. Get k-step transition probability matrix A^k
Compute $A = D^{-1}S$
Calculate $A^1, A^2, \ldots, A^K$, respectively
2. Get each k-step representations
For $k = 1$ to K
2.1 Get positive log probability matrix
calculate $\Gamma_1^k, \Gamma_2^k, \ldots, \Gamma_N^k$ ($\Gamma_j^k = \sum_p A_{p,j}^k$) respectively
calculate $\{X_{i,j}^k\}$
$X_{i,j}^k = \log\left(\frac{A_{i,j}^k}{\Gamma_j^k}\right) - \log(\beta)$
assign negative entries of X^k to 0
2.2 Construct the representation vector W^k
$[U^k, \Sigma^k, (V^k)^T] = SVD(X^k)$
$W^k = U_d^k (\Sigma_d^k)^{\frac{1}{2}}$
End for
3. Concatenate all the k-step representations
$W = [W^1, W^2, \ldots, W^K]$
Output
Matrix of the graph representation W

(*truncated random walk*). This method first samples uniformly a random vertex from the graph, then walks randomly to one of its neighbors and repeats this process. If the length of the vertex sequence reaches a certain preset value, then stop and start generating a new sequence. This procedure can be used to produce a large number of sequences from the graph.

Essentially, for an unweighted graph, this strategy of uniform sampling works, while for a weighted graph, a probabilistic sampling method based on the weights of the edges is needed, which is not employed in DeepWalk. In this paper, we propose an Enhanced SGNS (E-SGNS) method suitable for weighted graphs. We also note that DeepWalk optimizes an alternative loss function (hierarchical softmax) that is different from negative sampling.

First, we consider total K-step loss L on whole graph

$$L = f(L_1, L_2, \ldots, L_K)$$

where $f(\cdot)$ is a linear combination of its arguments defined as follows:

$$f(\varphi_1, \varphi_2, \cdots, \varphi_K) = \varphi_1 + \varphi_2 + \cdots + \varphi_K$$

We focus on the loss of a specific pair (w,c) which are i-th and j-th vertex in the graph. Similar to Section 3.2, we assign partial derivative to 0, and get

$$Y_{i,j}^{E-SGNS} = \log\left(\frac{M_{i,j}}{\sum_t M_{t,j}}\right) - \log(\beta)$$

where M is transition probability matrix within K step(s), and $M_{i,j}$ refers to transition probability from vertex i to

vertex j. $Y_{i,j}^{E-SGNS}$ is a factorized matrix for E-SGNS, and

$$M = A^1 + A^2 + \cdots + A^K$$

The difference between E-SGNS and GraRep model is on the definition of $f(\cdot)$. E-SGNS can be considered as a linear combination of K-step loss, and each loss has an equal weight. Our GraRep model does not make such a strong assumption, but allows their (potentially non-linear) relationship to be learned from data in practice. Intuitively, different k-step transition probabilities should have different weights, and linear combination of these may not achieve desirable results for heterogeneous network data.

5.2 Intrinsic Relation Between Sampling and Transition Probabilities

In our approach, we used transition probabilities to measure relationship between vertices. Is this reasonable? In this subsection, we articulate the intrinsic relation between sampling and transition probabilities.

Among the sequences generated by random walk, we assume vertex w occurs total times:

$$\#(w) = \alpha_w \gamma K$$

where γ is the total length of sequences and α_w is the probability of observing the vertex w in such all sequences. We regard vertex w as the current vertex, then the expected number of times that we see c_1 as its direct neighbor (1-step away from w) is:

$$\#(w, c_1) = \alpha_w \gamma K \cdot p_1(c|w)$$

This holds for both uniform sampling for unweighted graphs or our proposed probabilistic sampling for weighted graphs.

Further, we analyze the expected number of times of co-occurrence for w and c of context window size 2,

$$\#(w, c_2) = \alpha_w \gamma K \sum_{c'} p(c_2|c') \cdot p(c'|w) = \alpha_w \gamma K \cdot p_2(c|w)$$

where c' can be any vertex bridging w and c_2. That is, c' is shared neighbor between w and c_2. Similarly, we can derive the equations for $k = 3, 4, \cdots, K$:

$$\#(w, c_3) = \alpha_w \gamma K \cdot p_3(c|w)$$

$$\vdots$$

$$\#(w, c_K) = \alpha_w \gamma K \cdot p_K(c|w)$$

then we add them up and divide both sides by K, leading to:

$$\#(w, c) = \alpha_w \gamma \sum_{k=1}^{K} p_k(c|w)$$

where $\#(w, c)$ is the expected co-occurrence count between w and c within K step(s).

According to definition of $M_{w,c}$, we can get

$$\#(w, c) = \alpha_w \gamma M_{w,c}$$

Now, we can also compute the expected number of times we see c as the context vertex, $\#(c)$:

$$\#(c) = \sum_{w} \alpha_w \gamma M_{w,c}$$

where we consider the transitions from all possible vertices to c. To find the relationship with SGNS model, we consider a special case for α_w here. If we assume α_w follows an uniform distribution, we have $\alpha_w = \frac{1}{N}$. We plug these expected counts into the equation of $Y_{i,j}^{E-SGNS}$, and we arrive at:

$$Y_{w,c}^{E-SGNS} = \log\left(\frac{\#(w, c) \cdot |\mathcal{D}|}{\#(w) \cdot \#(c)}\right) - \log(\lambda)$$

where $\mathcal{D}$ is the collection of all observed pairs in sequences, that is, $|D| = \gamma K$. This matrix Y^{E-SGNS} becomes exactly the same as that of SGNS as described in [16].

This shows SGNS is essentially a special version of our GraRep model that deals with linear sequences which can be sampled from graphs. Our approach has several advantages over the slow and expensive sampling process, which typically involves several parameters to tune, such as maximum length of linear sequence, sampling frequency for each vertex, and so on.

6. EXPERIMENTAL DESIGN

In this section, we assess the effectiveness of our GraRep model through experiments. We conduct experiments on several real-word datasets for several different tasks, and make comparisons with baseline algorithms.

6.1 Datasets and Tasks

In order to demonstrate the performance of GraRep, we conducted the experiments across three different types of graphs – a social network, a language network and a citation network, which include both weighted and unweighted graphs. We conducted experiments across three different types of tasks, including clustering, classification, and visualization. As we mentioned in earlier sections, this work focuses on proposing a novel framework for learning good representation of graph with global structural information, and aims to validate the effectiveness of our proposed model. Thus we do not employ alternative more efficient matrix factorization method apart from SVD, and focused on the following three real-world datasets in this section.

1. **20-Newsgroup**[1] is a language network, which has approximately 20,000 newsgroup documents and is partitioned by 20 different groups. In this network, each document is represented by a vector with tf-idf scores of each word, and the cosine similarity is used to calculate the similarity between two documents. Based on these similarity scores of each pair of documents, a language network is built. Following [29], in order to show the robustness of our model, we also construct the following 3 graphs built from 3, 6 and 9 different newsgroups respectively (note that NG refers to "Newsgroups"):

3-NG:*comp.graphics, comp.graphics and talk.politics.guns;*
6-NG: *alt.atheism, comp.sys.mac.hardware, rec.motorcycles, rec.sport.hockey, soc.religion.christian and talk.religion.misc;*
9-NG: *talk.politics.mideast, talk.politics.misc, comp.os.ms-windows.misc, sci.crypt, sci.med, sci.space, sci.electronics, misc.forsale, and comp.sys.ibm.pc.hardware*

Besides randomly sampling 200 documents from a topic as described in [29], we also conduct experiment on all documents as comparison. The topic label on each document is considered to be true.

This language network is a fully connected and weighted graph, and we will demonstrate the results of clustering, using graph representation as features.

[1] qwone.com/~jason/20Newsgroups/

2. **Blogcatalog**[2] is a social network, where each vertex indicates one blogger author, and each edge corresponds to the relationship between authors. 39 different types of topic categories are presented by authors as labels.

Blogcatalog is an unweighted graphs, and we test the performance of the learned representations on the multi-label classification task, where we classify each author vertex into a set of labels. The graph representations generated from our model and each baseline algorithm are considered as features.

3. **DBLP Network**[3] is a citation network. We extract author citation network from DBLP, where each vertex indicates one author and the number of references from one author to the other is recorded by the weight of edge between these two authors. Following [26], we totally select 6 different popular conferences and assign them into 3 groups, where WWW and KDD are grouped as *data mining*, NIPS and ICML as *machine learning*, and CVPR and ICCV as *computer vision*. We visualize the learned representations from all systems using a visualisation tool t-SNE [31], which provides both qualitative and quantitative results for the learned representations. We give more details in Section 6.4.3.

To summarize, in this paper we conduct experiments on both weighted and unweighted graphs, and both sparse and dense graphs, where three different types of learning tasks are carried out. More details of the graph we used are shown in Table 2.

6.2 Baseline Algorithms

We use the following methods of graph representation as baseline algorithms.

1. **LINE** [25]. LINE is a recently proposed method for learning graph representations on large-scale information networks. LINE defines a loss function based on 1-step and 2-step relational information between vertices. One strategy to improve the performance of vertices with small degrees is to make graph denser by expanding their neighbors. LINE will get the best performance, if concatenating the representation of 1-step and 2-step relational information and tuning the threshold of maximum number of vertices.

2. **DeepWalk** [20]. DeepWalk is a method that learns the representation of social networks. The original model only works for unweighted graph. For each vertex, truncated random walk is used to translate graph structure into linear sequences. The skip-gram model with hierarchical softmax is used as the loss function.

3. **E-SGNS**. Skip-gram is an efficient model that learns the representation of each word in large corpus [18]. For this enhanced version, we first utilize uniform sampling for unweighted graph and probabilistic sampling proportional to weight of edges for weighted graph, to generate linear vertex sequences, and then introduce SGNS to optimize. This method can be regarded as a special case of our model, where different representational vector of each k-step information is averaged.

4. **Spectral Clustering** [23]. Spectral clustering is a reasonable baseline algorithm, which aims at minimizing Normalized Cut (NCut). Like our method, Spectral clustering also factorize a matrix, but it focuses on a different matrix of the graphs – the *Laplacian Matrix*. Essentially, the difference between spectral clustering and E-SGNS lies on their different loss function.

6.3 Parameter Settings

As suggested in [25], for LINE, we set the mini-batch size of stochastic gradient descent (SGD) as 1, learning rate of starting value as 0.025, the number of negative samples as 5, and the total number of samples as 10 billion. We also concatenate both 1-step and 2-step relational information to form the representations and employ the reconstruction strategy for vertices with small degrees to achieve the optimal performance. As mentioned in [20], for DeepWalk and E-SGNS, we set window size as 10, walk length as 40, walks per vertex as 80. According to [25], LINE yielded better results when the learned graph representations are L2 normalized, while DeepWalk and E-SGNS can achieve optimal performance without normalization. For GraRep, we found the L2 normalization yielded better results. We reported all the results based on these findings for each system accordingly. For a fair comparison, the dimension d of representations is set as 128 for Blogcatalog network and DBLP network as used in [25], and is set as 64 for 20-NewsGroup network as used in [20]. For GraRep, we set $\beta = \frac{1}{N}$ and maximum matrix transition step K=6 for Blogcatalog network and DBLP network and K=3 for 20-NewsGroup network. To demonstrate the advantage of our model which captures global information of the graph, we also conducted experiments under various parameter settings for each baseline systems, which are discussed in detail in the next section.

6.4 Experimental Results

In this section, we present empirical justifications that our GraRep model can integrate different k-step local relational information into a global graph representation for different types of graphs, which can then be effectively used for different tasks. We make the source code of GraRep available at http://shelson.top/.

6.4.1 20-Newsgroup Network

We first conduct an experiment on a language network through a clustering task by employing the learned representations in a k-means algorithm [2].

To assess the quality of the results, we report the averaged Normalized Mutual Information (NMI) score [24] over 10 different runs for each system. To understand the effect of different dimensionality d in the end results, we also show the results when dimension d is set to 192 for DeepWalk, E-SGNS and Spectral Clustering. For LINE, we employ the reconstruction strategy proposed by their work by adding neighbors of neighbors as additional neighbors to improve performance. We set k-max=0, 200, 500, 1000 for experiments, where k-max is a parameter that is used to control how the higher-order neighbors are added to each vertex in the graph.

As shown in Table 3, the highest results are highlighted in bold for each column. We can see that GraRep consistently outperforms other baseline methods for this task. For DeepWalk, E-SGNS and Spectral Clustering, increasing

[2]leitang.net/code/social-dimension/data/blogcatalog.mat
[3]aminer.org/billboard/citation

Table 2: Statistics of the real-world graphs

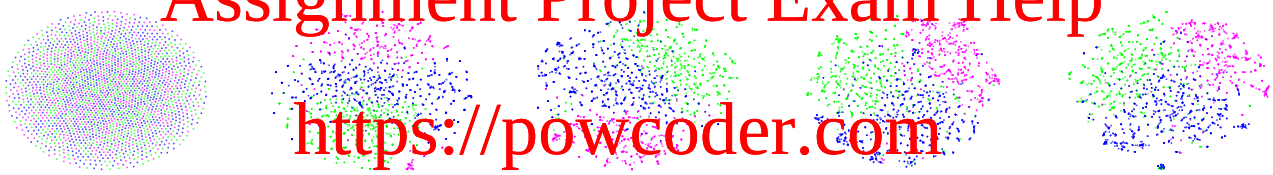
	Language Network		Social Network	Citation Network
Name	20-NewsGroup (200 samples)	20-NewsGroup (all data)	Blogcatalog	DBLP (author citation)
Type	weighted	weighted	unweighted	weighted
#(V)	600, 1200 and 1800	1,720, 3,224 and 5,141	10,312	7,314
#(E)	Fully connected	Fully connected	333,983	72,927
Avg. degree	—	—	64.78	19.94
#Labels	3, 6 and 9	3, 6 and 9	39	3
Task	Clustering	Clustering	Classification	Visualization

Table 3: Results on 20-NewsGroup

	200 samples			all data		
Algorithm	3NG(200)	6NG(200)	9NG(200)	3NG(all)	6NG(all)	9NG(all)
GraRep	**81.12**	**67.53**	**59.43**	**81.44**	**71.54**	**60.38**
LINE (k-max=0)	80.36	64.88	51.58	80.58	68.35	52.30
LINE (k-max=200)	78.69	66.06	54.14	80.68	68.83	53.53
DeepWalk	65.58	63.66	48.86	65.67	68.38	49.19
DeepWalk (192dim)	60.89	59.89	47.16	59.93	65.68	48.61
E-SGNS	69.98	65.06	48.47	69.04	67.65	50.59
E-SGNS (192dim)	63.55	64.85	48.65	66.64	66.57	49.78
Spectral Clustering	49.04	51.02	46.92	62.41	59.32	51.91
Spectral Clustering (192dim)	28.44	27.80	36.05	44.47	36.98	47.36
Spectral Clustering (16dim)	69.91	60.54	47.39	78.12	68.78	57.87

Table 4: Results on Blogcatalog

Metric	Algorithm	10%	20%	30%	40%	50%	60%	70%	80%	90%
Micro-F1	GraRep	**38.24**	**40.31**	**41.34**	**41.87**	**42.60**	**43.02**	**43.43**	**43.55**	**44.24**
	LINE	37.19	39.82	40.88	41.47	42.19	42.72	43.15	43.36	43.88
	DeepWalk	35.93	38.38	39.50	40.39	40.79	41.28	41.60	41.93	42.17
	E-SGNS	35.71	38.34	39.64	40.39	41.23	41.66	42.01	42.16	42.25
	Spectral Clustering	37.16	39.45	40.22	40.87	41.27	41.50	41.48	41.62	42.12
Macro-F1	GraRep	**23.20**	**25.55**	**26.69**	**27.53**	**28.35**	**28.78**	**29.67**	**29.96**	**30.93**
	LINE	19.63	23.04	24.52	25.70	26.65	27.26	27.94	28.68	29.38
	DeepWalk	21.02	23.81	25.39	26.27	26.85	27.36	27.67	27.96	28.41
	E-SGNS	21.01	24.09	25.61	26.59	27.64	28.08	28.33	28.34	29.26
	Spectral Clustering	19.26	22.24	23.51	24.33	24.83	25.19	25.36	25.52	26.21



(a) Spectral Clustering (b) DeepWalk (c) E-SGNS (d) LINE (e) GraRep

Figure 3: Visualization of author citation network. Each point indicates one author. Green: *Data Mining*, magenta: *computer vision* and blue: *Machine Learning*.

the dimension d of representations does not appear to be effective in improving the performance. We believe this is because a higher dimension does not provide different complementary information to the representations. For LINE, the reconstruction strategy does help, since it can capture additional structural information of the graph beyond 1-step and 2-step local information.

It is worth mentioning that GraRep and LINE can achieve good performance with a small graph. We believe this is because both approaches can capture rich local relational information even when the graph is small. Besides, for tasks with more labels, such as 9NG, GraRep and LINE can provide better performances than other methods, with GraRep providing the best. An interesting finding is that Spectral Clustering arrives at its best performance when setting dimension d to 16. However, for all other algorithms, their best results are obtained when d is set to 64. We show these results in Figure 5 when we assess the sensitivity of parameters.

Due to limited space, we do not report the detailed results with d=128 and d=256 for all baseline methods, as well as k-max=500, 1000 and without reconstruction strategy for LINE, since these results are no better than the results presented in Table 3. In Section 6.5, we will further discuss the issue of parameter sensitivity.

6.4.2 *Blogcatalog Network*

In this experiment, we focus on a supervised task on a social network. We evaluate the effectiveness of different graph representations through a multi-label classification task by regarding the learned representations as features.

Following [20, 27], we use the LibLinear package [10] to train one-vs-rest logistic regression classifiers, and we run this process for 10 rounds and report the averaged Micro-F1 and Macro-F1 measures. For each round, we randomly sample 10% to 90% of the vertices and use these samples for training, and use the remaining vertices for evaluation. As suggested in [25], we set k-max as 0, 200, 500 and 1000, respectively, and we report the best performance with k-max=500.

Table 4 reports the results of GraRep and each baseline algorithm. The highest performance is highlighted in bold for each column which corresponds to one particular train/test split. Overall, GraRep significantly outperforms other methods, especially under the setting where only 10% of the vertices are used for training. This indicates different types of rich local structural information learned by the GraRep model can be used to complement each other to capture the global structural properties of the graph, which serves

Table 5: Final KL divergence for the DBLP dataset

Algorithm	GraRep	LINE	DeepWalk	E-SGNS
KL divergence	1.0070	1.0816	1.1115	1.1009

as a distinctive advantage over existing baseline approaches, especially when the data is relatively scarce.

6.4.3 DBLP Network

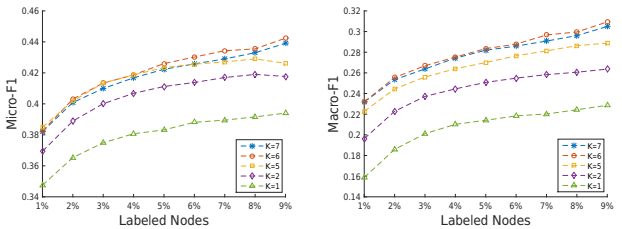
In this experiment, we focus on visualizing the learned representations by examining a real citation network – DBLP. We feed the learned graph representations into the standard *t-SNE* tool [31] to lay out the graph, where the authors from the same research area (one of *data mining, machine learning* or *computer vision*, see Section 6.1) share the same color. The graphs are visualized on a 2-dimensional space and the Kullback-Leibler divergence is reported [31], which captures the errors between the input pairwise similarities and their projections in the 2-dimensional map as displayed (a lower KL divergence score indicates a better performance).

Under the help of *t-SNE* toolkit, we set the same parameter configuration and import representations generated by each algorithm. From Figure 3, we can see that the layout using Spectral Clustering is not very informative, since vertices of different colors are mixed with each other. For DeepWalk and E-SGNS, results look much better, as most vertices of the same color appear to form groups. However, vertices still do not appear in clearly separable regions with clear boundaries. For LINE and GraRep, the boundaries of each group become much clearer, with vertices of different colors appearing in clearly distinguishable regions. The results for GraRep appear to be better, with clearer boundaries for each regions as compared to LINE.

Table 5 reports the KL divergence at the end of iterations. Under the same parameter setting, a lower KL divergence indicates a better graph representation. From this result, we also can see that GraRep yields better representations than all other baseline methods.

6.5 Parameter Sensitivity

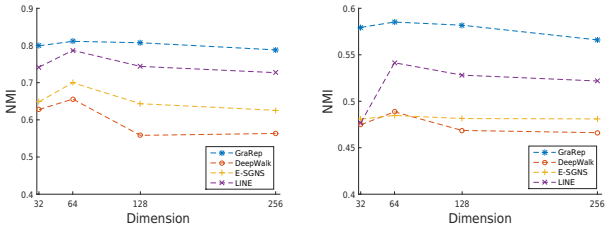
We discuss the parameter sensitivity in this section. Specifically, we assess the how the different choices of the maximal k-step size K, as well as the dimension d for our representations can affect our results.



(a) Blogcatalog, Micro-F1 (b) Blogcatalog, Macro-F1

Figure 4: Performance over steps, K

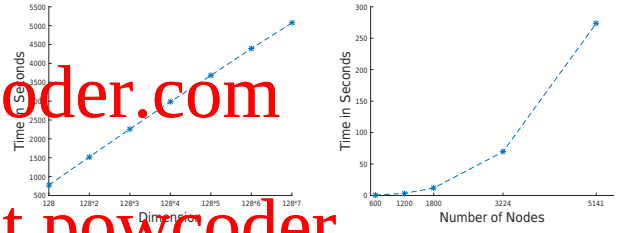
Figure 4 shows the Micro-F1 and Macro-F1 scores over different choices of K on the Blogcatalog data. We can observe that the setting K=2 has a significant improvement over the setting K=1, and K=3 further outperforms K=2. This confirms that different k-step can learn complementary local information. In addition, as mentioned in Section 3, when k is large enough, learned k-step relational information becomes weak and shifts towards a steady distribution. Empirically we can observe that the performance of K=7 is no better than K=6. In this figure, for readability and clarity we only present the results of K=1,2,3,6 and 7. We found the performance of $K = 4$ is slightly better than K=3, while the results for K=5 is comparable to K=4.



(a) 3NG, NMI (b) 9NG, NMI

Figure 5: Performance over dimensions, d

Figure 5 shows the NMI scores of each algorithm over different settings of the dimension d on 3NG and 9NG data. We can observe that GraRep consistently outperforms other baseline algorithms which learn representations with the same dimension. This set of experiments serves as an additional supplement to Table 3. Interestingly, all algorithms can obtain the optimal performance with $d = 64$. As we increase d from 64 to larger values, it appears that the performances of all the algorithms start to drop. Nevertheless, our GraRep algorithm is still superior to the baseline systems across different d values.



(a) Blogcatalog, running time (b) 20NG, running time

Figure 6: Running time as a function of a) dimension and b) size of graph.

Figure 6 shows the running time over different dimensions as well as over different graph sizes. In Figure 6(a), we set K from 1 to 7 where each complementary feature vector has a dimension of 128. This set of experiments is conducted on the Blogcatalog dataset which contains around 10,000 vertices. It can be seen that the running time increases approximately linearly as the dimension increases. In Figure 6(b), we analyze running time with respect to graph sizes. This set of experiment is conducted on different size of the 20-NewsGroup dataset. The result shows a significant increase of running time as graph size increases. The reason resulting in the large increase in running time is mainly due to high time complexity involved in the computation of the power of a matrix and the SVD procedure.

7. CONCLUSIONS

In this paper, GraRep, a novel model for learning better graph representations is proposed. Our model, with our k-step loss functions defined on graphs which integrate rich local structural information associated with the graph, captures the global structural properties of the graph. We also provide mathematical derivations justifying the model and

establish the connections to previous research efforts. Empirically, the learned representations can be effectively used as features in other learning problems such as clustering and classification. This model comes with one limitation: the expensive computation of the power of a matrix and SVD involved in the learning process. Future work would include the investigation of efficient and online methods to approximate matrix algebraic manipulations, as well as investigation of alternative methods by employing deep architectures for learning low-dimensional representations in place of SVD.

8. ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their helpful comments. The authors would also like to thank Fei Tian, Qingbiao Miao and Jie Liu for their suggestions or help with this work. This paper was done when the first author was an intern at SUTD. This work was supported by SUTD grant ISTD 2013 064 and Temasek Lab project IGDST1403013.

9. REFERENCES

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

[2] D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. In *SODA*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

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

[4] J. A. Bullinaria and J. P. Levy. Extracting semantic representations from word co-occurrence statistics: A computational study. *BRM*, 39(3):510–526, 2007.

[5] J. A. Bullinaria and J. P. Levy. Extracting semantic representations from word co-occurrence statistics: stop-lists, stemming, and svd. *BRM*, 44(3):890–907, 2012.

[6] J. Caron. Experiments with lsa scoring: Optimal rank and basis. In *CIR*, pages 157–169, 2001.

[7] P. Comon. Independent component analysis, a new concept? *Signal processing*, 36(3):287–314, 1994.

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

[9] C. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[10] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *JMLR*, 9:1871–1874, 2008.

[11] M. U. Gutmann and A. Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *JMLR*, 13(1):307–361, 2012.

[12] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[13] C. Jutten and J. Herault. Blind separation of sources, part i: An adaptive algorithm based on neuromimetic architecture. *Signal processing*, 24(1):1–10, 1991.

[14] V. Klema and A. J. Laub. The singular value decomposition: Its computation and some applications. *Automatic Control*, 25(2):164–176, 1980.

[15] T. K. Landauer, P. W. Foltz, and D. Laham. An introduction to latent semantic analysis. *Discourse processes*, 25(2-3):259–284, 1998.

[16] O. Levy and Y. Goldberg. Neural word embedding as implicit matrix factorization. In *NIPS*, pages 2177–2185, 2014.

[17] K. Lund and C. Burgess. Producing high-dimensional semantic spaces from lexical co-occurrence. *BRMIC*, 28(2):203–208, 1996.

[18] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

[19] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. *EMNLP*, 12, 2014.

[20] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *SIGKDD*, pages 701–710. ACM, 2014.

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

[22] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Incremental singular value decomposition algorithms for highly scalable recommender systems. In *ICIS*, pages 27–28. Citeseer, 2002.

[23] J. Shi and J. Malik. Normalized cuts and image segmentation. *PAMI*, 22(8):888–905, 2000.

[24] A. Strehl, J. Ghosh, and R. Mooney. Impact of similarity measures on web-page clustering. In *Workshop on Artificial Intelligence for Web Search (AAAI 2000)*, pages 58–64, 2000.

[25] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. Line: Large-scale information network embedding. In *WWW*. ACM, 2015.

[26] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su. Arnetminer: extraction and mining of academic social networks. In *SIGKDD*, pages 990–998. ACM, 2008.

[27] L. Tang and H. Liu. Relational learning via latent social dimensions. In *SIGKDD*, pages 817–826. ACM, 2009.

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

[29] F. Tian, B. Gao, Q. Cui, E. Chen, and T.-Y. Liu. Learning deep representations for graph clustering. In *AAAI*, 2014.

[30] P. D. Turney. Domain and function: A dual-space model of semantic relations and compositions. *JAIR*, pages 533–585, 2012.

[31] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *JMLR*, 9(2579-2605):85, 2008.