University of Passau

doctoral thesis

# Graph Representation Learning for Social Networks

*Author:*
Fatemeh Salehi Rizi

*Supervisor:*
Prof. Dr. Michael Granitzer


*A thesis submitted in fulfillment of the requirements for the degree of Doctor of Natural Sciences*

*in the*

Chair of Data Science
Faculty of Computer Science and Mathematics

January, 2021

To my sister Elaheh.

# Abstract

Online social networks provide a rich source of information about millions of users worldwide. However, due to sparsity and complex structure, analyzing these networks is quite challenging and expensive. Recently, graph embedding emerged to map networked data into low-dimensional representations, i.e. vector embeddings. These representations are fed into off-the-shelf machine learning algorithms to simplify and speed up graph analytic tasks. Given the immense importance of social network analysis, in this thesis, we aim to study graph embedding for social networks in three directions.

Firstly, we focus on social networks at microscopic level to primarily encode the structural characteristic of users' personal networks so-called ego networks. These representations are utilized in evaluation tasks whose performance depends on relational information from direct neighbors. For example, social circle prediction and event attendance inference both need structural information from neighbors in social networks.

Secondly, we explore assessing the content of vector embeddings in terms of topological properties. This could be explained via two proposed approaches: 1) a learning to rank algorithm in which the model weights reveal the importance of properties at subgraph level (ego networks), 2) a regression model for direct approximation of network statistical properties at vertex level.

Thirdly, we propose extensions of graph embedding to capture sign or additional content of social networks. Users in social media often express their feelings and attitudes towards others which forms sentiment links besides social links. We design a joint objective function whose terms capture semantics of both social and sentiment links simultaneously. We also propose a multi-task learning framework for networks with attributes and labels by stacking autoencoders. The weights of the learning tasks are automatically assigned via an adaptive loss weighting layer.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Motivation

Online social networks nowadays contain a wealth of useful information about social interactions, relations, and user generated content. Social network analysis thus became an important tool for solving various real-world applications such as epidemic prediction [CCF08, GK04, PSV01, FMWFM11], natural disaster mitigation [NSJ12, BOF+97, Fre11], social influence dissemination [MF94, MF93, KKT03], information and trust propagation [JE10, BXM+13, RP11], crime and criminal detection [Bol14, HLJ15, OE12], personalized recommendations [QFZM13, GZC+09, SZ08], targeted advertising [YDCL06, MB09], social healthcare analysis [DA10, TY12], viral marketing [ML10, DTSS10], and academic network analysis [TZY+08, LMWDO10, DBS07].

To process the networked data, traditional methods mainly rely on handcrafted features, such as kernel functions [VSKB10], graph statistics (e.g. degree, clustering coefficient) [BCM11], carefully engineered features [LNK07], and probabilistic models [WSP07, HMK04]. However, these conventional algorithms mostly suffer from theoretical and practical difficulties such as high computational time and space. Recently, graph embedding (or network embedding) emerged to learn a mapping from raw input nodes to points in a low-dimensional vector space, so-called vector embedding (or representation). The goal is to optimize this mapping so that network properties are reflected in the geometric relationship between the learned vectors. These representations are then input as features into off-the-shelf machine learning algorithms whose speed and performance obviate the need for applying complex classification models directly on the original graph.

Graph embedding has been already applied to social networks with the aim of performing accurate and fast link prediction [GL16, OCP+16, ZLL+17, LZZ+17, LZZ+18], anomaly detection [HAMH16, PLL+18, LJSP18, SD14, YCA+18], node classification [YLZ+15, WCZ16, LHZC18, HYL17, DDS16] and clustering [TGC+14, CLX15, WCW+17, TNJ16, WXCY17]. Yet there exist many social network analysis tasks need to be solved by graph embedding.

A comparatively less explored but evidently important subject in social networks is the structural characteristic of users' personal networks so-called ego networks. In sociology, ego networks are fundamental to determine key facets of human behavior, such as trust, sharing of resources, and formation of communities. Therefore, ego network embedding can be used to simplify or speed up applications which need users' behavior and relations in small groups like social circle prediction or event attendance inference. Further analysis on social networks structure (e.g. centrality measures) has led to solve community detection [For10], shortest distance approximation [EJS76], link prediction [LNK07], node ranking [PA06] and recommendation [WLZ12]. For the sake of simplicity and efficiency, vector embeddings can be examined on retaining centrality measures or geodesic distance of nodes. Moreover, social networks can contain attributes, signs, and labels which initiate another set of applications such as signed link prediction, attribute inference, and edge label prediction. To further facilitate the latter tasks, vector embeddings containing rich semantics of network information can be leveraged via appropriate predictor models.

Given the immense importance of social network analysis, in this thesis, we aim to study and extend graph embedding for social networks in three directions. First, social networks are explored in a more fine-grained level to encode the subgraph around a focal node (ego) which ends in solving certain social network analysis tasks. Second, the recent graph embedding methods are probed if they preserve topological properties of social networks. Third, special forms of social networks are investigated if adapted embedding models can describe their data and structure.

## 1.2 Challenges

The major challenges in graph embedding for social networks are as follows:

- **Variability of neighborhood patterns:** Most of the current graph embedding approaches scrutinize social networks at macroscopic level, while there exist various neighborhood patterns at microscopic level, e.g. around an ego. According to Muhammad et al. [MVL15], the neighborhood pattern in ego networks can form eight distinct trends: star, complete, dense, strongly linked, linked neighbors, powerful ego node, strong ego neighbor, and less cohesive star. Ego network structure can help to identify influential spreaders [ACLG+16] or to organize users into different social circles [ML14]. Therefore, it is required to design subgraph embedding models in which personal social networks are effectively encoded. As such, the usefulness of subgraph embedding goes beyond the task of basic subgraph classification.

- **Incomplete understanding of learned embeddings:** Over the past few years, plenty of embedding methods have been proposed to simplify a range of different graph analysis tasks. These methods tend to learn as much topological information as possible, however what structure is exactly being captured

is currently not known. One aspect of this issue is to explore the interpretability of vector embeddings. Thus, further investigation on learned embeddings is needed to discover which network property (e.g. centrality measures, distance, density) is actually preserved there.

- **Diversity of social networks:** The data from real social networks can be heterogeneous or homogenous, weighted or unweighted, signed or unsigned, labeled or unlabeled, with associated content or multiple social roles. Furthermore, the tasks of social network analysis can vary widely, ranging from node-focused problems (e.g. node recommendation, link prediction) to graph-focused problems (e.g. community detection, graph generation). This diversity in data types and tasks requires designing different architectures to tackle specific problems.

## 1.3 Scientific Contributions

Based on the challenges defined above, we present the following scientific contributions:

- **Ego network embedding:** We make use of Paragraph Vector [MSC+13] to encode the neighborhood pattern of an ego network into a single vector called "Loc". The Loc vector then acts as a complementary feature to improve the performance of two downstream evaluation tasks namely social circle prediction and event attendance inference.

- **Exploring vector embedding:** We utilize RankSVM [Joa02] to examine the content of vector embeddings on retaining network centrality (e.g. closeness) at subgraph level (ego network). We also investigate the extent to which network centrality is preserved in embeddings at vertex level. A regression model fed with embeddings is used to directly approximate centrality values. Empirical experiments show each of the embedding techniques can potentially capture different semantics of network centrality in ego networks. Following the success of closeness approximation at vertex level, we attempt to compute the shortest distance between nodes via a neural regressor. We report our experimental evaluations on several benchmark datasets showing low distortion errors compared to the baseline methods.

- **Deep learning for special networks:** Given a network with sentiment and social links, we expand the objective of LINE [TQW+15] to learn representations observing both sentiment and social links. We validate the efficiency and effectiveness of our model by empirical evaluations on two real-world datasets. Another set of networks contain both attribute and labels which need to be modeled through joint learning paradigms. A possible approach is to stack autoencoders as an end-to-end model in which graph connectivity, node labels,

and available attributes are captured at the same time. The model conducts three embedding tasks optimizing three objective functions where weights for objectives are automatically learned. To ensure a fair comparison, we run several experiments on widely used datasets containing both node label and attribute.

## 1.4 Methodology

We use an experimental research methodology throughout this thesis. After related work in Chapter 2, we approach a scientific question in each chapter by conducting experiments with different variables. There exist different graph embedding methods which are compared using multiple datasets. All claims and conclusions are drawn from the conducted experiments and their respective results values.

## 1.5 Structure

This current chapter already introduced graph embedding for social networks and its inherent applications. Chapter 2 gives a general overview of graph embedding techniques for plain, heterogeneous, attributed, and signed networks. Therein, the recent embedding methods for social networks besides the usual applications are briefly described. This thesis consists of three major parts.

Part I includes Chapter 3 and Chapter 4 which introduce the use of ego network embedding for practical applications such as social circle prediction and inferring event attendance. These chapters describe the proposed models and datasets, discuss appropriate evaluation metrics, and conclude results and achievements.

Part II involves Chapter 5 and Chapter 6 which probe the content of different embeddings in regard to network topological features. This covers models and experiments for estimating vertex centrality and shortest path distance in social networks.

Part III studies special types of social networks in Chapter 7 and Chapter 8. Chapter 7 investigates how sentiment links along with social links can be encoded via a joint objective function. The proposed model is evaluated on two standard tasks namely signed link prediction and node recommendation. Chapter 8 presents stacked autoencoders for joint feature learning on network structure and additional attached content. The model includes an additional neural layer which assigns optimal weights to the multiple objectives.

Finally, Chapter 9 summarizes the findings and contributions of this work. The limitations of the proposed methods along with possible future works are briefly addressed at the end of the chapter.

## 1.6 Publications

The following papers are published in the context of this work:

- **Global and Local Feature Learning for Ego Network Analysis:** In this paper, we introduce the usage of Paragraph Vector [MSC+13] to model neighborhood patterns of ego networks and subsequently predict social circles. We assess our proposed method on ego networks of Facebook, Google+, and Twitter. This paper forms Chapter 3 of the thesis which explains the importance of ego network embedding specifically for social circle prediction

  > Fatemeh Salehi Rizi, Michael Granitzer, and Konstantin Ziegler. "Global and local feature learning for ego network analysis." *In 2017 28th International Workshop on Database and Expert Systems Applications (DEXA), pp. 98-102. IEEE, 2017.*

- **Properties of Vector Embeddings in Social Networks:** In this work, we probe vector embeddings in terms of vertex centrality via two proposed approaches. Our ranking approach unveils eigenvector and betweenness centrality as the most captured properties at subgraph level (ego networks). By a direct mapping, closeness centrality shows a more precise approximation at vertex level almost for all embedding techniques. Chapter 5 provides more detail on this work.

  > Fatemeh Salehi Rizi and Michael Granitzer. "Properties of vector embeddings in social networks". *Algorithms 10, no. 4 (2017): 109.*

- **Shortest Path Distance Approximation Using Deep Learning Techniques:** In this paper, we provide a direct mapping from pairs of vectors to real topological distances. Our experimental results on social network data demonstrate relatively low distortion errors in specific for shorter distances. Following the investigation on vector embeddings, this paper is expanded to study the distance between nodes in Chapter 6.

  > Fatemeh Salehi Rizi, Joerg Schloetterer, and Michael Granitzer. "Shortest Path Distance Approximation Using Deep Learning Techniques." *In 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), pp. 1007-1014. IEEE, 2018.*

- **Predicting Event Attendance Exploring Social Influence:** In this work, social influence is simply calibrated by network embedding techniques. A combination of textual and network structural features is deployed to make attendance prediction. Empirical evaluations on two event-related datasets reveal the impact of additional network features. This work is another example to show the importance of local neighborhood embedding in Chapter 4.

Fatemeh Salehi Rizi and Michael Granitzer. "Predicting event attendance exploring social influence." *In Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, pp. 2131-2134. ACM, 2019.*

- **Signed Heterogeneous Network Embedding in Social Media:** In this research, we integrate semantics of sentiment and social links via an extended version of LINE [TQW+15] which obtains embeddings in linear time complexity. Our empirical evaluations showed the generated embedding outperforms baselines in both signed link prediction and node recommendation tasks. This paper is the base of Chapter 7 in which we explain how to embed link heterogeneity in social networks.

  Fatemeh Salehi Rizi and Michael Granitzer. "Signed heterogeneous network embedding in social media." *In Proceedings of the 35th Annual ACM Symposium on Applied Computing, pp. 1877-1880. 2020.*

- **Multi-Task Network Embedding with Adaptive Loss Weighting.** In this paper, we stack autoencoders to build an end-to-end model (called JAME) capable of handling data integration in social networks. Experimental evaluations on real-world datasets demonstrate the effectiveness and efficiency of JAME compared to the baseline methods. Chapter 8 explains more detail about this paper.

  Fatemeh Salehi Rizi and Michael Granitzer. "Multi-Task Network Representation Learning with Adaptive Loss Weighting." *In 2020 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), pp. 1-5, IEEE, 2020.*

# Chapter 2

# Background and Related Work

This chapter provides background and overview on the recent neural graph embedding techniques and their applications in social network analysis. With the growth of online social networks such as Facebook and Twitter, graph analysis has become a radically novel area of data mining. Other than social networks, graphs in general provide a natural abstraction to represent the relationship between multiple entities. Graph analysis can solve practical applications in diverse domains like social science [MVW14], biology [Maa11, ZVTD+18], recommendation systems [LMY+12], crime detection [TGB11, FDMCF14], natural language processing [Uts15], and medicine [BGL11].

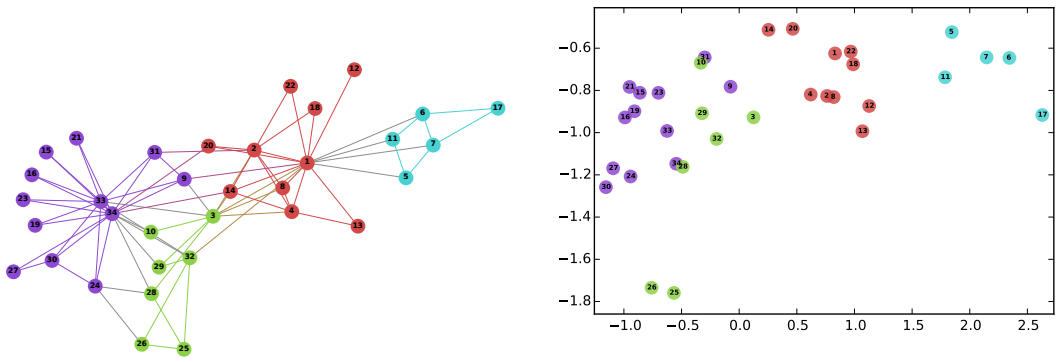
Typically, social networks reflect social interactions between people, biological networks form protein-to-protein iterations, citation networks show how publications are cited by authors in the same field, and word co-occurrence networks help for language modeling. A more extensive analysis on complex networks can lead to solve various applications such as friend recommendation in social networks, diseases prediction in protein interaction networks, identifying criminal groups in communication networks, and highly cited papers in citation networks. The latter tasks can formally fall into the category of link prediction [LNK07, GL16, OCP+16, BG17, TMKM18, SWRG19], node classifcation [BCM11, GL16, TQW+15, PARS14, WCZ16, BG17, TMKM18], graph clustering [Sch07, CZC+17], or node centrality calculation [MRV16].

However, with today's large-scale networks, performing graph analysis and visualization imposes serious challenges. For instance, most of the traditional graph algorithms run iteratively on the adjacency matrix which usually cause high computational costs in time and space [NJ, ZGL03]. An alternate solution is to use hand-engineered features created by network statistics and other measures [GER08] which is a labor process. Traditional parallel computing platforms such as Mapreduce [DG08] are not well suited for networked data, and the other graph analytic platforms like Pregel [MAB+10], Giraph [MSLH15] and Graphx [XGFS13] have limited efficiency for real-world networks with the scale-free property.

A possible solution for large-scale graph analysis is to map the networked data

into low-dimensional latent space, and then run off-the-shelf machine learning algorithms over those representations. This process of converting the networked data into vector space is known as graph embedding (or network embedding). Initially, researchers developed graph embedding algorithms inspired by dimensionality reduction techniques. They first build a similarity matrix based on the neighborhood structure, then embed that matrix into a low-dimensional vector space such that connected nodes are closer to each other. However, the majority of works based on this rationale suffer from scalability issues. For example, Laplacian Eigenmaps [BN01] and Locally Linear Embedding (LLE) [RS00] have time complexity of $O(|V|^2)$, where $V$ is the set of nodes. Therefore, obtaining scalable graph embedding techniques which leverage the sparsity of real-world networks has drawn great attention during the last decades.

The great success of neural networks for learning compact descriptors of data in language modeling [CS17, CW08, SBM12], speech and image processing [DYDA11, HOT06, HDY+12, GLO+16, KSH12, WWH+14] encouraged researchers to apply deep learning models to the networked data. Thus, neural graph embedding emerged with the goal of mapping nodes in the large-scale networks into latent vectors (or representations). A significant amount of research has been made in the past few years to encode different types of networks via nonlinear deep learning models. As an example, Figure 2.1 shows 2D representations of applying DeepWalk [PARS14] to the Zachary's karate club network [Zac77].




(a) Input: Karate Graph (b) Output: Representation

Figure 2.1: Graph embedding by DeepWalk [PARS14] on Zachary's Karate Club network [Zac77]. Different node colors show a modularity-based clustering on the input network. All the nodes with structural proximity are close to each other in the latent space.

Network embedding models are supposed to primarily preserve first-order, second-order, and higher-order proximity in the latent space, however this transformation is not straightforward. The real-world networks with millions of nodes are usually sparse, nonlinear, scale-free, dynamic, heterogeneous, singed, or with associated content. Thus, the need for designing scalable and flexible embedding

models which adapt to such situations is felt keenly in the graph analysis domain.

Over a few years, there have been a large category of graph embedding models for practical purposes. In this thesis, we focus on graph embedding models which leverage neural networks to finally mitigate social network analysis problems. In the following, we build a taxonomy of graph embedding techniques whose category is classified into plain networks, heterogeneous networks, signed networks, and attributed networks. Further, we discuss the recent applications of graph embedding in the context of social network analysis.

## 2.1 Definitions and Preliminaries

We start by definitions and preliminaries whose terminologies have been delineated similar to Wang *et al.* [WCZ16].

**Definition 2.1.1.** (Graph) A graph $G = (V, E)$ contains a collection of $V = \{v_1, \cdots, v_n\}$ vertices (a.k.a. nodes) and $E = \{e_{ij}\}_{i,j=1}^{n}$ edges. The adjacency matrix $A$ of graph $G$ is defined such that $A_{ij} = 1$ if node $v_i$ is connected to node $v_j$, else $A_{ij} = 0$. In the weighted graphs, $w_{ij}$ represents a non-negative weight associated with the edge between $v_i$ and $v_j$. For each node $v_i$, we denote the set of its direct neighbors as $N(v_i)$.

**Definition 2.1.2.** (First-order proximity) The first-order proximity between nodes $v_i$ and $v_j$ is measured by the weight $w_{ij} > 1$. The higher weight indicates the higher proximity.

**Definition 2.1.3.** (Second-order proximity) The second-order proximity of nodes $v_i$ and $v_j$ is determined by the similarity of two neighborhoods $N(v_i)$ and $N(v_j)$.

**Definition 2.1.4.** (Heterogeneous Network) A heterogeneous network $G = (V, E, \rho, \psi)$ consists of multiple types of nodes and edges. Each node $v_i \in V$ in $G$ is associated with a node type $\rho(v_i)$, and each edge $e_{ij} \in E$ is associated with an edge type $\psi(e_{ij})$. Function $\rho : V \mapsto T_v$ presents a node type mapping where $T_v$ is the set of node type, $|T_v| > 1$. $\psi : E \mapsto T_e$ is an edge type mapping function where $T_e$ denotes the set of edge types, $|T_e| > 1$.

**Definition 2.1.5.** (Attributed Network) An attributed network is formally denoted as $G = (V, E, X)$ where $V$ and $E$ denote the sets of nodes and edges respectively. $X = \{x_1, \cdots, x_m\}$ is a set of attributes associated with nodes to describe node properties. Each node $v_i \in V$ holds an attribute vector $[x_1(v_i), \ldots, x_m(v_i)]$ where $x_j(v_i)$ is the $j^{th}$ attribute value of the node $v_i$.

**Definition 2.1.6.** (Signed Network) A signed network is defined as $G = (V, E, \xi)$ where V and E denote the sets of nodes and edges respectively. $\xi : E \mapsto \{-1, 0, +1\}$ is a function which assigns $+1$ if the edge between $v_i$ and $v_j$ is positive (friend/trust), $-1$ if the edge is negative (foe/distrust) and 0, if there is no edge.

**Definition 2.1.7.** (Graph embedding) Graph embedding over $G = (V, E)$ is a mapping $f : v_i \mapsto \phi_i \in \mathbb{R}^d \ \forall i \in \{1, \ldots, |V|\}$ such that $d \ll |V|$, and the function $f$ preserves first-order, second-order, or higher-order proximity defined on the graph $G$.

## 2.2 Network Representation Learning

In this section, we build a taxonomy of graph embedding methods based on network types which reviews the recent and major works under each category. The embedding methods fall into four broad categories: 1) plain networks, 2) heterogeneous networks, 3) signed networks, and 4) attributed networks. We first describe the characteristic of each category, then provide a summary of representative approaches using the notation in Table 2.1

Table 2.1: Summary of notations.

| | |
|---|---|
| $G$ | Graphical representation of the data |
| $V$ | Set of nodes in the graph |
| $E$ | Set of edges in the graph |
| $A$ | Adjacency matrix, $\|V\| \times \|V\|$ |
| $d$ | Number of dimensions |
| $\phi_i$ | Embedding of node $v_i$ , $1 \times d$ |
| $f$ | Mapping function |
| $X$ | Attribute matrix, $\|V\| \times k$ |
| $Y$ | Label matrix, $\|V\| \times m$ |
| $S$ | Similarity matrix, $\|V\| \times \|V\|$ |

### 2.2.1 Plain Networks

Plain networks refer to homogeneous, undirected, static, and unsigned networks. The key and fundamental goal of the embedding process is to preserve the structural property of the original network. DeepWalk [PARS14], node2vec [GL16], and LINE [TQW+15] are pioneering works which employ neural networks for graph analysis tasks. DeepWalk and node2vec first conduct random walks to collect node pairs whose distance is in the range of a local vicinity. Then, they use the Skip-gram model [MSC+13] of word embedding which leverages a shallow neural network architecture to generate embeddings. To explore the input graph locally and globally, node2vec additionally involves Breadth-First Search (BFS) and Depth-First Search (DFS) during the random walk. Given a walk $v_1, \ldots, v_i, \ldots v_l$ with length $l$, Skip-gram maximizes the probability of neighbors in the walk, given the representation of the central node:

$$\max_{\phi} P(v_{i-w}, \ldots, v_{i+w} | \phi_i), \tag{2.1}$$

where $w$ denotes the size of context neighbors around $v_i$ and $\phi_i$ refers to the vector representation of the node $v_i$. The Skip-gram model ignores the ordering constraint, then Equation 2.1 is transformed to:

$$\max_{\phi} \sum_{-w<j<w} \log P(v_{i+j}|\phi_i). \tag{2.2}$$

The conditional probability of a node pair $P(v_j|\phi_i)$ is defined using the softmax function:

$$P(v_{i+j}|\phi_i) = \frac{\exp(\phi_{i+j}^T \phi_i)}{\sum_{k=1}^{|V|} \exp(\phi_k^T \phi_i)}. \tag{2.3}$$

The softmax computation at the output layer of Skip-gram runs over all nodes which costs time and memory. DeepWalk and node2vec apply two strategies to approximate softmax, hierarchical softmax [MSC+13] and negative sampling [MSC+13]. Hierarchical softmax constructs a binary tree in which leaf nodes present the words in vocabulary and the intermediate nodes present internal parameters. In this way, Equation 2.3 does not need to enumerate all the nodes but only a path from the root to the corresponding leaf. The optimization problem indeed narrows down to maximize the probability of a particular path in the tree. Given a path to leaf $v_i$ with the sequence of nodes $(b_0, b_1, \ldots, b_{\log(|V|)})$, where $b_0 = root$ and $b_{\log(|V|)} = v_i$, then Equation 2.3 becomes:

$$P(v_{i+j}|\phi_i) = \prod_{t=1}^{\log|V|} P(b_t|\phi_i). \tag{2.4}$$

The time complexity of the Skip-gram model is bounded to $O(w(d + d \log|V|))$, where $w$ is the context window size, $d$ is the number of dimensions, and $\log|V|$ is the time to build the hierarchical softmax over $V$ vertexes. The other idea is negative sampling which randomly picks nodes as noise instead of running a sum over all nodes. This means for two given neighbors $v_i$ and $v_{i+j}$, a noise node $v_k$ is sampled according to a uniform distribution $P_n(v_i) \sim \frac{1}{|V|}, \forall v_i \in V$ [MSC+13]. Then, the log-probability $\log P(v_{i+j}|\phi_i)$ is calculated as:

$$\log \sigma(\phi'^T_{i+j} \phi_i) + \sum_{k=1}^{K} \mathbb{E}_{v_t \sim P_n} \log \sigma(\phi'^T_k \phi_i), \tag{2.5}$$

where $\sigma(.)$ denotes the sigmoid function $\sigma(x) = \frac{1}{1+exp(-x)}$, and $K$ is the number of negative samples.

LINE [TQW+15] differently samples node pairs connected by edges multiple times considering the constant edge weights. The first-order proximity for each edge $(v_i, v_j)$ is defined via the joint probability $P(v_i, v_j)$, and the second-order proximity via the conditional probability $P(v_i|v_j)$. To maximize these probabilities,

the idea of negative sampling [MSC+13] is tied to the formulation whose terms draw away $K$ negative node pairs (non-existing edges) during training. In total, the time complexity of Skip-gram with negative sampling can be calculated approximately in $O(n)$ when $n$ is the size of training data [KK17].

DeepWalk is intended as the baseline of many recent works such as WalkLets [PKS16], struc2vec [FRS17], and many other subsequent works [PWZ+16, NLR+18, GL16, DCS17, DLTW18, CPHS18]. Walklets [PKS16] modifies the random walk strategy of DeepWalk by skipping over multiple nodes in the ongoing walk. This is performed for different skip lengths which resemble adjacency matrix $A$ powers up to a given order. Struc2vec [FRS17] preserves the structural role of nodes in the network through two steps. First, building a hierarchical multilayer graph, then running random walks over that synthetic graph which seamlessly collects samples for the Skip-gram learning.

Graphs inherently present a nonlinear structure which is a good fit for architectures like autoencoders with capability of learning nonlinear manifolds. SDNE [WCZ16] deploys stacked autoencoders and implements multiple layers of nonlinear functions to map the networked data into vector space. Indeed, the network structure property and sparsity is managed via the following joint optimization function:

$$\sum_{i=1}^{|V|} \|(\hat{A}_i - A_i) \odot \mathbf{b_i}\| + \alpha \sum_{i,j=1}^{|V|} A_{ij} \|\phi_i - \phi_j\| + \lambda \mathcal{L}_{reg}, \tag{2.6}$$

here $A_i$ is the $i^{th}$ row of the adjacency matrix and $\hat{A}_i$ is the reconstructed one. $\odot$ performs the Hadamard product and $\mathbf{b_i} = \{b_{ij}\}_{i,j=1}^{|V|}$ is used to handle the sparsity of the adjacency matrix. If $A_{i,j} = 0$ then $b_{i,j} = 1$, else $b_{i,j} = \beta > 1$. The parameter of $\beta$ controls the reconstruction weight of the nonzero elements in the input graph. $\phi_i$ and $\phi_j$ denote the feature representation of nodes $v_i$ and $v_j$ respectively. The first term preserves second-order proximity using stacked autoencoders, and the second term stands for first-order proximity applying Laplacian Eigenmaps. The parameter of $\alpha$ balances the weight of two terms and $\lambda$ adjusts the regularization term described in [WCZ16]. Overall, the model minimizes the reconstruction error to globally capture the network underlying structure. The time complexity of SDNE is proportional to $O(cdI|V|)$, where $d$ is the dimension size of embeddings, $c$ is the average degree of the network, and $I$ is the number of training iterations.

Motivated by Pairwise Mutual Information (PMI) in language modeling [LGD15, BL07], DNGR [CLX16] combines the random walk strategy with a deep autoencoder architecture. The idea is to first calculate the PMI matrix $M$ from the co-occurred nodes appeared in walks, then input the matrix to stacked autoencoders to obtain embeddings. The PMI matrix as input ensures that the autoencoder can capture higher-order proximity and aids robustness of the model in presence of noise in the graph. The objective function of the DNGR model is formulated as:

$$\arg\min_{\theta_1,\theta_2} \sum_{i=1}^{|V|} \|m_i - g_{\theta_2}(f_{\theta_1}(\hat{m}_i))\|^2, \tag{2.7}$$

where $m_i$ is the $i^{th}$ instance of the PMI matrix, and $\hat{m}_i$ is the reconstructed input data of $m_i$. The goal is to estimate $\theta_1$ and $\theta_2$ so as to minimize the distance between the PMI matrix and its reconstruction. The transformation functions $f_{\theta_1}$ and $g_{\theta_2}$ are responsible for encoding and decoding in the autoencoder respectively. The time complexity of DNGR is proportionate to the size of nodes $|V|$ in the network.

Lately, there has been dispersion efforts for applying convolutional neural networks (CNNs) [DBV16,HVG11] to large-scale networked data. Graph Convolutional Network (GCN) [KW16a] is a prominent technique which integrates two input matrices from feature matrix $X \in \mathbb{R}^{n\times k}$ with $k$ features, and the adjacency matrix $A$ over multiple layers. The input representations of nodes at the $l^{th}$ layer of graph convolution is denoted by $H^{(l-1)}$ and the output representations by the matrix $H^{(l)}$. The initial representations are naturally the original input features:

$$H^{(0)} = X, \tag{2.8}$$

which is input to the first GCN layer. At each layer, GCN applies transformations to the feature vector $x_i$ (of the node $v_i$) and averages that hidden representation with $v_i$'s neighbors. The joint update for all nodes becomes a simple sparse matrix multiplication:

$$H^{(l)} = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{(l-1)}W^{(l-1)}), \tag{2.9}$$

where $W_l$ is the trainable weight matrix at layer $l^{th}$, and $\hat{A} = A + I_n$ is the adjacency matrix with self-connections. $I_n$ is the identity matrix and $\hat{D}$ is a diagonal matrix that $\hat{D}_{i,i} = \sum_j A_{i,j}$. Intuitively, these steps smooth the hidden representations locally along the edges of the graph and eventually enforce similar predictions among close neighbors. The time complexity of the convolution operation is estimated in $O(|E|fd)$, where $|E|$ is the number of edges, $f$ is the number of convolution filters, and $d$ is the embedding dimension size.

GCN has been deployed as the base of many recent works such as Variational Graph AutoEncoder (VGAE) [KW16b], FastGCN [CMX18], and Parametric graph convolution [TNS18]. VGAE is an enhancement over GCN using variational autoencoders to learn latent representations for undirected graphs. FastGCN [CMX18] proposes an adaptive sampling scheme instead of fixed sampling. Nodes are sampled in each convolutional layer, then integral transformations are applied to those nodes under certain probability measures. This adaptive node sampling via normalized degrees can reduce variances and lead to enhanced performance. Parametric graph convolution [TNS18] generalizes a convolutional filter by controlling the influence of the filter size which improves the performance of the regular GCN.

GraphGAN [WWW+18] adopts Generative Adversarial Nets (GAN) [GPAM+14] to build a generator and a discriminator which act like minmax game. The generator approximates the distribution of connectivity patterns in the network and generates fake node pairs to fool the discriminator. The discriminator attempts to distinguish the fake vertex pairs by generator from the real ones. The goal of the discriminator is to maximize the probability of assigning correct labels to real and generated samples. While the generator minimizes the probability, the discriminator identifies a correct sample among the generated one. The sigmoid function is used to model the discriminator $D$ and the softmax function models the generator $G$. The goal of GraphGAN is to estimate $\theta_G$ and $\theta_D$ so as playing the following two-player minimax game with the value function $V(G, D)$:

$$\min_{\theta_G} \max_{\theta_D} V(G,D) = \sum_{c=1}^{V} \mathbb{E}_v \sim P_{true}(.|v_c)[\log D(v, v_c; \theta_D))] + \mathbb{E}_v \sim G(.|v_c; \theta_G)[\log(1 - D(v, v_c; \theta_D)]. \tag{2.10}$$

The generator $G(v|v_c; \theta_G)$ tries to generate nodes whose connections resembles the neighbors of $v_c$ by approximating the underlying true connectivity distribution $P_{true}(v|v_c)$. The discriminator $D(v, v_c; \theta_D)$ attempts to discriminate the true neighbors of $v_c$ from those generated by $G$. This is done via calculating the probability of existing edge between $v$ and $v_c$. The time complexity of GraphGAN in each iteration is proportional to $O(|V| \log |V|)$, where $|V|$ is the number of nodes.

NetGAN [BSZG18] combines GAN and LSTMs to generate graphs based on a random walk strategy. At training, a generator produces plausible random walks through an LSTM network, then a discriminator identifies fake random walks from the real ones. After training, a new graph is obtained by normalizing a co-occurrence matrix computed based on the random walks coming from the generator. DynGAN [MGHR] leverages generative adversarial networks and recurrent networks to capture temporal and structural information of dynamic networks. The evolution pattern of the network is captured in an adversarial manner while the model predicts node embeddings. KBGAN [CW17] proposes an adversarial learning framework to improve the performance of a wide range of existing knowledge graph embedding models.

### 2.2.2 Heterogeneous Networks

With today's networks, some graph analysis tasks demand the data to be modeled from heterogeneous networks [SLZ+16,SH13] with nodes and edges of different types. For example, nodes in a citation network can have three different types such as papers, authors, and venues which are connected by edges. Heterogeneous network embedding aims to unify the heterogeneous types of nodes and links into a shared low-dimensional space.

Metapath2vec [DCS17] extends DeepWalk [PARS14] to conduct random walks over the heterogeneous network to gather metapaths. A metapath is represented in the form of $v_1 \xrightarrow{R_1} v_2 \xrightarrow{R_2} \dots v_t \xrightarrow{R_t} v_{t+1} \dots \xrightarrow{R_{l-1}} v_l$, where $R = R_1 \circ R_2 \circ R_3 \circ R_{l-1}$ describes the composition relations between node types $v_1$ to $v_l$. These paths are supposed to capture both the structural and semantic relationship between different types of nodes. Similar to DeepWalk, the collected paths are fed to a heterogeneous Skip-gram model to learn the representation of nodes by maximizing the probability of heterogeneous co-occurred nodes. Formally, the goal is to estimate $\theta$ so as to maximize the likelihood of a dataset of node-context pairs:

$$\arg\max_{\theta} \sum_{v \in V} \sum_{t \in T_v} \sum_{c_t \in N(v)} \log P(c_t|v;\theta), \tag{2.11}$$

where $t$ denotes the node type, $N(v)$ is the set of neighbors for $v$, and $P(c_t|v;\theta)$ is the probability of node co-occurrence modeled by a softmax function.

HNE [CHT+15] is another effort for mapping multi-modal objects into a shared space such that the similarity between the objects is preserved. The input of the HNE model is a heterogeneous network with text-text, text-image, and image-image interactions. The CNNs structure is utilized as building blocks to learn image features while fully connected layers are used to extract discriminative representations for text data. HNE receives a pair of text documents from the left and processes in a left-to-right direction. The outputs from the embedding layer are the vectorized representation of available network objects in the common latent space. These are further channeled to a prediction layer which computes the loss by minimizing the distance between topologically connected objects:

$$\begin{aligned} \min_{U,V} \frac{1}{N_{II}} \sum_{v_i,v_j \in \mathcal{V}_I} L(x_i, x_j) + \frac{\lambda_1}{N_{TT}} \sum_{v_i,v_j \in \mathcal{V}_T} L(z_i, z_j) + \\ \frac{\lambda_2}{N_{IT}} \sum_{v_i \in \mathcal{V}_I, v_j \in \mathcal{V}_T} L(x_i, z_j) + \lambda_3 L_{reg}, \end{aligned} \tag{2.12}$$

where $U \in \mathbb{R}^{d_I \times r}$ and $V \in \mathbb{R}^{d_T \times r}$ are $r$-dimensional transformation matrices for the image and text domain. $\mathcal{V}_T$ and $\mathcal{V}_I$ represent two disjoint subsets of text and image domain, $N_{TT}$, $N_{TI}$, and $N_{II}$ correspond number of edges types for text-text, text-image, and image-image interactions respectively. The first loss term preserves the text to text similarity, the second term preserves image to image similarity, the third component stands for text to image similarity, and finally the last component is a regularizer $L_{reg} = \|U\|_2 + \|V\|_2$. The parameters $\lambda_1$, $\lambda_2$ and $\lambda_3$ are the three balancing parameters which control the emphasis among loss terms.

PTE [TQM15] exetends the idea of LINE [TQW+15] to further adopt networks with vertex heterogeneity. The heterogeneous network here is constructed by combining three bipartite networks: 1) word-word network, 2) word-document network,

and 3) word-label network. PTE optimizes a joint objective function which collectively embed the three bipartite networks into a low-dimensional space:

$$\min -w_{ij} \sum_{(i,j)\in E_{ww}} \log P(v_i|v_j) - w_{ij} \sum_{(i,j)\in E_{wd}} \log P(v_i|d_j) - w_{ij} \sum_{(i,j)\in E_{wl}} \log P(v_i|l_j), \tag{2.13}$$

where $v_j$ is a word node, $d_j$ is a document node, and $l_j$ is a label node. These three loss terms try to minimize the negative log-likelihood of co-occurred nodes in word-word, word-document, and word-label networks. The authors suggested two algorithms to perform the learning process, a joint training and a pre-training with fine-tuning.

Motivated by metapath2vec [DCS17], HIN2Vec deploys a shallow feedforward network that inputs a pair of nodes $(v_i, v_j) \in V$ to predict the relationship type $r \in \mathbb{R}$ between them. More formally, the objective function of HIN2Vec maximizes the prediction probability if a relationship exists between $v_i$ and $v_j$:

$$\max \sum_{v_i,v_j,r\in\mathcal{D}} \log O(v_i, v_j, r), \tag{2.14}$$

here $\mathcal{D}$ a dataset of node-node-type triplets, and $O(v_i, v_j, r)$ maximizes $P(r|v_i, v_j)$, if $R(v_i, v_j, r) = 1$, otherwise minimizes the probability. $R$ is a binary value that indicates whether $v_i$ and $v_j$ have a relationship $r$.

There exist some other lines of work such as TransE [BUGD+13] and ProxEmbed [LZZ+17] which basically aim at encoding rich interactions among flexible-typed nodes in heterogeneous graphs.

### 2.2.3 Signed Networks

Signed networks [LHK10b, TCAL16] are graphs with positive and negative signs on edges which usually determine friend or foe relationship between entities. Substantial research work has been carried out on signed network embedding to accurately reflect these relationships in the embedding space.

SIDE [KPLK18] extends the random walk strategy of node2vec [GL16] to capture sign and direction available in the sequence of nodes. The structural balance theory is applied to infer the sign of co-occurred pairs with non-existing edges. This theory states that users of a signed social network should be able to have their friends closer than their foes. To update the model parameters, SIDE follows Skip-gram with negative sampling:

$$\max \sum_{u,v \in \mathcal{D}} [\log P(u,v) - \sum_{j=1}^{k} \log P(u,v_j)] + \frac{\lambda}{2} L_{reg}, \tag{2.15}$$

where $\mathcal{D}$ contains co-occurred pairs, and $L_{reg}$ is a regularization term. The objective function tries to maximize the likelihood $P(u,v)$ between two positively connected nodes $u$ and $v$, i.e. $\text{sign}(u,v) > 0$, and to minimize the likelihood for negatively connected nodes. Formally, the likelihood $P(u,v)$ is defined as follows:

$$P(u,v) = \begin{cases} \sigma(W_u^{out}.W_v^{in} + b_u^{out+} + b_v^{in+}) & \text{if } \text{sign}(u,v) > 0 \\ \sigma(-W_u^{out}.W_v^{in} + b_u^{out-} + b_v^{in-}) & \text{if } \text{sign}(u,v) < 0 \\ \sigma(-W_u^{out}.W_v^{in}) & \text{if } v \text{ is a noise,} \end{cases} \tag{2.16}$$

where $W_u^{out}$ and $W_v^{in}$ are signed proximity terms, and $b^{out\pm}$ and $b^{in\pm}$ are bias terms. The latter part of the objective function acts as a regularizer of the bias terms $L_{reg} = (\|b^{out+}\|^2 + \|b^{out-}\|^2 + \|b^{in+}\|^2 + \|b^{in-}\|^2)$. The time complexity of SIDE can be estimated as $O(\gamma l|V|)$, where $\gamma$ is the number of walks per node and $l$ indicates the walk length.

SiNE [WTA+17] designs a deep learning architecture for signed network embedding again by exerting structural balance theory. For a given node triplet $(v_i, v_j, v_k)$ with edges $e_{ij} = 1$ and $e_{ik} = -1$, the similarity between $v_i$ and $v_j$ is larger than $v_i$ and $v_k$, i.e. $s(v_i, v_j) > s(v_i, v_k)$. SiNE consists of two deep networks in which one preserves the sign by structural balance theory and the other one captures the network structure. That means for $e_{ij} = 1$, nodes $v_i$ and $v_j$ should be closer in the vector space than $v_i$ and $v_k$. The following objective function maximizes the similarity for positively connected nodes:

$$\begin{aligned} \min_{\Phi,\phi_0} \frac{1}{C} [ & \sum_{(\phi_i,\phi_j,\phi_k) \in \mathcal{P}} \max(0, s(\phi_i,\phi_k) + \delta - s(\phi_i,\phi_j)) + \\ & \sum_{(\phi_i,\phi_j,\phi_0) \in \mathcal{P}_0} \max(0, s(\phi_i,\phi_0) + \delta_0 - s(\phi_i,\phi_j))] + \alpha L_{reg}, \end{aligned} \tag{2.17}$$

where $\Phi = \{\phi_1, \phi_2, \ldots, \phi_{|V|}\}$ is a set of vector representations for $V$ nodes, $\mathcal{P}$ and $\mathcal{P}_0$ are two sets of triplets, $C = |\mathcal{P}| + |\mathcal{P}_0|$ is the size of the training set, and $\delta$ is a threshold to regulate the similarities. $L_{reg} = \Re(\theta) + \|\Phi\|^2 + \|\phi_0\|^2$ is a regularization term to avoid overfitting, and $\alpha$ is a parameter to control the contribution of the regularizer. The key step of optimizing is to compute the gradient of $\max(0, s(\phi_i,\phi_k) + \delta - s(\phi_i,\phi_j))$ and $\max(0, s(\phi_i,\phi_0) + \delta_0 - s(\phi_i,\phi_j))$ with respect to $\Phi$ and $\phi_0$. Finally, the gradient descent algorithm is used to update all of the parameters.

SNE [YWX17] adopts a log-bilinear model [MK13] to encode sign and network structure at once. Given a path $h$, SNE predicts the representation of a target

node $v$ by linearly combining the representation of its context nodes. To involve the signed relationship between nodes, two signed-type vectors are incorporated into the log-bilinear model. Also, a scoring function $s$ measures similarity between the actual and predicted representations. The objective function of SNE is stated as:

$$\max \sum_{v \in V} \log \frac{\exp(s(v,h))}{\sum_{v_j \in V} \exp(s(v_j,h))}, \tag{2.18}$$

here the log-likelihood is modeled by a softmax function. The goal is to maximize the likelihood of a target node $v$ available in the path of nodes $h$.

Another work by Wang et al. [WATL17] called SNEA presents a network embedding method on signed attributed networks. The objective function of SENA consists of two components, one for modeling user attributes and another for signs using structural balance theory. During training, the joint objective function optimizes these two components simultaneously.

### 2.2.4 Attributed Networks

Today's real-world networks are often associated with additional features (e.g. profile attributes or textual contents) which provide rich semantic information. Therefore, it is desirable to have network embedding methods which encode the rich content of the network into more accurate representations. However, how to combine attributes with the network topology in network embedding arouses considerable research challenges.

TADW [YLZ+15] jointly embeds both network structural features and nodes' textual content by taking advantage of the DeepWalk approach. DeepWalk is shown as equivalent to factorizing the PMI matrix of vertex-context pairs. TADW uses a DeepWalk-derived matrix factorization and an inductive matrix completion [ND14] to learn final representations. Formally, the textual matrix $T \in \mathbb{R}^{|V| \times k}$ with $k$ features is integrated into embeddings by factorizing the PMI matrix $M \in \mathbb{R}^{|V| \times |V|}$:

$$\min_{W,H} \sum \|M - W^T HT\|^2 + \frac{\lambda}{2}(\|W\|^2 + \|H\|^2). \tag{2.19}$$

Here the first term represents the low rank matrix decomposition of $M$, and the second term is a regularizer. The concatenation of $W$ and $HT$ are used as $2d$-dimensional representations of vertices. The time complexity of TADW in each iteration is $O(n_0(M)d + kd|V| + d^2|V|)$, where $n_0(M)$ refers to the number of non-zero entries and $d$ denotes the low rank of $M$.

DANE [GH18] aims to jointly learn the network structure and attribute information through autoencoders composed of multiple nonlinear units. Given the adjacency matrix $A \in \mathbb{R}^{|V| \times |V|}$, the higher-order proximity matrix is defined as $M = A + A^2 + A^3 + \cdots + A^t$ with $t$ orders. The joint model receives the higher-

order proximity matrix $M$ and attribute matrix $Z \in \mathbb{R}^{|V| \times k}$ from two separate branches and maps them to a shared low-dimensional latent space. The hidden layers $H^M$ and $H^Z$ denote representations from the topological structure and attributes respectively. DANE preserves the semantic proximity of attributes by minimizing the reconstruction loss between the input $Z$ from encoder and the output $\hat{Z}$ from decoder. Similarly, the higher-order proximity is preserved by minimizing the distance between $M$ and the reconstructed matrix $\hat{M}$. Thus, the objective function for both attribute proximity and higher-order proximity in the network is formulated as:

$$\min \sum_{i=1}^{|V|} \|\hat{Z}_i - Z_i\|_2^2 + \sum_{i=1}^{|V|} \|\hat{M}_i - M_i\|_2^2. \tag{2.20}$$

DANE also preserves the first-order proximity and attributes proximity simultaneously by maximizing the probability $P_{ij}^M$ and $P_{ij}^Z$ between the node $v_i$ and $v_j$:

$$P_{ij}^M = \frac{1}{1 + exp(-H_i^M (H_j^M)^T)} \tag{2.21}$$

$$P_{ij}^Z = \frac{1}{1 + exp(-H_i^Z (H_j^Z)^T)} \tag{2.22}$$

The final latent representations $H^M$ and $H^Z$ are obtained by minimizing the following objective:

$$\min - \sum_{E_{ij}>0} \log P_{ij}^M - \sum_{E_{ij}>0} \log P_{ij}^Z - \sum_i [\log P_{ii} - \sum_{E_{ij}=0} \log(1 - P_{ij})]. \tag{2.23}$$

These two representations are then concatenated to build consistent and complementary information from the network structure and attributes. The time complexity is dominated by the computation of the similarity $H^M (H^Z)^T$ in each iteration which is bounded to $O(|V|^2)$.

DeepGL [RZA17] proposes a deep architecture which learns representations for nodes and attributes in a hierarchical manner. The input graph is first collapsed into subgraphs (graphlets) at different levels. Then, DeepGL learns a set of relational feature operations which are essential to generate the higher level features. Thus, each layer of the deep model combines features from lower order subgraphs to generate higher order subgraphs employing the relational feature operations. The formal optimization function of DeepGL is defined as:

$$\arg\max_{\phi_i \notin \Phi} [s(y, \phi_i) - \beta \sum_{\phi_j \in \Phi} s(\phi_i, \phi_j)], \tag{2.24}$$

where $\Phi$ is the current set of selected features, $s(.)$ is a similarity function, and $\beta$ controls the balance between maximizing relevance and minimizing redundancy. The objective function finds a set of features $\phi_i$ that maximizes the similarity to the label $y$ and minimizes the similarity to the set of selected features $\phi_j$. The complexity of DeepGL is proportional to $O(k(|V| + k|E|))$, where $k$ is the number of node features.

There exist other efforts that use deep architectures to jointly encode network connectivity and additional attributes. CAN [MLBZ19] is a co-embedding model which takes the advantage of both variational autoencoders and GCNs to encode the network structure and attributes. The model inputs the adjacency matrix $A$ and attribute matrix $X$, then outputs Gaussian distributions as latent embeddings separately for nodes and attributes. CANE [TLLS17] offers a context-aware network embedding model whose convolutional layers with max-pooling are applied over different regions of the network.

## 2.3 Graph Embedding for Social Networks

With the advent of graph embedding, social network analysis has been shifted to the deployment of classification and clustering models on low-dimensional latent representations. Typical examples include epidemic trend prediction [LAH07], online advertisement targeting [LZT15], personalized recommendation [STLS06], social healthcare [TY12], social influence analysis [PWX16], viral marketing [CWW10], and citation network analysis [DBS07, GZZ+13].

In social networks, users (nodes) are usually associated with semantic labels relevant to certain aspects about them, such as affiliation, interest, or belief. However, these networks are often partially or sparsely labeled due to the high cost of node selection and labeling. The main goal of node classification is to predict labels for unlabeled nodes by leveraging connectivity patterns of labeled ones extracted from the network structure. A common practice is to extract node features using graph embedding techniques, and then apply machine learning classifiers like support vector machine, Naïve Bayes, or logistic regression for prediction. Different from these steps, some more recent works [DDS16, HYL17, MBM+17] design end-to-end frameworks to combine the two tasks, so that the discriminative information inferred from labels can directly benefit the learning of network embedding.

Social networks are not always complete as some friendship links between users can be missing even if they are friends in real-life. The aim of link prediction is to infer the presence of new emerging links in the future based on the observed links and the network evolution mechanism [LNK07, AHZ11, Zho11]. A well-understood approach is first to learn network embeddings, then to apply logistic regression to predict the existence of links between unseen node pairs under homogeneous network settings [GL16, OCP+16, ZLL+17], or heterogeneous ones [LZZ+17, LZZ+18].

Another challenging task is to detect anomaly in social networks such as spam-

ming, fraud, or phishing nodes which occur rare or unexpected and deviate from the majority of normal users. Network embedding projects the discrete and structural information of the network into the latent space. Thereafter, the statistical or geometrical algorithms are used in measuring the degree of isolation or outlierness of network components [HAMH16, LJSP18, PLL+18].

Node clustering is another important social network analysis problem which partitions the network into a set of clusters (communities), so that nodes in the same cluster are more similar than those from other clusters. Most of the efforts have resort to network embedding techniques for node clustering as disjointed tasks. They first embed nodes to low-dimensional vectors, then apply clustering algorithms to group vertexes [TGC+14, CLX15, WCW+17]. Some others such as Tang et al. [TGC+14] and Wei et al. [WCW+17] define node clustering and embedding as a unified objective to generate cluster-induced vector embeddings.

More nuanced works seek to encode users' profile information through well-defined objective functions. Zhang et al. [ZYZZ17] propose User Profile Preserving Social Network Embedding (UPP-SNE) which leverages profile information to encode more meaningful embedding. Users' profiles differ from the textural features due to noisy, sparse, incomplete and topic-inconsistent structure. UPP-SNE attempts to filter out the noise and extract the key information from profiles by performing a nonlinear mapping guided by the network structure. Formally, an approximated kernel mapping [RR08] is used to construct user embedding $\phi_i$ from user profile features:

$$\phi_i = \frac{1}{\sqrt{d}}[\cos(\mu_1^T x_i), \ldots, \cos(\mu_d^T x_i), \sin(\mu_1^T x_i), \ldots, \sin(\mu_d^T x_i)]^T, \tag{2.25}$$

where $x_i$ denotes the profile feature vector of vertex $v_i$, and $\mu_i$ stands for the corresponding coefficient vector. Further, the objective of DeepWalk [PARS14] is used to supervise the learning of the nonlinear mapping and make user profiles and network structure complement each other:

$$\min_{\phi} -\log P(\{v_{i-w}, \cdots, v_{i+w}\} \setminus v_i | \phi_i), \tag{2.26}$$

where $\{v_{i-w}, \ldots, v_{i+w}\}$ is the set of context neighbors of the node $v_i$ within $w$ window size in the given random walk sequence.

Noise-Resilient Similarity Preserving (NSP) [QHW+19] is another embedding technique confined for social networks. A noise in social networks is a false link exists in the observed network or an actual link which is missing in the current observation. NSP first calculates a similarity index vector $S_v = \{s_1, \cdots, s_i, \cdots, s_\gamma\}$ consisting of $\gamma$ single similarity indices from [ZXZZ15]. For any node pair $(v_i, v_j)$, the similarity index vector is defined as $S_v(v_i, v_j) = \{s_1(v_i, v_j), \ldots, s_\gamma(v_i, v_j)\}$. Given $S_v$, NSP defines the comprehensive similarity index between nodes $v_i$ and $v_j$ as:

$$S(v_i, v_j) = \sum_{s_k \in S_v} \frac{s_k(v_i, v_j) \max(s_k) - \min(s_k)}{\max(s_k) - \min(s_k)}, \tag{2.27}$$

where $\min(s_k)$ is the minimum $s_k$ value of all node pairs in the network, and $\max(s_k)$ is the maximum value. Thereafter, NSP computes a correction matrix $C$ based on $S$ and adjacency matrix $A$ applying a noise reduction process:

$$C_{ij} = A_{ij} \times \frac{S_{ij}}{\text{avg}(S)} \times \alpha, \tag{2.28}$$

where $\text{avg}(S)$ is the average of all the elements in $S$ and $\alpha$ is a threshold parameter such that $0 < \alpha < 1$. A link between $v_i$ and $v_j$ in the observed network with small similarity $S_{ij}$ is regarded as a false link. To reduce the influence of such a false link, the original $A_{ij}$ is reduced to reflect the actual connection strength. Finally, NSP fits a non-negative basis matrix $M \in \mathbb{R}^{n \times d}$ into the objective that enforces the embedding matrix $U$ for preserving the actual relationship between nodes. The goal is to minimize the distance between the correction matrix and embeddings which leads to the following objective function:

$$\min_{M,U} \ \|C - MU^T\|^2, \ s.t. \ M > 0, U > 0. \tag{2.29}$$

Updating this equation makes $M \times U$ to be as close as possible to the correction matrix $C$ which ends up with noise-resilient vector embeddings. The complexity of solving Equation 2.29 is bounded to $O(d^2|V| + d|V|^2)$, where $d \ll |V|$ is the dimension size of embeddings. Thus, the overall complexity of the update rules of NSP can be resolve in $O(d|V|^2)$.

There exists another work called LATTE [MBZ19] whose deep neural model is specified for network alignment [KZY13], community detection [VL07], information diffusion [KKT03]. Besides preserving the network structure, LATTE adds some application-oriented objectives into the optimization steps so as to guarantee the learned embeddings can be used in multiple external applications.

# Part I

# Ego Network Embedding

# Chapter 3

# Social Circle Prediction

## 3.1 Introduction

With the advent of social networks, a lot of efforts have been exerted to characterize social graphs at macroscopic level [UKBM11, BK09, HKP12, TSWY09]. The idea is to extract structural characteristics of the network to mitigate analytic problems like leader node identification or outlier detection. However, in the anthropology literature, it is required to determine key facets of human behavior, such as trust, sharing of resources, and formation of communities by investigation on the microscopic properties of social networks [MC84]. One effective way to succinctly describe local structures is breaking up the network into smaller sub-networks called ego networks [ML14]. An ego network is defined as a portion of a social network formed of a given focal node (ego) who has direct social relationships with the other persons (alters). Typically, alters are categorized by the ego into different social groups (social circles) such as family members, colleagues, sport teammates, etc. Figure 3.1 depicts an exemplary ego with her alters grouped in four social circles. Social circles such as Google circles and Facebook custom lists are great means of privacy protection as provide control on information boundary.

Ego networks have been an important subject of work in sociology and anthropology in the last decades. The majority of works attempted to characterize fundamental properties of ego network evolution [WPZ+15, HZL+16, TG16, LWB18]. Another interesting property of ego networks is alters' connectivity patterns which differ based on network density and local topology [MVL15]. The recent studies [ALGPC16, ACLG+16, ALGPC14] show some social network properties such as information diffusion strongly depends on the structure of users' ego networks. In practice, social circles can contribute to solve applications including content filtering and group recommendation [ML14]. For example, an event organizer may better target potential participants by issuing invitations around a core group of known experts or a user wants to read the latest news in social networks from his colleagues instead of scrolling through entire latest news [QLM12]. However, the major drawback is to assign hundreds of existing friends into circles which imposes a tedious

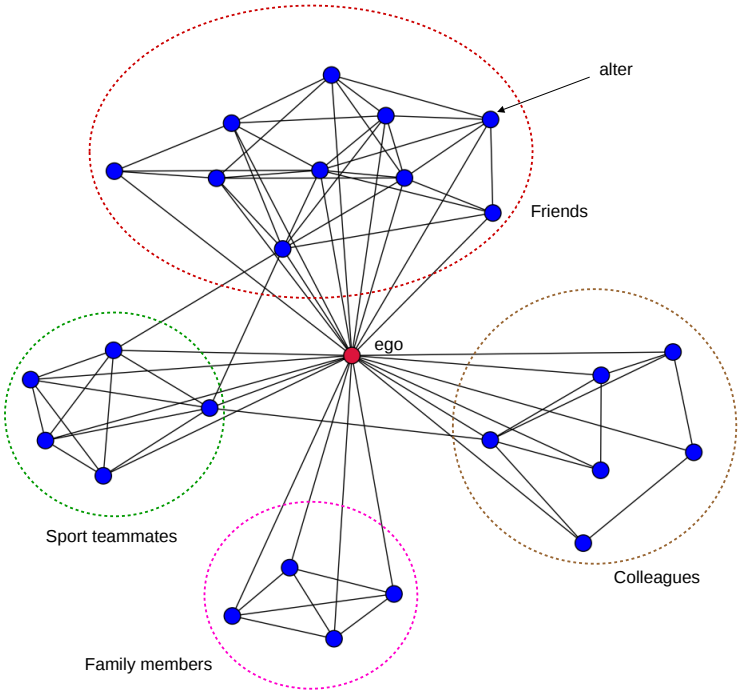



Figure 3.1: An ego network with four social circles.

and labor-intensive work.

Tracing this line of research [LYL+17, DLG15, WG13, ZN13, QLM12], most of the current social circle detection methods are unsupervised complex models exploring connectivity patterns in the original graph. The main semi-supervised approach is by McAuley et al. [ML14] which leverages both network structure and profile attributes to discriminate members from non-members in each circle. However, their probabilistic model still needs to inspect the complex connectivity of large-scale networks, and fails to refit the algorithm for a new added alter.

In graph analysis, usually, network structure is seized and deployed to make predictions about graphs. Therefore, graph embedding can be an effective means to tackle topological related tasks such as social circle prediction. Random walk based methods such as DeepWalk [PARS14], node2vec [GL16] reflect the global structure by walking over the entire network without restrictions. To encode more fine-grained structures like ego networks, we need to limit the walking distance to the diameter of ego networks.

Inspired by the successful completion of word embedding in graph analysis area [PARS14, GL16], we propose to make use of Paragraph Vector [LM14] to learn structural patterns within ego networks. Paragraph Vector projects both words and documents into a single semantic space and estimates word probabilities with the promising performance in word focused applications [AYGC16, DOL15, TKOT15, MT17]. The literature in word embedding presents two main structures for learning distributed representations: Continuous Bag-of-Words (CBOW) [MSC+13] and continuous Skip-gram [MSC+13]. More thoroughly review on word embedding shows

CBOW and Skip-gram make their performance different in specific situations. Skip-gram performs well with small training data and returns effective representations even with rare words or phrases. Whereas CBOW is faster to train with better accuracy in encoding more frequent words. For this reason, it is useful to examine CBOW in the context of graph embedding specifically for large-scale social graphs.

In this study, the key idea is to create artificial paragraphs by sampling sequences of nodes from ego networks and then to embed the paragraphs into a low-dimensional space. We also utilize CBOW to learn embeddings for nodes at global scale, where ego networks of the different size and density construct a social graph. We thus investigate the interplay of local and global structures within social networks to make the following contributions:

- We introduce local vector embedding (Loc) for egos in social graphs to complement the global representations, i.e. learning relations in a small vicinity instead of the entire graph.
- We apply global and local feature learning to the circle prediction problem.
- We demonstrate the effectiveness of local features compares through several experiments on real-world datasets.

In Section 3.3, we review the state-of-the-art for circle detection and prediction. In Section 3.3, we elaborate on local and global feature learning which is adopted for simplification of social circle prediction. In Section 3.4, we conduct empirical experiments to evaluate our proposed approach on three widely used datasets. Finally, we conclude and discuss the future work in Section 3.5.

## 3.2 Related Work

With the emergence of social networks, the problem of social circle detection and prediction has gained attention during the past few decades. Mcaulley et al. [ML14] formulate circle detection as a clustering problem on ego network of each user. The clustering algorithm receives the ego network structure and users' profile attributes as input. Like typical clustering algorithms, the model needs to calculate the pairwise similarity between alters as a function of common profile information and common edges. The initial empty clusters are filled while the algorithm assigns cluster labels to alters during multiple iterations. Although the unsupervised algorithm is quite effective to identify social circles, it is not particularly efficient for small networks with a few number of egos.

Yang et al. [YLL+14] present a multi-view clustering algorithm that extracts multiple quantitative features from users' ego networks. The idea is to integrate three views from structural, content, and interaction features. That means friends with common neighbors who share similar opinions more likely to fall in the same

circle. Thereafter, they interact within circles rather than cross circles. These observations are extracted as multiple quantitative features from users' ego networks to build the computational clustering model. However, such an approach heavily relies on users' generated content and frequent interactions which are not always publicly available.

Petkos et al. [PPK15] make use of Latent Dirichlet Allocation (LDA) for social circle generation which is typically applied for topic detection in documents. In ego networks, ego's friends resemble the documents and social circles are interpreted like produced topics. LDA is adapted to take into account both profile properties of friends and social links between them. Indeed, the vocabulary processed by LDA contains two types of elements, a set of properties and a set of user ids. Thus, network connectivity is jointly modeled along with profile property distribution in each circle. LDA allows a simpler model and a more accurate clustering, however it still suffers from high runtime complexity.

Tang et al. [TLXW14] take advantage of Bayesian Network (BN) to model social circle identification as a classification problem on a user's ego network. First, the social network data is transformed to become more suitable for Bayesian modeling. Second, an initial Bayesian network of egos is constructed using MMHC algorithm [TBA06] with proper nodes as parents and children. Lastly, by leveraging a carefully designed threshold, the BN model is used for accurate social circle prediction. The BN model outperforms Naïve Bayes, IBL, OneR and J48, however the computation for social network transformation is still costly.

## 3.3 Approach

Given a social graph $G$ with $m$ egos $\{u_1, u_2, \ldots, u_m\} \in U \subseteq V$, each ego $u$ with his alters compose an ego network $G_u$. We denote a set of $l$ alters $\{v_1, v_2, \ldots, v_l\}$ for the ego $u$ by $A_u \subseteq V$. These sets of alters in different ego networks may overlap. Below we explain how to encode the topological characteristics of the original social graph (consists of ego networks) at global and local scales.

### 3.3.1 Glo: Global Representations

Inspired by DeepWalk, we generalize the idea of CBOW [MSC+13] from word representation learning to network embedding, by using random walks to collect node contexts. More formally, given a random walk sequence with a pivot node $v_t$, we maximize the probability of observing the center node given the neighborhood context:

$$P(v_t | v_{t-w}, \ldots, v_{t-1}, v_{t+1}, \ldots, v_{t+w}), \tag{3.1}$$

where $w$ is the window size which restricts the size of node context. As such, a

mapping function Glo: $v \in V \rightarrow \mathbb{R}^{|V| \times d}$ converts nodes into vector representations where $d \ll |V|$ is the embedding size.

### 3.3.2 Loc: Local Representations

Given an ego $u$, we first conduct random walks over the subgraph $G_u$ to generate a synthetic paragraph called ego-walk. This means an ego-walk is built by a stream of short random walks started at every $v \in A_u \cup \{u\}$. We then apply the Distributed Memory Model of Paragraph Vectors (PV-DM) [LM14] to learn compact vector representations for each ego. PV-DM extends CBOW by assigning a unique id to each paragraph and updating the corresponding paragraph vector simultaneously with the other word vectors. Formally, given the ego-walk of ego $u_i$ with a center node $v_t$ we aim to maximize the average probability:

$$P(v_t|u_i, v_{t-w}, \ldots, v_{t-1}, v_{t+1}, \ldots, v_{t+w}), \tag{3.2}$$

where $w$ is the window size and $u_i$ is a unique ego id. We thus derive a mapping function Loc: $u \in U \rightarrow \mathbb{R}^{|U| \times d}$ which transfers an ego network into a coordinate vector with size $d \ll |V|$. Paragraph Vector indeed takes one step further as it not only trains node embedding but also ego network embedding directly. As shown in Figure 3.2, each ego is mapped to a unique vector represented by a column of the ego matrix $D$. Each alter is also mapped within a column of the matrix $W$. At the training phase, the ego vector and alter vectors are concatenated to predict the next alter in the ego-walk.

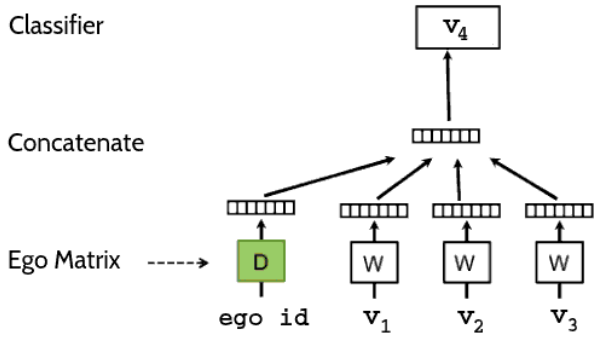



Figure 3.2: PV-DM for learning ego vectors. The concatenation of an ego vector with the context alters $v_1$, $v_2$, $v_3$ is used to predict the next alter $v_4$.

We now study the problem of automatic social circle prediction exploiting the obtained global and local features.

### 3.3.3 Social Circle Prediction

We formulate the problem of circle prediction as a multi-label classification task on a new added alter into the social graph. In accordance with Leskovec et al. [LM12], we also leverage users' profile information to improve the prediction task. The personal profiles including attributes are a rich source of information may reveal some hidden relationships in social networks. For example, if an ego and his alter both study at the same university most likely this alter belongs to the university circle. Accordingly, we involve profile similarity between alter and ego in addition to network topological features which possibly leads to more effective circle prediction. To formalize the intuition, we define the profile similarity vector between the ego $u$ and alter $v$, $\mathrm{Sim}(u, v) = (b_1, \ldots, b_k)$ as follows:

$$b_i = \begin{cases} 1 & \text{if } u.\,\mathrm{feat}_i = v.\,\mathrm{feat}_i, \\ 0 & \text{otherwise.} \end{cases} \tag{3.3}$$

Where $u.feat_i$ denote $i^{\text{th}}$ profile feature of the ego $u$, and $v.feat_i$ is $i^{\text{th}}$ profile feature of the alter $v$. Therefore, we encode profile attributes into pairwise binary vectors $\mathrm{Sim}(u, v) \in \{0, 1\}^k$ where $k$ is the number of profile features.

To design our predictor, we choose neural networks due to their ability to learn complex mappings between input and target spaces. In particular, feedforward networks has showed computational efficiency and empirical evidence supporting its generalizations for various problems [WCWW15]. Here we need a multi-label classification setting as a few alters belong to multiple social circles. We thus use a shallow feedforward network with the following possible inputs:

- where the model input is concatenation ($\oplus$) of global and local representations:
  - **LocGlo:** $\mathrm{Loc}(u) \oplus \mathrm{Glo}(v)$
  - **GloGlo:** $\mathrm{Glo}(u) \oplus \mathrm{Glo}(v)$
- where the model input is concatenation ($\oplus$) of network embedding and profile similarity vector:
  - **LocGLoSim:** $\mathrm{Loc}(u) \oplus \mathrm{Glo}(v \oplus \mathrm{Sim}(u, v)$
  - **GloGloSim:** $\mathrm{Glo}(u) \oplus \mathrm{Glo}(v) \oplus \mathrm{Sim}(u, v)$

Overall, the architecture of our classifier is described as follows:

- **Input layer:** A set of real-valued features which were described above.
- **Hidden layer:** A hidden layer with ReLU activation units.
- **Output layer:** Activation units same number as social circles with softmax function.

- **Optimizer:** RMSprop with adaptive learning rate method that already showed outstanding success in practice [TH12].

## 3.4 Experiments

This section describes our extensive experimental evaluations with the aim of examining constructed local and global features. We first introduce datasets used in these experiments, then compare model performance to the baselines in view of different features.

### 3.4.1 Datasets

We select a set of widely used networked datasets (Facebook, Twitter, Google+) with circle labels available on SNAP collection [ML14]. Table 3.1 describes the statistics of the datasets used in our experiments. Each social network dataset consists of nodes, social links, circles, and profile attributes. We publicly shared our reference code and datasets at `https://github.com/fatemehsrz/Ego-network_Embedding`.

Table 3.1: Statistics of the datasets.

| | | **Facebook** | **Twitter** | **Google+** |
|---|---|---|---|---|
| nodes | $\|V\|$ | $4,039$ | $81,306$ | $107,614$ |
| edges | $\|E\|$ | $8,8234$ | $1,768,149$ | $13,673,453$ |
| egos | $\|U\|$ | 10 | 973 | 132 |
| circles | $\|\mathcal{Y}\|$ | 46 | 100 | 468 |
| features | $k$ | 576 | 2273 | 4122 |

### 3.4.2 Experimental Setup

As the first step, we transform social networks into sets of sequences by multiple random walks similar to [PARS14]. Then, word2vec of gensim [ŘS10] is applied to the gathered sequences with the aim of deriving accurate coordinates. Following the baseline settings [GL16, TQW+15, PARS14], we set the embedding size $d = 128$, walk length $l = 10$, and context length $w = 10$. In a similar manner, we generate ego-walks which are sequences of nodes limited to one ego network, then apply doc2vec of gensim [ŘS10] to encode artificial paragraphs. For example, for the Facebook graph with 10 egos, we build a node corpus with 10 ego-walks.

The second step is to build profile similarity vectors based on the formulation described in section 3.3.3. Our datasets contain profile features which vary for each ego, hence we select $k = 100$ important features including occupation, education, gender, hometown, birth date, languages, location, and work together with their subbranches.

In the third step, we construct feature matrices by concatenation of network embeddings and similarity vectors. In detail, our input matrices are shaped considering the different combination of features where $X_{\text{LocGlo}} \in \mathbb{R}^{2d}$ and $X_{\text{GloGlo}} \in \mathbb{R}^{2d}$ contain only the network structure, $X_{\text{LocGloSim}} \in \mathbb{R}^{2d+k}$ and $X_{\text{GloGloSim}} \in \mathbb{R}^{2d+k}$ integrate profile features in addition.

In the multi-label setting, each of the alters is assigned to one or more social circles from a finite set $\mathcal{Y}$. We select 70% of alters as our training set and the remaining 30% as the test set. The batch size for all datasets is set to 32, and we run the model over 50 iterations.

### 3.4.3 Results

The performance of social circle prediction on Facebook, Google+, and Twitter is reported by F-score same as the baseline [ML14]. We validate the obtained results through 10-fold cross-validation. Table 3.2 demonstrates the average performance for social circle prediction across different folds. The maximum mean score within each scenario of feature combination is marked in bold and the standard deviation after cross-validation is less than 0.01.

As the first observation, replacing global representation with the local one improved the performance of the circle inference. The main reason for this advancement lies at the relevance of Loc representation that contributes more accurate information of egos' structures. While Glo representation holds the information mostly about global positioning of egos in the network.

Further experiments support our assumption that additional profile similarity features improve the performance over all datasets. In practice, we find our results very competitive to the baseline method [ML14] preforming better on the Facebook dataset. It is worth noting that our model spends only 5 minutes on network embedding and circle prediction for Facebook, however McAuley's approach takes up 1 hour over $1,000$ nodes [ML14].

Table 3.2: Social circle prediction performance reported by F-score.

| **Features** | **Facebook** | **Twitter** | **Google+** |
|---|---|---|---|
| GloGlo | 0.37 | 0.46 | 0.49 |
| LocGlo | 0.42 | 0.50 | 0.52 |
| GloGloSim | 0.40 | 0.49 | 0.51 |
| LocGloSim | **0.45** | 0.53 | 0.55 |
| $\Phi^1$ [ML14] | 0.38 | **0.54** | **0.59** |

## 3.5 Conclusion

Ego networks have been proposed to study several aspects of human behavior and social circles have emerged for privacy protection [SKLD14, SLKD12]. A major drawback is the usability problem; how to assign hundreds of alters to a few social circles. The problem of social circle perdition has been studied via direct characterization of network connectivity and profile information. Whereas, our current investigation focused on performance improvement in circle prediction with the help of neural networks. To this end, we employed deep language modeling techniques to transform the ego network structure into latent representations at local and global scales. These representations along with profile similarity vectors were fed into a feedforward network to predict circles of a new added alter in a more efficient running time.

Despite improvements on the performance, our study is restricted in a few directions. Firstly, we need the prior knowledge of egos for a new added alter to the social graph. However, from a practical point of view, we desire an algorithm that is able to make predictions on an alter without knowing her ego. Secondly, we follow the random walk strategy with equal probability of stepping in all directions which is not able to accurately model distinct neighborhood patterns. To properly capture prototypical patterns, e.g. star, we would need a finer control policy on walking transitions to nearest neighbors. Following that, we need specific evaluation tasks to examine embeddings on prototypical patterns, rather than typical classification tasks. Lately, Sub2vec [AZRP18] attempted to use Paragraph Vector for subgraph embedding with a more precise model and evaluation tasks. Our next chapter describes the usage of ego network embedding in event attendance prediction.

# Chapter 4

# Event Attendance Prediction

## 4.1 Introduction

Today's social networks have become a major channel for communication and information dissemination especially the spread of large public events. One interesting analytical scenario is to explore posts and comments to predict users' actual attendance to a certain event. Understanding people's behavior and feelings in events can offer valuable insights for effective event organization, community development, event recommendation, and mobility management.

Event participants often post their feelings, opinions, or experiences on social media in which information can quickly flow from one individual (or community) to another one. This information propagation may affect users' future decisions on what event and where to attend. The phenomenon of influence in online social networks has been widely studied in political elections [BFJ+12], obesity propagation [CF07], marketing and advertising [KKT03], and innovation adoption [TSWY09]. Some of the previous research work [DYM+14, ZL17] has shown people intend to participate in events with their friends or family members who are already linked in social media. Thereafter, users can propagate social influence within small groups whose opinions affect others' event attendance. However, there have been little work involving social influence to infer event attendance [ZL19, ZL17] which are limited to Event-Based Social Networks (EBSNs).

Interestingly, the network topological pattern (e.g., dense, sparse) can explain how the social influence spread over the time [KW06, CG16]. Thus, we speculate that the network structure of direct friends plays a role to encourage users for event attendance. This small group connectivity can be modeled via graph embedding algorithms.

To extract the network structure, we apply some of the recent graph embedding methods such as node2vec [GL16], HARP [CPHS18], Poincarè [NK17], and Loc [RGZ17] which have already shown prominent performance in graph analytic tasks. node2vec follows word2vec [MSC+13] and confirmed to be effective in

a wide variety of graph analysis tasks [GF17]. Later node2vec is extended by HARP [CPHS18] which first collapses the original network into a series of successively smaller graphs, then recursively embeds coarsened graphs to the larger ones. HARP explores underlying local and global structures via star and edge collapsing which eventually results accurate embeddings. Additionally, the hierarchical nature of social networks [MCP+13] motivates to apply Poincarè [NK17] since the hyperbolic space is an appropriate representation for the hierarchical data. Hyperbolic embedding excels in capturing both node similarity and node hierarchy, hence, is expected to encode more effective embeddings. Loc [RGZ17] differently limits the random walk range to the user's personal network with the aim of obtaining a more fine-grained structure stashed in embeddings.

Nowadays, Twitter presents one of the most popular platforms in which users express their emotions, opinions, and views about public events. Twitter data analysis and mining has gained immense importance in the recent years due to a wide variety of real-world applications. For instance, sentiment analysis in marketing to know how the public reacts to products, in politics to determine views of people regarding specific situations, and in risk prevention to detect if some people are being attacked or harassed. However, there exists little work focused on Twitter data analysis to infer users' actual attendance at the large public events.

To the best of our knowledge, only Lira et al. [dLMO+17] explore non-geotagged tweets from two music festivals to infer event participation. They extract three types of features: text features, temporal features, and social features. Text features refer to the textual content of the post modeled by Bag-of-Words. Temporal features present the time of the post in days with respect to the event. Social features describe the social profile of the posting user with the number of followers, number of followees, and the ratio between them. However, the authors totally neglect the impact of friends' influence on event participation and in contrast opt out of it.

In this chapter, we propose to extend the work by Lira et al. [dLMO+17] in which textual features are augmented to predict the attendance. We additionally encode Twitter network data by graph embedding and aggregate these new features with the former textual features. In summary, we make the following contributions:

- We extract network topological features by graph embedding to inspect the impact of social connections on event attendance.
- We investigate how different embedding techniques perform to predict event participation.
- We show additional network features lead to better performance compared to the state-of-the-art work.

The remaining of the chapter is organized as follows. Section 4.2 quickly reviews the relevant works for predicting event attendees. Section 4.3 describes preliminaries and the proposed method. Section 4.4 presents our evaluation results on two event-related datasets. Finally, Section 4.5 concludes the chapter.

## 4.2 Related Work

Social networks like Twitter widely reflect public events, hence become a rich source of information for event data analysis. Recently, there has been growing research interest in developing tools for inferring event participants exploring social media. Lira et al. [dLMO+17] collect textual and temporal features from non-geotagged tweets for attendance prediction, yet they ignore the impact of friends influence on users' decisions.

There exists another line of research in which geotagged posts in social media are probed to infer event attendance. Magnuson et al. [MDM15] associate each tweet with a real-world event given geolocation tags. Further, each user will have an interest profile about the events that she already attended. The list of events is gathered via a web crawl of Eventbrite[1], a popular event organization website with a database of past and upcoming events. Attended users have been identified through sentiment analysis of users' opinions posted on social media. The intensity of the user's sentiment is mapped into a rating number which reflects the interest of that user in the event. The final model generated from the above data provides geographic recommendations based on the time and location of the user. Botta et al. [BMP15] attempt to estimate the number of event participants at a given time by analyzing mobile sensors and geolocated tweets. A linear regression separates attendees and non-attendees relying on similar relationships in SMS activity, Internet activity, and Twitter activity.

Event-Based Social Networks (EBSNs) such as Meetup[2] and Sched[3] are the newly emerging platforms to publish events online and attract others to attend offline. Usually, EBSNs provide groups with specific themes such as hiking, writing, or health and try to encourage group members to attend real-world events. There exists an abundance of studies aim at finding the interest of a user by analyzing the user's past behaviors available in EBSNs. Authors in [ZZC15] explore EBSNs to accumulate three sets of features including semantic, temporal, and spatial features. Semantic features indicate how frequently users have attended similar events in the past. Event similarity measures are used to identify former relevant events. Temporal features refer to the users' temporal preference like date and time, while spatial features obtained from users' location preference. A set of classifiers such as logistic regression, J48 decision tree, and Naïve Bayes are fed with the collected features to predict the attendance.

An EBSN does not only contain online interactions like other social networks but also includes offline social interactions captured from offline activities. Liu et al. [LHT+12] analyze real data collected from Meetup to investigate network properties such as heavy-tailed degree distribution and strong locality of social interactions. An extended version of the Fiedler method [FS14] is deployed to incorporate online-

[1] https://www.eventbrite.com/
[2] https://www.meetup.com/
[3] https://sched.com/

offline heterogeneity of EBSN into community detection process. Due to the short life-time, recommending events can significantly differ from the usual recommendation problems like movies or places. Event recommendation is valid after creating the event online and before the event starts which leads to a cold start problem. The authors design multiple diffusion patterns to capture information flow within the heterogeneous EBSNs. These patterns take the community structure into account to yield the best prediction performance.

Social influence in EBSNs groups plays a role when individuals decide to attend a particular event. Zhang et al. [ZL17] propose a reconstructed propagation network to model the role of social influence on event popularity prediction. This is beneficial to public concerns track and decision-making for organizations to know event popularity. A set of contextual features (spatial, group, temporal, and semantic) and group-based features are fit into the Classification and Regression Tree (CART) [Loh11] for predicting event popularity. These events are organized by different social groups on EBSNs. In [DYM+14], authors model the social relationship between a user and a host (event holders) to infer the attendance. The activity host sends invitations to her followers, hence, if a user responds to the host, most likely she will attend the activities organized by that host. The collected features in this work including content preference, context (spatial and temporal), and social influence are finally fed into a Singular Value Decomposition with Multi-Factor Neighbor-hood (SVD-MFN) to predict the attendance.

Compared to these approaches we do not specifically deal with location-based social networks (EBSNs) but instead focus on popular social networks (e.g. Twitter) in which events are more publicly and frequently reflected.

## 4.3 Approach

Given a social graph $G = (V, E)$, we aim to predict whether a user $u \in V$ will attend event $e$ or not. We utilize two types of features to infer the user actual attendance:

- *Textual* features are extracted from the textual content posted by the user $u_i$. We employ Bag-of-Words [Har54] which is a common and straightforward numerical representation for short texts [FT]. Given tweets, we first remove emoticons, and Twitter specific stopwords to store a stemmed version of keywords (unigrams and bigrams). We thus create an array of size $|T|$ (vocabulary with finite small size) with 1s in the position of the words belonging in this tweet and 0s everywhere else. Thus, we would have a vector $\tau_i \in \{0, 1\}^{|T|}$ representing each posted tweet.

- *Network* features describe the neighborhood pattern around a user $u$ in $G$. Graph embedding models are used presenting the neighborhood structure

around the user $u_i$ in real-valued vectors $\phi_i \in \mathbb{R}^d$, $d \ll |V|$. We take the following graph embedding techniques into consideration:

- node2vec [GL16] encodes the structure around a node $u$ by maximizing the probability of observing the $w$ nodes before and after $u$ in the random walk.
- HARP [CPHS18] first collapses the graph $G$ into $l$ coarsened graphs $(G_0, \cdots, G_l)$ in a hierarchical manner. Then, node2vec is recursively applied from $G_l$ to $G_0$ such that embeddings from $G_i$ initialize the model for $G_{i+1}$.
- Poincarè [NK17] represents a geometrical ball to encode hierarchical patterns in complex networks. Given a node pair $(v_i, v_j)$ within the radius of $r$, the goal to maximize the following probability:

$$P((v_i, v_j) = 1|\phi_i) = \frac{1}{e^{(\delta(v_i,v_j)-r)/t} + 1}. \tag{4.1}$$

  The parameter $t$ specifies the steepness of the logistic function in optimization, and $\delta$ is the Poincarè distance as:

$$\delta(v_i, v_j) = \arccos\left(1 + 2\frac{\|v_i - v_j\|^2}{(1 - \|v_i\|^2)(1 - \|v_j\|^2)}\right). \tag{4.2}$$

- Loc [RGZ17] first conducts random walks in a limited range around a central node (ego), then inputs the generated walks into the Paragraph Vector [LM14] which finally results a single ego vector.

We now model the prediction task as a binary classification performed by a feed-forward neural network. The model receives a combination of textual and network features to predict users' attendance. Formally, we define a function $f$:

$$f : \tau_i \oplus \phi_i \mapsto \{0, 1\}, \tag{4.3}$$

that maps concatenation ($\oplus$) of $\tau_i$ (textual features) and $\phi_i$ (network features) to infer $u_i$ attendance. The binary value of 1 stands for attendance and 0 for non-attendance. Our predictor is composed of an input layer, a hidden layer, and an output layer with a single neuron. The size of the input layer depends on the size of vector embeddings $d$ and vocabulary of tweets $|T|$. We set the rectified linear unit (ReLU) [NH10] as activation functions of the hidden layer. ReLU is widely used in deep neural networks for its computational advantages in training [GBB11, MdLFH19]. The number of neurons in the hidden layer is the mean number of neurons used in the input layer [Hea08]. The output layer is functioned by a sigmoid [KO11] unit as for binary classification. Our optimizer is Adam [KB14] due to its robustness and a few parameters to tune.

## 4.4 Experiments

In this section, we start with a description over our event datasets, then present the experimental settings and performance analysis.

### 4.4.1 Datasets

We use the same datasets as in [dLMO+17] to fairly compare our results to the baseline. Tweets in these datasets are extracted from two music festivals namely VFestival and Creamfields both held in the UK (2016) [dLMO+17]. We additionally crawl social graphs of Creamfields users with 671 nodes and 723 edges, and VFestival with 649 nodes and 714 links. There exist a few users who deleted their accounts over time, hence we obtain fewer posts than the original dataset. Table 4.1 illustrates the total number of tweets and the respective percentage of positive (attended) and negative (not-attended) labels in each dataset. Our reference code and data are publicly available at https://github.com/fatemehsrz/Event_Attendance_Prediction.

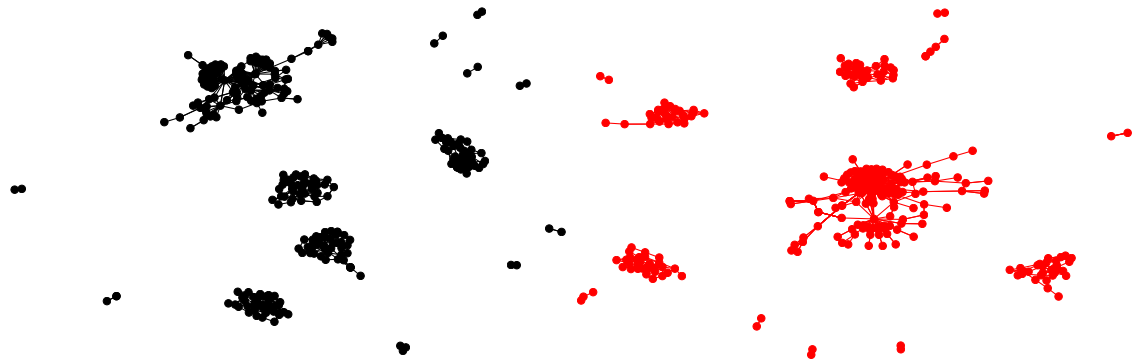
Table 4.1: Statistics of the Twitter data.

| Dataset | Timestamp | Labled | pos% | neg% |
|---|---|---|---|---|
| VFestival | before | 320 | 44 | 56 |
| | during | 329 | 47 | 53 |
| | overall | 649 | 46 | 54 |
| Creamfields | before | 330 | 46 | 54 |
| | during | 341 | 42 | 58 |
| | overall | 671 | 44 | 56 |

To examine the role of additional network features, we apply node2vec, HARP, Poincarè, and Loc to the crawled Twitter graph. Figure 4.1 depicts the structure of attendees networks crawled from Twitter for both VFestival and Creamfields. It can be noticed that social influence causes users to participate in the events as disjoint communities (social groups). Most of the attendees are members of larger groups and few joined only with a friend. This can confirm the general argument in social science that states social groups create a psychological process to strongly influence individuals behavior [VA13].

### 4.4.2 Parameters and Environment

For node2vec, we tune $p = q = 1$ which corresponds to explore local and global neighborhoods equally. For the other parameters, we use the defaults from the paper. HARP applies a hierarchical paradigm based on node2vec, hence we follow the same tuning procedure as node2vec. For Poincaré, we determine the radius



(a) VFestival (b) Creamfields

Figure 4.1: Social groups of attendees in VFestival and Creamfields.

$r > 1$ and the steepness as in $t = 0.01$ [NK17]. In the feedforward network, weights are initialized randomly, the learning rate is set to 0.01, and the model runs for 50 iterations. We run experiments on a CPU machine Intel Xeon(R) with a Core(TM) i5 processor and 32GB memory.

### 4.4.3 Results and Discussions

Our Neural Network (NN) classifier is fed with the concatenation($\oplus$) of input features:

- $\tau\oplus$Loc: Textual feature and Loc embedding
- $\tau\oplus$HARP: Textual feature and HARP embedding
- $\tau\oplus$N2V: Textual feature and node2vec embedding
- $\tau\oplus$Poincarè: Textual feature and Poincarè embedding

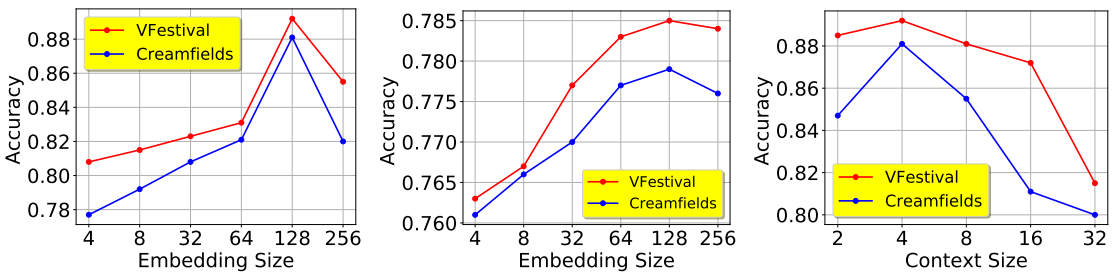
Additionally, we feed the model with single features considering only textual or network features. The model is run using 5-fold cross-validation, while preserving the proportion of positive and negative instances in each fold. The quality of the prediction is measured in terms of accuracy, precision, recall, and F1 score. Table 4.2 compares our results against the most recent work [dLMO+17] whose accuracy reached the maximum value through Gradient Boosting Decision Trees (GBDT). The maximum scores are marked in bold and the standard deviation after cross-validation is less than 0.01. The overall superior performance of our model reveals the potential role of friends' networks on event attendance. We observe that Loc embedding achieves better performance compared to the other techniques. Accuracy is already above 89% across both datasets which explains the power of restricted random walks to encode personal social networks. On the other hand, Poincarè shows the lowest performance which appears to lie at the core of its loss function. Poincarè maps

Table 4.2: Classification performance on event attendance prediction.

| Dataset | Classifier | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|---|
| VFestival | GBDT | 0.805 | 0.826 | 0.675 | 0.736 |
| | $NN_{\tau \oplus Loc}$ | **0.903** | **0.891** | **0.823** | **0.855** |
| | $NN_{\tau \oplus HARP}$ | 0.892 | 0.886 | 0.784 | 0.821 |
| | $NN_{\tau \oplus N2V}$ | 0.823 | 0.792 | 0.745 | 0.768 |
| | $NN_{\tau \oplus Poincarè}$ | 0.785 | 0.735 | 0.706 | 0.720 |
| | $NN_{\tau}$ | 0.782 | 0.756 | 0.667 | 0.708 |
| | $NN_{HARP}$ | 0.769 | 0.691 | 0.745 | 0.717 |
| Creamfields | GBDT | 0.865 | 0.838 | 0.764 | 0.793 |
| | $NN_{\tau \oplus Loc}$ | **0.894** | **0.878** | **0.815** | **0.845** |
| | $NN_{\tau \oplus HARP}$ | 0.881 | 0.857 | 0.774 | 0.814 |
| | $NN_{\tau \oplus N2V}$ | 0.803 | 0.786 | 0.758 | 0.752 |
| | $NN_{\tau \oplus Poincarè}$ | 0.779 | 0.735 | 0.689 | 0.712 |
| | $NN_{\tau}$ | 0.777 | 0.729 | 0.686 | 0.707 |
| | $NN_{HARP}$ | 0.710 | 0.732 | 0.661 | 0.695 |

many of the nodes on the border of the hyperbolic ball [GBH18] which can cause the loss of critical local properties [SB18].

We also study the dependence of performance on tuning hyperparameters of the embedding methods. Figure 4.2 depicts accuracy changes on different context sizes $w$ and dimension sizes $d$. In HARP, accuracy raises along with the increase of dimension size up to $d = 128$, but further dimensions do not provide useful information for the task. Similarly, Poincaré shows the best performance for $d = 128$ and drops afterwards. The length of the context window in node2vec and HARP characterizes the range of neighborhood locality captured by both models. As shown for HARP, a larger window causes accuracy loss since faraway neighbors appear more in co-occurred pairs. While our attendance inference model supposes direct friends' influence each other for event participation.




(a) HARP (b) Poincaré (c) HARP

Figure 4.2: Impact of hyperparameters on the classification performance.

## 4.5 Conclusion

The small-world property of social networks directly impacts the ability of the network to spread information about events. This has been observed that positive or negative posts can have an influence on friends' decision-making. Twitter is one of the platforms that large public events are widely folded and discussed. Therefore, we explored both Twitter posts and graph structure to study the event attendance problem. A key detail of our proposed approach was to exploit user's connectivity patterns in social networks applying graph embedding techniques. We used event datasets from the state-of-the-art to demonstrate the utility of network topological features to improve the prediction performance.

Despite the early success in this study, a few limitations still exist. Firstly, the current embedding models are not exactly designed to model influence diffusion in social networks. Thus, there is a need to develop embedding methods in which local influence and user similarity are both mapped into the representations. Secondly, user's activity and history are completely neglected. However, users' comments, stories, and attended events can build a set of preferences. Thirdly, the role of EBSNs in connection with other social networks, e.g. Twitter, is not discussed or investigated. EBSNs organize groups that users choose to join, hence identifying these users' posts on the other social media can potentially reveal attendance behavior and tendency. In the next part, we will explore embeddings further on the topological structures which are captured during training.

# Part II

# Exploring Vector Embeddings

# Chapter 5

# Understanding the Content of Vector Embeddings

## 5.1 Introduction

The critical role of social networks in studying people's activities and behaviors has motivated researchers for exhaustive analysis on network topological structure. For instance, how to identify the social influencers whose opinions and beliefs are quickly spread through society. Therefore, much of the research work emerged with the goal of comparing nodes in terms of connectivity patterns.

One of the frequent questions in social networks is to measure how central a node is compared to others and what position or prestige plays in the network. The concept of centrality usually capture complementary aspects of a node's position, hence a centrality measure can be more appropriated for some applications and less for others. For example, the spread of information is often affected by the influence of certain nodes with higher closeness centrality. Identifying these influential nodes helps to better understand the complex networks specifically for predicting and controlling network evolution [BBBG10, KNT10, LBKT08].

Recently, random walk based graph embedding techniques perform successfully to preserve proximity among nodes so that neighbors in a small vicinity are embedded together. However, little effort is dedicated to dive deep into why these graph embedding methods are successful and what structure is being saved in latent representations. In this chapter, we explore if some of the known and mathematically understood topological properties [New10] are projected in the embedding space. For the sake of completeness, embeddings are investigated at both subgraph and vertex level. We first investigate if vector embeddings learned something analogous with traditional properties at subgraph level, i.e. in ego networks. Ego networks generically split a social graph into subgraphs with central nodes called egos [ML14]. The ego network structure allows hidden patterns, anomalies, and features to be discovered that can be missed when the entire graph is analyzed [ACLG+16].

To achieve this, a model which relates the properties of ego networks to embedding vectors is needed, for example, a mapping from linear combination of network properties to ego similarity in embedding space. We approach this problem from an information-retrieval perspective such that topological relations between egos in the network are used to predict ego ranking. In fact, we define a learning to rank problem where pairwise similarities between egos in the network form a query and corresponding ground-truth rankings are derived from pairwise similarities in the embedding space. The goal of training is to approximate a ranker which minimizes a ranking loss function over ego networks for the queries, target egos, and associated ranks that are given in the training set. Therefore, we leverage the knowledge of network topology to produce node ranking meanwhile the model weights reveal the importance of those topological properties.

In particular, we use RankSVM [Joa02] because it accepts pairwise preferences between items as training data and is suitable for large-scale learning problems even with more missing data [K+12]. To feed the model, we construct a feature matrix by computing similarity between centrality distributions in ego networks. For ground-truth ranking, we calculate the pairwise similarity between ego vectors in the embedding space. Thereafter, RankSVM inputs the centrality similarity values and tries to learn a linear combination of weights which gives relevance scores for the candidate pairs. These relevance scores should respect the preference relations given already as the rank labels to the algorithm. Such discovery could provide a practical framework to experimentally explain which topological structures are retained in the representations.

At vertex level, we turn our attention to a regression task which defines a mapping from embeddings into centrality values. An accurate approximation confirms embeddings yield insights into the properties of the network. Therein, we use a feedforward neural network as a regressor which inputs embeddings and outputs centrality values (e.g. betweenness, closeness, and eigenvector). Since the computation of some centrality measures based on network structure is very high, this approach can ensure efficient and time-saving centrality approximation.

In summary, we present the following contributions:

- We propose to make use of learning to rank for identifying network properties preserved by graph embedding at subgraph level. According to the work by Malmi et al. [MTT+16], RankSVM has a strong capability to relate features from two spaces while revealing the importance of each feature. We thus use RankSVM to relate centrality similarity between two egos to their embedding similarity. The retrieved weights from the linear model are considered as importance for those centrality measures. A centrality measure with the highest contribution to the similarity of two egos can be considered as the highest explanatory factor regarding their relatedness in embedding space.
- We examine whether vector embeddings can directly predict the centrality values of each vertex using a feedforward network. Degree, betweenness, close-

ness, and eigenvector centrality are our main focus.

- We provide detailed experimental evidence for our claims over several graph embedding techniques and centrality measures.

The remainder of the chapter is organized as follows. Section 5.2 provides a brief overview on relevant works. Section 5.3 introduces preliminaries and required notations. Section 5.4 defines how to explore the content of embeddings and approximate centralities. Section 5.5, illustrates the experimental setup and evaluate our proposed approaches. Finally, Section 5.6, draws the conclusion and discusses the limitations.

## 5.2 Related Work

Exploring the content of vector embeddings is still quite young area in the field of data representation learning. Bonner et al. [BKB+19] search for a mapping from the embedding space to a range of topological features, if such mapping exists, those topological features are also approximated in the embedding space. In particular, they investigate the content learned by DeepWalk [PARS14], node2vec [GL16], SDNE [WCZ16], and Poincaŕe [NK17] applying supervised and unsupervised models to the embeddings. This discovery indicates that graph embedding can learn approximations of known topological features. Specifically, eigenvector centrality shows the best reconstruction by many of the methods. The authors attempt to provide a key insight into how graph embedding learns to create high quality representations.

Balogh et al. [BBDT] investigate the semantic content of word embeddings using a common sense knowledge base whose tools explore directions of vectors in sparse embeddings. Knowledge bases indeed give a computational approach for interpretability by providing explicit meaning of words along with quantifiable validity. Their methodology analyzes the paths between concepts in ConceptNet [SH12] which finally assigns labels to the dimensions of the vector embedding.

Tsvetkov et al. [TFL+15] propose an evaluation measure called QVEC to examine the quality of word embeddings. QVEC estimates a correlation between dimensions of a word vector and the semantic categories gained from SemCor [MLTB93]. Finally, Senel et al. [ŞUY+18] determine explicit assignments to word embedding dimensions with specific interpretability scores to measure semantic coherence.

## 5.3 Definitions and Preliminaries

In this study, we aim at investigating generated embeddings by DeepWalk [PARS14], node2vec [GL16], LINE [TQW+15], and Loc [RGZ17]. Our main focus is on topological features measured at subgraph level like centrality distribution in the direct

neighbors of ego $u_i$ denoted as $N(i)$. We explore the following widely used centrality measures in social networks [ZAL14].

**Degree centrality:** This centrality is defined as the number of edges incident upon a node $v$:

$$dc(v) = \deg(v). \tag{5.1}$$

**Betweenness centrality:** It measures how often a node acts like a bridge in the shortest path of two other nodes. Formally, betweenness of a node $v$ is then defined as:

$$bc(v) = \sum_{s \neq v \neq t} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}}, \tag{5.2}$$

where $\sigma_{s,t}(v)$ is the number of shortest paths between nodes $s$ and $t$ that pass through node $v$. $\sigma_{s,t}$ is the number of all shortest paths between node $s$ and $t$ in the graph.

**Closeness centrality:** This is defined as the inverse of sum over all the shortest distances from a node to the others. Formally, closeness of a node $v$ is defined as:

$$cc(v) = \frac{1}{\sum_{j=1}^{|V|} \delta(v, v_j)}, \tag{5.3}$$

where $\delta(v, v_j)$ is the length of the shortest path between $(v, v_j)$.

**Eigenvector centrality:** This measure generalizes degree centrality by incorporating the importance of the neighbors. The eignvector centrality of the vertex $v_i$ is proportional to the structural importance of their connected neighbors:

$$ec(v_i) = \frac{1}{\lambda} \sum_{j=1}^{|V|} A_{ij} ec(v_j), \tag{5.4}$$

where $A$ is the adjacency matrix of the graph $G$ and $\lambda$ is a constant value.

## 5.4 Approach

### 5.4.1 Exploring the Content of Embeddings

We aim at assessing embeddings in terms of centrality measures in the subgraph level (within ego networks). Given vectors $\phi_i$ and $\phi_j$, first, the inner-product determines the similarity between two egos in the embedding space:

$$f(\phi_i, \phi_j) = \phi_i \cdot \phi_j. \tag{5.5}$$

This embedding similarity is then used to give a rank label to each ego pair. For given egos $u_i, u_j, u_k$, the rank $y_{ij}$ is higher than $y_{ik}$ when $f(\phi_i, \phi_j) > f(\phi_i, \phi_k)$. As such, we assign a unique ranking to each ego pair based on their similarity in the embedding space. Due to the random walk strategy behind embedding creation, we assume embeddings of the egos $u_i$ and $u_j$ can be approximated by the network properties in their neighborhoods $N(i)$ and $N(j)$. Let $p_i$ and $p_j$ denote centrality distribution in the neighborhood $N(i)$ and $N(j)$ respectively, we define the centrality similarity beween $u_i$ and $u_j$ using Kullback-Leibler (KL) divergence [KL51] as:

$$s(u_i, u_j) = 1 - \mathrm{KL}(p_i \| p_j). \tag{5.6}$$

Here the pairwise similarity in the embedding space $f(\phi_i, \phi_j)$ gets related to the pairwise similarity in centrality distribution $s(u_i, u_j)$. We formulate our approach such that the similarity of two embeddings $\phi_i$ and $\phi_j$ can be approximated by a weighted sum of over the network centralities:

$$f(\phi_i, \phi_j) \sim \sum_{t=1}^{k} w_t s_t(u_i, u_j), \tag{5.7}$$

where $s_t$ is the similarity value for $t^{th}$ network centrality and $w_t$ is the weight for that measure. To estimate the optimal weights $w$, we cast the problem into a learning to rank model. We assume the availability of $m$ training examples $\{x_i, y_i\}_{i=1}^{m}$ where $m$ is the number of possible ego pairs in the network. Our feature matrix is denoted by $X := (s_1, \ldots, s_k) \in \mathbb{R}^{m \times k}$ with $k$ centrality measures, and our label matrix containing rankings represented as $Y \in \mathbb{R}^{m \times 1}$. The goal of the training is to obtain the weight matrix $W \in \mathbb{R}^{m \times k}$ such that the objective of RankSVM defined over the training data is minimized. More precisely, we train the model such that $w_i^T x_i > w_j^T x_j, \forall y_i > y_j$ by optimizing the following objective function,

$$\min_{w} \quad \frac{1}{2} w w^T + C \sum_{y_i > y_j} \max(0, 1 - w^T (x_i - x_j)), \tag{5.8}$$

where $C > 0$ is a regularization parameter and $w \in \mathbb{R}^k$ is a weight vector which reveals the importance of each centrality.

### 5.4.2 Centrality Approximation

At the aim of centrality approximation in the vertex level, we formulate a regression problem by employing neural networks fed by embeddings. Formally, we define a mapping $f$ from vector embeddings to centrality values:

$$f : \mathbb{R}^d \mapsto \mathbb{R}^+, \tag{5.9}$$

where $d$ is the size of vectors learned by one of the embedding techniques, i.e. DeepWalk, node2vec, LINE, or Loc [RGZ17]. The architecture of our model is described as follows,

- The input layer is fed with the vector embeddings of size $d$.
- The hidden layer consists of a dense layer with ReLU [Roj13] activation units.
- The output layer has a single sigmoid [Roj13] unit since the normalized centrality values fall in the range of $[0, 1]$.

## 5.5 Experiments

This section presents the experimental results for inspecting the content of vector embeddings. We evaluate the proposed methods on several ego network datasets as well as a syntactic network.

### 5.5.1 Datasets

Our first goal was to investigate assessing the content of embeddings in terms of centrality measures. Due to relying on distributional properties in ego networks, we seek datasets providing such structure with focal nodes and their neighbors. We thus take several ego-shaped datasets from major social networking sites, e.g. Facebook, Google+, and Twitter provided by SNAP [ML14]. Table 5.1 describes the details of the datasets used in our experiments. Our reference code and data are publicly available at `https://github.com/fatemehsrz/Properties_of_Embeddings`.

Table 5.1: Statistics of the social network datasets.

| | | **Facebook** | **Twitter** | **Google+** |
|---|---|---|---|---|
| nodes | $\|V\|$ | $4,039$ | $81,306$ | $107,614$ |
| edges | $\|E\|$ | $88,234$ | $1,768,149$ | $13,673,453$ |
| egos | $\|U\|$ | 10 | 973 | 132 |

We also generate a synthetic graph dataset to validate our experiments. Given the number of nodes $|V|$ and average degree $d$, we generate a synthetic network using the preferential attachment model by Barabási et al. [AB02]. Here we set $|V| = 4,000$ and $d = 20$ to simulate the Facebook graph by generating $79,600$ edges. To fairly compare the synthetic graph to the real one, we break it into 10 ego networks applying a modularity-based graph clustering algorithm [CNM04]. We further pore over the synthetic graph by comparing centrality distribution in

the composed clusters and the real ego networks from Facebook. As an instance, Figure 5.1 compares the centrality distribution of two samples: ego '686' of the Facebook and ego '5' of the synthetic graph. The goodness of fit can be measured by Kolmogorov-Smirnov (KS) statistic and p-value tests [Ste74] where low KS and high p-value indicate identical distributions. Table 5.2 provides a comparison over centrality distribution (e.g. degree distribution) in ego networks of the real and synthetic network. For each pair of egos, we can see a low KS statistic (around 0.1) and a high p-value which validate identical centrality distributions in the real and synthetic ego networks [Ste74]. We then use our synthetic network to further verify the proposed models on vector embedding investigation.

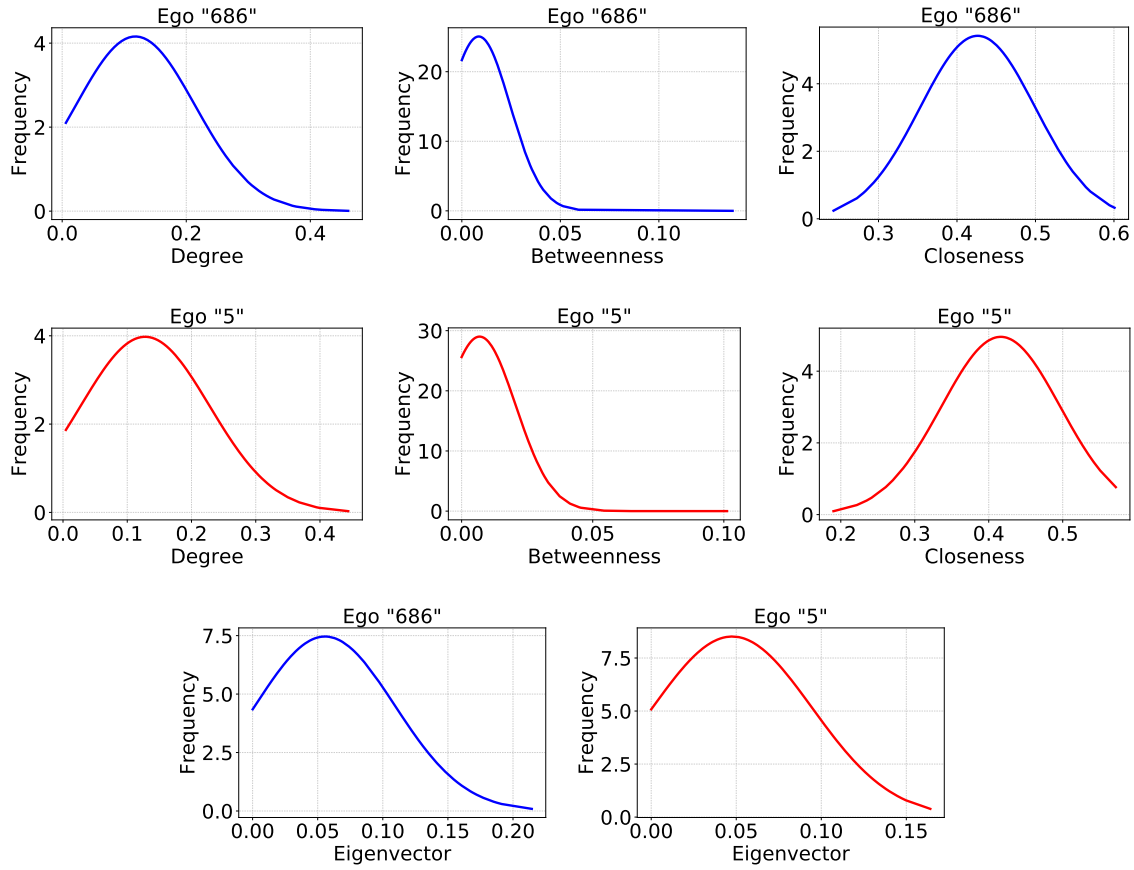



Figure 5.1: Centrality distribution of ego network '686' in Facebook and ego '5' in the synthetic graph.

### 5.5.2 Parameter Settings

To learn representations, we implemented the baseline works using the code released by the authors. For fair comparison, the embedding dimension is fixed to $d = 128$ in the baselines. The other parameters are tuned to be optimal:

- **DeepWalk:** We set the window size $w = 10$, walk length $l = 40$, and the

Table 5.2: Comparing centrality distributions in real and synthetic ego networks.

| Ego (real) | Ego (synthetic) | Centrality | KS statistic | *p*-value |
|---|---|---|---|---|
| 686 | 5 | degree | 0.12865 | 0.12432 |
| | | closeness | 0.10099 | 0.35831 |
| | | betweenness | 0.10684 | 0.29330 |
| | | eigenvector | 0.07805 | 0.68533 |
| 686 | 8 | degree | 0.12865 | 0.10042 |
| | | closeness | 0.14099 | 0.15841 |
| | | betweenness | 0.05684 | 0.59340 |
| | | eigenvector | 0.10805 | 0.39473 |
| 414 | 5 | degree | 0.11865 | 0.13402 |
| | | closeness | 0.08899 | 0.35851 |
| | | betweenness | 0.10684 | 0.23456 |
| | | eigenvector | 0.08805 | 0.69573 |
| 414 | 8 | degree | 0.13865 | 0.22532 |
| | | closeness | 0.11543 | 0.35551 |
| | | betweenness | 0.10334 | 0.25510 |
| | | eigenvector | 0.06805 | 0.55573 |

number of walks per node $\gamma = 80$.

- **node2vec:** This algorithm operates like as DeepWalk, however the hyperparameter $p$ and $q$ control the walking strategy. With $q > 1$ and $p < \min(q, 1))$, the random walk is biased towards nodes close to the start node within a small locality (BFS manner). In contrast, with $q < 1$ and a large $p > 1$, the walk is more inclined to visit nodes which are further away from the start node. Such behavior reflects a DFS-like exploration which encourages outward deep walks. We then consider two settings: node2vec$_q$ keeps the walk local with $p = 2^{-8}$, $q = 2^8$, while node2vec$_p$ walks more exploratory with $p = 2^8$, $q = 2^{-8}$.
- **LINE:** This method acts like DeepWalk with a restriction on node sampling whose pairs are limited to direct neighbors [TQW+15].
- **Loc:** Loc [RGZ17] applies Paragraph Vector to a set of walks, hence settings are similar to DeepWalk.

### 5.5.3 Exploring the Content of Embeddings

We first calculate egos' pairwise similarity in embedding space to build the ground-truth ranking. Then, the feature matrix is built by computing structural similarity between egos comparing degree, betweenness, closeness, and eigenvector distributions. To train the RankSVM model, we randomly split the input data into training (80%) and test (20%) sets. Table 5.3 reports the absolute values of estimated weights after 5-fold cross-validation. The accuracy of SVM in all experiments is around 75%,

Table 5.3: The importance weights reported for degree, closeness, betweenness, and eigenvector centrality.

| Dataset | Weight | DeepWalk | LINE | Loc | node2vec$_q$ | node2vec$_p$ |
|---|---|---|---|---|---|---|
| Facebook | $w_{dc}$ | $0.09 \pm 0.02$ | $0.15 \pm 0.05$ | $\mathbf{0.92 \pm 0.06}$ | $\mathbf{0.82 \pm 0.01}$ | $0.15 \pm 0.07$ |
| | $w_{cc}$ | $0.01 \pm 0.04$ | $0.07 \pm 0.00$ | $0.09 \pm 0.01$ | $0.04 \pm 0.00$ | $0.06 \pm 0.11$ |
| | $w_{bc}$ | $\mathbf{0.64 \pm 0.03}$ | $\mathbf{0.55 \pm 0.07}$ | $0.17 \pm 0.03$ | $0.01 \pm 0.04$ | $0.13 \pm 0.04$ |
| | $w_{ec}$ | $\mathbf{0.64 \pm 0.02}$ | $\mathbf{0.68 \pm 0.08}$ | $0.08 \pm 0.01$ | $0.07 \pm 0.00$ | $\mathbf{0.56 \pm 0.01}$ |
| Twitter | $w_{dc}$ | $0.07 \pm 0.09$ | $0.09 \pm 0.05$ | $\mathbf{0.87 \pm 0.08}$ | $\mathbf{0.53 \pm 0.01}$ | $0.04 \pm 0.01$ |
| | $w_{cc}$ | $0.15 \pm 0.00$ | $0.00 \pm 0.08$ | $0.13 \pm 0.05$ | $0.04 \pm 0.17$ | $0.02 \pm 0.04$ |
| | $w_{bc}$ | $\mathbf{0.51 \pm 0.04}$ | $\mathbf{0.69 \pm 0.00}$ | $0.19 \pm 0.07$ | $0.11 \pm 0.10$ | $0.16 \pm 0.02$ |
| | $w_{ec}$ | $\mathbf{0.71 \pm 0.05}$ | $\mathbf{0.58 \pm 0.01}$ | $0.12 \pm 0.03$ | $0.03 \pm 0.01$ | $\mathbf{0.49 \pm 0.01}$ |
| Google+ | $w_{dc}$ | $0.02 \pm 0.04$ | $0.00 \pm 0.10$ | $\mathbf{0.84 \pm 0.01}$ | $\mathbf{0.65 \pm 0.00}$ | $0.08 \pm 0.05$ |
| | $w_{cc}$ | $0.05 \pm 0.11$ | $0.04 \pm 0.09$ | $0.07 \pm 0.08$ | $0.09 \pm 0.07$ | $0.00 \pm 0.11$ |
| | $w_{bc}$ | $\mathbf{0.55 \pm 0.05}$ | $\mathbf{0.53 \pm 0.07}$ | $0.17 \pm 0.00$ | $0.14 \pm 0.00$ | $0.03 \pm 0.00$ |
| | $w_{ec}$ | $\mathbf{0.63 \pm 0.03}$ | $\mathbf{0.68 \pm 0.06}$ | $0.12 \pm 0.02$ | $0.07 \pm 0.03$ | $\mathbf{0.69 \pm 0.01}$ |
| Synthetic | $w_{dc}$ | $0.00 \pm 0.00$ | $0.15 \pm 0.03$ | $\mathbf{0.66 \pm 0.01}$ | $\mathbf{0.64 \pm 0.01}$ | $0.00 \pm 0.00$ |
| | $w_{cc}$ | $0.14 \pm 0.09$ | $0.07 \pm 0.00$ | $0.00 \pm 0.00$ | $0.09 \pm 0.15$ | $0.16 \pm 0.06$ |
| | $w_{bc}$ | $\mathbf{0.61 \pm 0.14}$ | $\mathbf{0.63 \pm 0.08}$ | $0.08 \pm 0.04$ | $0.18 \pm 0.03$ | $0.07 \pm 0.04$ |
| | $w_{ec}$ | $\mathbf{0.66 \pm 0.08}$ | $\mathbf{0.60 \pm 0.16}$ | $0.22 \pm 0.03$ | $0.17 \pm 0.07$ | $\mathbf{0.51 \pm 0.01}$ |

which tells us explaining some network properties is missing. We summarize our observations as follows:

- The weights tend to be high for degree, betweenness, and eigenvector, however show a negligible value for closeness centrality.
- LINE and DeepWalk present higher weights for eigenvector and betweenness which is in correspondence with the results found in [BKB+19]. This can be attributed to the global traverse in these methods which allows to capture the global structure required in both betweenness and eigenvector calculations. Betweenness is based on shortest path enumeration over a node, and eigenvector calculates a global ranking of all vertexes based solely on their location at a global view.
- node2vec$_q$ with large $q > 1$ and Loc [RGZ17] constrain the walks within limited areas of the network. This gives larger weights to the degree centrality as a local measure and smaller weights to other centralities.
- node2vec$_p$ with $p > 1$ shows a DFS behavior which encourages outward exploration walks to the depth of the graph. The high weights of eigenvector centrality can confirm the ability of these random walks to capture structural properties at the global scale.

### 5.5.4 Centrality Approximation

To characterize network structural properties saved within embeddings in vertex level, we approximate the centrality values directly by our regressor. To do so, we

randomly select 70% of nodes in the Facebook graph as training and the remaining as test set. Since the regressor outputs real numbers, we estimate the error between real centrality value and predicted one by Root Mean Square Error (RMSE) and Coefficient of Variation of RMSE or briefly CV(RMSE). Table 5.4 demonstrates the standard deviation and mean value of centrality besides the estimated errors.

Table 5.4: Root Mean Square Error (RMSE) and CV(RMSE) reported on centality approximation in the Facebook graph.

| **Centrality** | **Mean** | **Std** | **Model** | **RMSE** | **CV(RMSE)** |
|---|---|---|---|---|---|
| Degree | 0.012581 | 0.01516 | DeepWalk | 0.01848 | 1.46908 |
| | | | node2vec$_q$ | 0.02074 | 1.64857 |
| | | | node2vec$_p$ | 0.02226 | 1.76898 |
| | | | LINE | 0.03132 | 2.48925 |
| | | | Loc | 0.01681 | 1.33579 |
| **Closeness** | **0.25302** | **0.02319** | DeepWalk | **0.04755** | **0.18795** |
| | | | node2vec$_q$ | **0.05906** | **0.23342** |
| | | | node2vec$_p$ | **0.06260** | **0.24740** |
| | | | LINE | **0.04695** | **0.18555** |
| | | | Loc | **0.03647** | **0.14413** |
| Betweenness | 0.00105 | 0.01180 | DeepWalk | 0.01630 | 15.44306 |
| | | | node2vec$_q$ | 0.01568 | 14.84949 |
| | | | node2vec$_p$ | 0.01554 | 14.72446 |
| | | | LINE | 0.01761 | 16.68014 |
| | | | Loc | 0.02024 | 19.17428 |
| Eigenvector | 0.01036 | 0.02355 | DeepWalk | 0.02412 | 2.32737 |
| | | | node2vec$_q$ | 0.02664 | 2.56986 |
| | | | node2vec$_p$ | 0.02731 | 2.63491 |
| | | | LINE | 0.02534 | 2.44495 |
| | | | Loc | 0.02724 | 2.62788 |

RMSE measures the difference between actual and predicted values, the lower error, the better approximation. As shown in Table 5.4, closeness centrality is approximated sufficiently well since RMSE stays in an acceptable range. The mean value of closeness is around 0.25, concurrently the standard deviation shows a small value (around 0.02) which approves most of the nodes hold centrality values near to the mean. With the RMSE significantly lower than the mean value, we conclude that most of the nodes received correct centrality approximation. Additionally, we report CV(RMSE) whose values keep the lowest error on closeness approximation compared to the others. Closeness centrality measures the average distance (shortest path) from a node to others in the graph. This means the nodes with low closeness receive information flows sooner than others. A possible interpretation on the results is the existence of local manifolds in embedding methods which causes shorter distances to be preserved in representations. Although closeness centrality has not been a relevant factor for node ranking in subgraphs, a nonlinear mapping at the vertex

level can fairly estimate this centrality.

## 5.6 Conclusion

In this chapter, we explored making a step in the direction of reconstructing network properties from vector embeddings. One hypothesis was if any relation from the topological property of ego networks to ego vector similarity can be found via RankSVM. We presented an extensive set of experiments studying this problem across three datasets explaining four common network centrality measures. Another idea was to directly approximate the centrality of each vertex performing a regression task which showed success mostly for closeness centrality. This could give a key insight into the quality of representations to hold information such as the length of shortest path among nodes.

Our research aims at probing embeddings generated for social graphs consist of multiple ego networks. However, we bluntly ignored to study further topological features (e.g. number of triangles, local clustering score) not only at subgraph level but also at vertex level. Moreover, our learning to rank approach misses some standard tasks for further validation specifically on closeness centrality. It is beneficial if these tasks explain the relation between centrality in the subgraph and vertex levels. The early success on closeness centrality encourages us to study if latent representations retain the shortest distance among nodes in the next chapter.

# Chapter 6

# Shortest Path Distance Approximation

## 6.1 Introduction

With the recent growth of social networks how to efficiently compute the shortest distance between nodes still remains a challenge in the research community. Specifically to solve the fundamental problems in graph analysis such as measuring centrality of nodes [BV14, Fre77, Sab66, HC14], strength of relevance [LNK07, MC13, YBLS08], computing social distance [IK17, MBF+19], social similarity [CDF+13, HC14, LNK07], and detecting friends influence [ACKP13, DSRZ13, RBS11]. A vast variety of applications benefit from the knowledge of shortest distance within social graphs, like, selecting more trustworthy sellers in e-commerce regarding distances to sellers, and filtering out query results using shortest distance in websites like LinkedIn.

Many of the current algorithms have been developed for computing the exact distance for single source shortest path (SSSP). However, they mostly succeed on small and medium size networks and do not scale up with the network size. For today's massive networks traditional exact algorithms, such as Breadth-First Search (BFS) [Moo59], take an overlong computation time. In an unweighted graph with $|V|$ nodes and $|E|$ edges, BFS algorithm computes paths from a single source to all other vertices in no less than $O(|E| + |V|log|V|)$ time. The runtime of the approach can be improved by heuristics like A* search [HNR68] whose performance totally depends on the heuristic.

Another category of works which solve all pairs shortest path (APSP) calculates the exact distance between nodes in the graph. The Floyd-Warshall algorithm calculates all pairs distances employing dynamic programming in an intuitive way with a runtime of $O(|V|^3)$. Recently, T. M. Chan [Cha12] presented a faster implementation of Floyd-Warshall in which all the edge weights are integer and eventually runs out in $O(|E||V|)$. However, for today's million networks, computing the exact dis-

tance can take up to one minute for a pair of nodes [PBCG09]. Therefore, pairwise distance computation in such graphs can take an overlong runtime, i.e. hours for a thousand pairs. Alternatively, approximate shortest distance methods emerged as a feasible solution that scales up to larger networks.

A large category of approximation methods relies upon the use of landmarks whose position in the network helps to speed up distance calculation for other nodes. These methods first fix a set of $k$ landmarks to precompute the distance between landmarks and others, then use triangle inequality to compute the distance for any other pairs in $O(k)$. In this way, computing all pairs shortest distance in unweighted graphs can solve in $O(k|V|^2)$ [TK14]. As emblematic examples, Orion [ZSW+10] and Rigel [ZSZZ11] use landmarks to map nodes in coordinate space and then calculate the distance. However, selecting a perfect minimal set of landmarks is a NP hard problem [QCCY12] and landmark-based algorithms usually fail to accurately estimate distances between close pairs [KSW04].

In this chapter, we describe an improvement to the runtime of shortest distance approximation by exploiting the advancements of graph embedding techniques. The idea of the modification is based on the use of vector embeddings in flow distance [GGLZ17] and closeness approximation [SRG17]. This observation motivates us to first learn vector embeddings, then input representations into a regression model that yields real-valued distances. This proposed paradigm allows us to shift the complex landmark selection to applying a simple regression task to latent representations. Our contribution consists of random landmark selection, running BFS traversals from landmarks to other nodes, and an accurate analysis over distance approximation via the regression task. In detail, we first learn node representations applying node2vec [GL16] and Poincarè [NK17], then construct a feature matrix by composing node pairwise embeddings. Finally, a neural regression is devoted to input the feature matrix and return the approximated distances.

Due to sparse density in real-world social networks [JKB15], the number of edges is roughly a constant factor of the number of vertices. Therefore, the process of collecting training pairs and approximating the distance ends in linear runtime complexity. In summary, we investigate some of the recent graph embedding techniques to make the following contributions:

- We propose to utilize node2vec [GL16] and Poincarè [NK17] for shortest distance approximation in social networks.
- We demonstrate a feedforward network fed by embeddings can predict the shortest path distance effectively and efficiently, especially for the shorter paths.
- We conduct experiments on real-world network showing our approach outperforms the prominent baseline methods in terms of approximation accuracy.

The rest of the chapter is structured as follows. Section 6.2 provides an overview of the recent existing shortest distance approximation techniques. Section 6.3 elabo-

rates on our proposed method. Section 6.4 summarizes the experimental evaluation results. Finally, Section 6.5 concludes the chapter remarks.

## 6.2 Related Work

The shortest distance calculation is one of the key measures to represent network structural relations. Since traditional exact methods based on BFS usually do not scale up with network size, many studies switched to approximation techniques. The large body of literature [GBSW10,PBCG09,QXSW13,QCCY12,TACGB+11] aiming at distance approximation using landmarks due to the high speed of calculation in complex networks. The main contribution with $k$ landmarks is to satisfy the triangle inequality which reduces the approximation time to $O(k)$.

Orion [ZSW+10] maps nodes within the original graph to a fixed Euclidean coordinate space so that node distances are preserved. With the help of coordinates, a simple Euclidean distance computation can estimate the shortest distance in the constant time of $O(1)$. Orion indeed positions entire nodes in the coordinate space according to their relative distance to given $k$ landmarks. Landmarks minimize the number of shortest path calculations due to the limitation on BFS runs bounded to $k$ times. The total time to compute landmark coordinates in Orion takes $O((k^2d + kd)|V|)$, where $n$ is the number of nodes in the graph, $k$ is the total number of landmarks, and $d$ is the number of coordinate dimensions.

Rigel [ZSZZ11] expands Orion [ZSW+10] by mapping a large graph into a hyperbolic space with a low distortion error. The main idea is similar to Orion for choosing the $k$ landmarks with the highest degree in the network. Rigel further parallelizes the costly embedding process across multiple servers, allowing the system to quickly embed millions of nodes. The initial step of coordinate computation scales roughly in linear time to the graph size, i.e. $O(kd|V|)$, for a graph with $V$ nodes, $k$ landmarks, and $d$ coordinate dimensions. Given the graph coordinate system, Rigel approximates a distance query in $O(1)$.

Some variants of the $A^*$ algorithm take the advantage of landmarks to achieve better performance. Goldberg et al. [GW05] present an $A^*$ based search for landmarks algorithm which adopts triangle inequality. They select an average of 20 landmarks distributed at the corners of the graph which leads to speed up route planning. Bauer et al. [BDS+10] try to systematically push speed-up techniques into Dijkstra's algorithm such as adding goal-direction techniques to the hierarchical techniques. They also suggest a hierarchical $A^*$ based search for landmarks in dense graphs.

Gutman [Gut04] offers a technique which stores a reach value and a Euclidean coordinate for any given node in the graph. Gutman's approach can be combined with the $A^*$ algorithm to outperform Goldberg et al. [GW05] given only one landmark but performs worse given 16 landmarks. As a drawback, this approach depends on domain-specific assumptions and takes a longer preprocessing complexity with inap-

plicability to dynamic settings. Potamias et al. [PBCG09] offer to select nodes with higher centralities as landmarks that yields higher accuracy and less computational space.

Tretyakov et al. [TACGB+11] present an improvement by building shortest path trees to maintain the paths between each landmark and every other nodes. The use of the shortest path trees is specifically suitable for continuously evolving graphs. They further propose a greedy landmark selection algorithm that provides the best coverage of all shortest paths in a random sample of nodes.

Selecting an optimal set of landmarks is an NP-hard problem [PBCG09], instead, we focus on advanced findings of graph embedding to achieve higher accuracy and to speed up distance approximation.

## 6.3 Approach

### 6.3.1 Distance Approximation

Given $G = (V, E)$ as unweighted undirected graph, we apply graph embedding to generate real-valued vectors $\phi_i \in \mathbb{R}^d$ for every node $v_i \in V$. For a pair of nodes $(v_i, v_j)$ with the exact shortest path distance $\delta_{ij} \in \mathbb{R}^+$, the goal is to approximate the distance as $\hat{\delta}_{ij} \in \mathbb{R}^+$ employing a neural regression model. Formally, we define $f$ as approximation function,

$$f : \phi_i \times \phi_j \mapsto \mathbb{R}^+, \tag{6.1}$$

that maps a pair of coordinates to a real-valued distance. Our training pairs are gathered by exploring the entire graph at global scale. Following the landmark-based techniques, we first select a set of $k$ landmarks such that $k \ll n$, then run BFS traversals to the remaining $(|V| - k)$ nodes which gathers $k(|V| - k)$ pairs. For each training pair $(v_i, v_j)$, we merge vectors of the incident nodes via binary operations adopted from [GL16]. Table 6.1 illustrates the detailed definition of the binary operations namely subtraction, concatenation, average, and point-wise multiplication. Eventually, the composed vectors build a feature matrix as the input of our feedforward approximator.

Our proposed neural model consists of an input layer, a hidden layer, and an output layer. The number of units in the input layer depends on the binary operation, for instance, subtraction needs $d$ neurons while concatenation requires $2d$. Conventionally, the rectified unit (ReLU) is used as the activation function of hidden layers [Aga]. The output layer is a single unit of softplus [GBB11] within the range of $[0, \infty]$ in reference to the regression transformation. As optimizer, Adam [KB14] performs adaptive momentum with relative fast convergence for large-scale learning. We later assess the quality of the regressor by estimating Mean Absolute Error (MAE) and Mean Relative Error (MRE).

Table 6.1: Choice of binary operation.

| Operator | Symbol | Definition |
|---|---|---|
| Subtraction | $\ominus$ | $\phi_i - \phi_j$ |
| Concatenation | $\oplus$ | $(\phi_i, \phi_j)$ |
| Average | $\oslash$ | $\frac{\phi_i + \phi_j}{2}$ |
| Hadamard | $\odot$ | $\phi_i * \phi_j$ |

### 6.3.2 Computational Complexity

The proposed modifications to the shortest distance approximation can decrease the runtime complexity to linear. Learning latent representations takes precomputation time $O(|V|)$ for a graph with $V$ nodes [GF17]. Computation of BFS traversals from $k$ landmarks to the other $k(|V| - k)$ nodes takes the time of $O(k(|V| + |E|))$ when BFS on sparse graphs consumes $O(|V| + |E|)$. Our distance approximation benefits from the speed of feedforward networks which solve a distance query independent of the graph size in $O(1)$ [LSSS14]. Overall, all of the computational steps can be done at most in the time of $O(|V|) + O(k(|V| + |E|)) + k(|V| - k)O(1) + c$, where $c$ is a constant time for the test step. This computation time is still much faster than landmark-based methods of $O(k|V|^2)$ complexity.

## 6.4 Experiments

In this section, we conduct empirical evaluations on real-world datasets to demonstrate the effectiveness and efficiency of our proposed method. We first describe the details of our used datasets, then adjust the hyperparameters of the embedding techniques. Finally, we verify our approach by comparing the real and approximated distances.

### 6.4.1 Datasets

We use the following social networks representing four different orders of magnitude in terms of network size, the statistics of which is provided in Table 6.2:

- **Facebook:** This network is built on profile and social relation data from 10 ego networks gathered by SNAP [LM12]. The binary attributes are constructed based on users' profile information.
- **BlogCatalog:** This network describes relationships among bloggers from the BlogCatalog website, where attributes are generated by users as a short description of their blogs. The node label represents the topic categories provided by each author [ZL09].

- **YouTube:** This is the friendship network of the YouTube video-sharing website. Nodes are users and undirected edges indicate available subscribe relations [YL15]

- **Flickr:** This network is extracted from the Flickr online photo-sharing platform, in which links show the relation between pictures of the same location or the same gallery [MMG+07].

Table 6.2 describes number of nodes, number of edges, and average shortest distance in the network $\overline{\delta}$. To be in line with Orion [ZSW+10] and Rigel [ZSZZ11], we additionally report our results on the Flickr dataset. The implemented source code and data are publicly shared at `https://github.com/fatemehsrz/Shortest_Distance`.

Table 6.2: Statistics of the social network datasets.

| Dataset | Nodes | Edges | $\overline{\delta}$ |
|---|---|---|---|
| Facebook | 4,039 | 88,234 | 4.31 |
| BlogCatalog | 10,312 | 333,983 | 2.72 |
| YouTube | 1,134,890 | 2,987,624 | 5.55 |
| Flickr | 1,715,255 | 15,551,250 | 5.13 |

### 6.4.2 Parameters and Environment

We tune the hyperparameters to be optimal using a grid search. In node2vec [GL16], we set the number of walks $\gamma = 10$, walk length $l = 80$, and context size $w = 10$. With $p = q = 1$, the walking strategy explores the graph at local and global scales equally. For Poincaré, we adjust the parameters $r > 1$ and $t = 0.01$ [NK17] running the algorithm in 50 iterations. The dimension size in both embedding techniques is tuned to $d = 32$ and $d = 128$ which reveal the impact of more features on distance approximation. We train the regression model with the learning rate $lr = 0.01$ running for 15 iterations on a CPU machine (Intel Xeon(R) with a Core(TM) i5 processor and 32GB memory).

### 6.4.3 Approximation Quality

We first need to round real-valued predicted distances to the nearest integers as small fractions are not taken into account. Then, we measure the performance of our model by Mean Absolute Error (MAE) and Mean of Relative Error (MRE). The Relative Error (RE) is widely used in the study of shortest distance evaluations [ZSZZ11, ZSW+10, GBSW10, RS18]:

$$RE = \frac{|\hat{\delta} - \delta|}{\delta}, \tag{6.2}$$

where $\delta$ is the actual distance calculated by the BFS algorithm and $\hat{\delta}$ the approximated one. However, MRE has a natural tendency to get smaller for the larger distances which ends to be an unfair evaluation metric. We thus apply MAE as the most natural measure of average error magnitude with fast and robust computation.

### 6.4.4 Training and Test Data

To gather training pairs, we run multiple BFS traversals rooted from each landmark to the other remaining nodes. For the smaller networks such as Facebook and BlogCatalog, we fix the number of landmarks $k = 100$ while for two other larger networks $k = 5$ would be sufficient. We omit the distances with length 1 since finding the direct neighbors has linear time complexity $O(|V|)$ [KPST11]. Through the node sampling, once we find a path between two nodes, we additionally retrieve the node pairs included in that path. In this way, we keep gathering many of node pairs only with few BFS runs. Figure 6.1 depicts histograms of the distance distribution in which shorter paths ($\delta = 2$ or $\delta = 3$) occur more frequently than longer paths. Due to the small-world property of social networks, the average distance tends to be small ($\overline{\delta} < 6$) in the most of these networks [AIY13]. We therefore limit the maximum distance in our experiments to 5 and longer distances remain for future work. To gather test pairs, we follow the same strategy as the training set but with a smaller set of landmarks. The statistics of the training and test pairs are illustrated in Table 6.3.

Table 6.3: Statistics of training and test data.

| Dataset | Nodes | Training pairs | Test pairs |
|---|---|---|---|
| Facebook | 4,039 | 1,022,640 | 109,978 |
| BlogCatalog | 10,312 | 1,409,700 | 88,316 |
| YouTube | 1,134,890 | 2,452,757 | 184,413 |
| Flickr | 1,715,255 | 2,579,437 | 112,967 |

### 6.4.5 Baseline Experiment

We set up a baseline measurement in which the distance predictor is a simple linear regression. The generated vectors by node2vec or Poincaré are input to the model, then the performance is measured by MAE. Here the embedding size is fixed to $d = 128$ and the results are reported after 5-fold cross-validation. Table 6.4 demonstrates the baseline errors along with impact of different binary operations. This can be

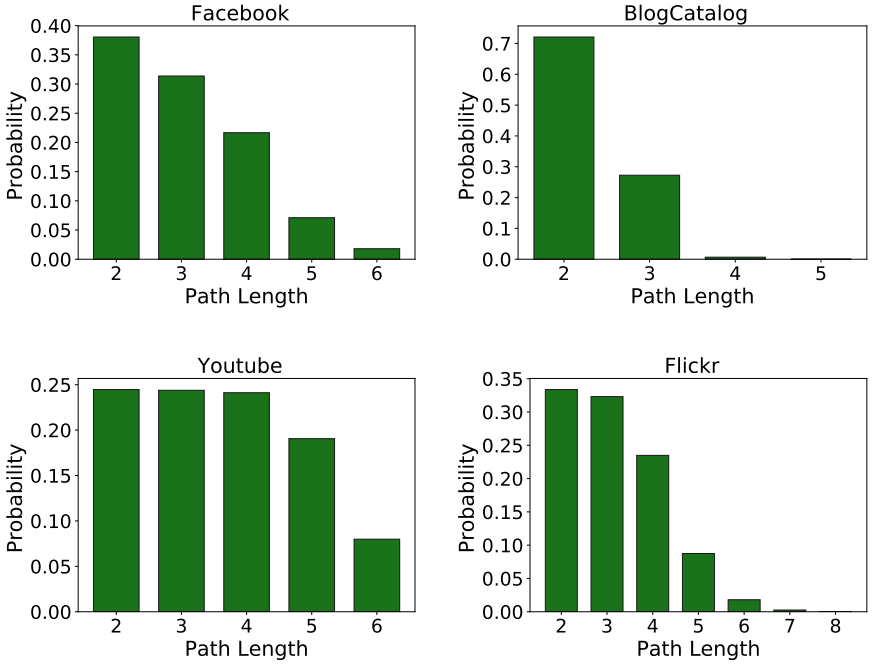



Figure 6.1: Distance distribution in the training set.

observed that node2vec indicates a smaller range of error compared to Poincaré specifically for the average operation.

Table 6.4: Baseline errors reported by Mean Absolute Error (MAE).

| **Dataset** | **Embedding** | **MAE** | | | |
|---|---|---|---|---|---|
| | | $\ominus$ | $\oplus$ | $\oslash$ | $\odot$ |
| Facebook | node2vec | 0.679 | 0.663 | **0.546** | 0.653 |
| | Poincaré | 0.801 | 0.788 | 0.656 | 0.767 |
| BlogCatalog | node2vec | 0.413 | 0.453 | **0.407** | 0.423 |
| | Poincaré | 0.427 | 0.450 | 0.436 | 0.447 |
| YouTube | node2vec | 0.708 | 0.722 | **0.695** | 0.739 |
| | Poincaré | 0.983 | 1.267 | 1.195 | 1.099 |
| Flickr | node2vec | 0.633 | 0.694 | **0.574** | 0.739 |
| | Poincaré | 0.944 | 0.891 | 0.831 | 0.822 |

### 6.4.6 Results and Discussions

We discuss our results in an attempt to demonstrate the performance of the neural regressor on distance approximation. Table 6.5 compares the results of node2vec and Poincarè where the minimum error within each scenario is marked in bold. Overall, the most success comes from node2vec embeddings with the size of 128 showing

Table 6.5: Mean Absolute Error (MAE) and Mean Relative Error (MRE) for shortest path distance approximation.

| Dataset | Embedding | | MAE | | | | MRE | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | $\ominus$ | $\oplus$ | $\oslash$ | $\odot$ | $\ominus$ | $\oplus$ | $\oslash$ | $\odot$ |
| Facebook | node2vec | 32 | 0.480 | 0.415 | 0.233 | 0.531 | 0.175 | 0.164 | 0.068 | 0.188 |
| | | 128 | 0.197 | 0.258 | **0.118** | 0.217 | 0.071 | 0.099 | **0.038** | 0.081 |
| | Poincaré | 32 | 0.592 | 0.594 | 0.552 | 0.604 | 0.214 | 0.211 | 0.218 | 0.212 |
| | | 128 | 0.437 | 0.315 | 0.372 | 0.608 | 0.169 | 0.115 | 0.142 | 0.246 |
| BlogCatalog | node2vec | 32 | 0.277 | 0.242 | 0.197 | 0.193 | 0.092 | 0.103 | 0.067 | 0.067 |
| | | 128 | 0.220 | 0.275 | 0.159 | **0.154** | 0.077 | 0.119 | 0.064 | **0.059** |
| | Poincaré | 32 | 0.338 | 0.338 | 0.343 | 0.338 | 0.108 | 0.108 | 0.112 | 0.108 |
| | | 128 | 0.331 | 0.354 | 0.277 | 0.338 | 0.115 | 0.138 | 0.097 | 0.108 |
| YouTube | node2vec | 32 | 0.676 | 0.265 | 0.455 | 0.625 | 0.230 | 0.066 | 0.163 | 0.223 |
| | | 128 | 0.344 | **0.154** | 0.174 | 0.244 | 0.101 | **0.034** | 0.040 | 0.061 |
| | Poincaré | 32 | 1.095 | 0.708 | 1.134 | 0.774 | 0.429 | 0.264 | 0.446 | 0.291 |
| | | 128 | 1.270 | 1.185 | 1.746 | 0.771 | 0.497 | 0.468 | 0.681 | 0.262 |
| Flickr | node2vec | 32 | 0.699 | 0.295 | 0.564 | 0.525 | 0.250 | 0.086 | 0.183 | 0.198 |
| | | 128 | 0.238 | **0.168** | 0.181 | 0.222 | 0.171 | **0.074** | 0.178 | 0.179 |
| | Poincaré | 32 | 0.995 | 0.808 | 1.022 | 0.874 | 0.349 | 0.284 | 0.429 | 0.278 |
| | | 128 | 0.803 | 0.662 | 0.807 | 0.764 | 0.397 | 0.432 | 0.566 | 0.364 |

MAE around 15% across all datasets. These results expose that independent of the network size, our method predicts the distance fairly well compared to the simple linear regression.

**Embedding techniques and embedding size:** The fine-tuning of dimensions in Table 6.5 shows larger dimensions lead to more accurate distance approximation. However, as the dimension increases, the time for learning embeddings and distance approximation raises subsequently. The innate hierarchical structure in social network [RB03] motivates us to examine Poincarè representations for distance approximation. Poincarè encodes graph data in a hierarchical manner to alleviate overfitting and complexity issues that often Euclidean embedding faces in these types of data. Therefore, we expect Poincarè to preserve the node distance more accurate than node2vec in the vector space. Nevertheless, Table 6.5 shows the error observed for node2vec is remarkably lower than Poincarè embedding.

The network embedding by node2vec is strongly tied to the random walk strategy with a hidden metric space which leads to preserve flow distances [GGLZ17]. This explains why node2vec yields more distance-preserved embeddings compared to Poincarè. Typically, the scale-free networks present a logarithmical scaling on average distance [ZLG+09], therefore, shortest path stays as a local feature. On the other hand, Poincarè embeds the root of nodes at the center of a hyperbolic ball and recursively embed the children in the tree by spacing them around the sphere. The radius of the sphere can be set to control the distortion of nodes within or around the sphere. The higher radius maps nodes further on the border which causes leaf

nodes to remain close to each other. However, the explicit distances from different hierarchies at global scale might not be preserved.

**Effect of binary operators:** The use of binary operations can provide insights into how various feature combination is beneficial in different scenarios. Table 6.5 shows the average operator estimates smaller errors on Facebook while concatenation works better on the YouTube dataset. The reason could lie at the minor difference within the inherent structure of the example graphs. We leave further investigation on how binary operations affect vector combination for the future work.

**Comparison to the state-of-the-art:** Rigel [ZSZZ11] and Orion [ZSW+10] are the most prominent works to approximate shortest distance via landmarks. Figure 6.2 depicts MAE on different path lengths applying our method (by node2vec), Rigel, and Orion to the Flickr dataset. We notice that our method consistently outperforms Orion and Rigel specifically on longer paths. Orion presents a graph coordinate system in which the distance is calculated by subtraction of two coordinates. However, Orion ignores the distance between non-landmarks during coordinate generation. This biased input cannot characterize the network structure well and leads to an inaccurate approximation. Rigel makes use of the hyperbolic space to design a graph coordinate system, however still random walk based property of node2vec captures semantics of the distance more effectively.

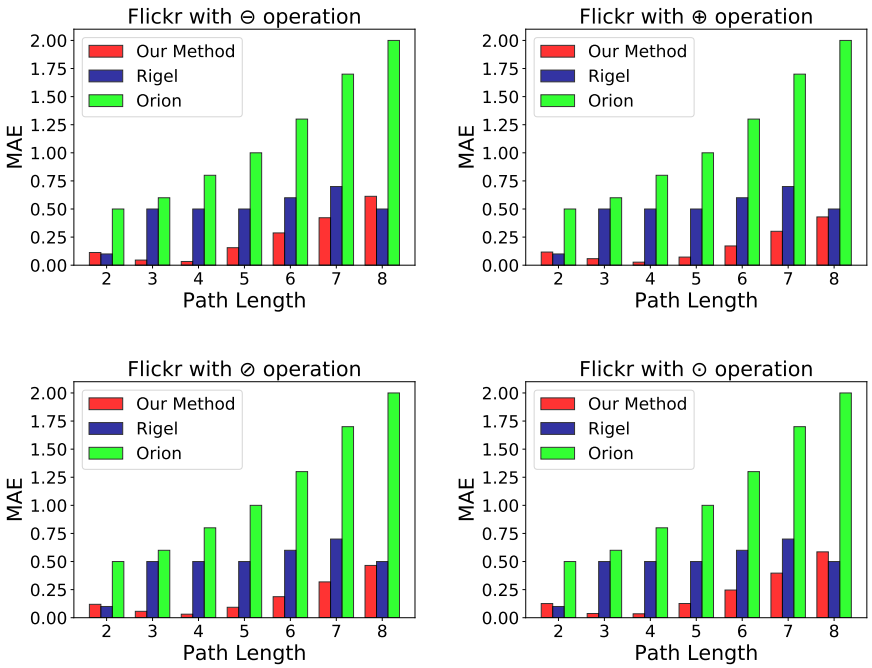



Figure 6.2: Baseline comparison reported by Mean Absolute Error (MAE).

**Error distribution over path lengths:** We explore the error range against path lengths across multiple binary operations. Figure 6.3 shows the larger errors caused

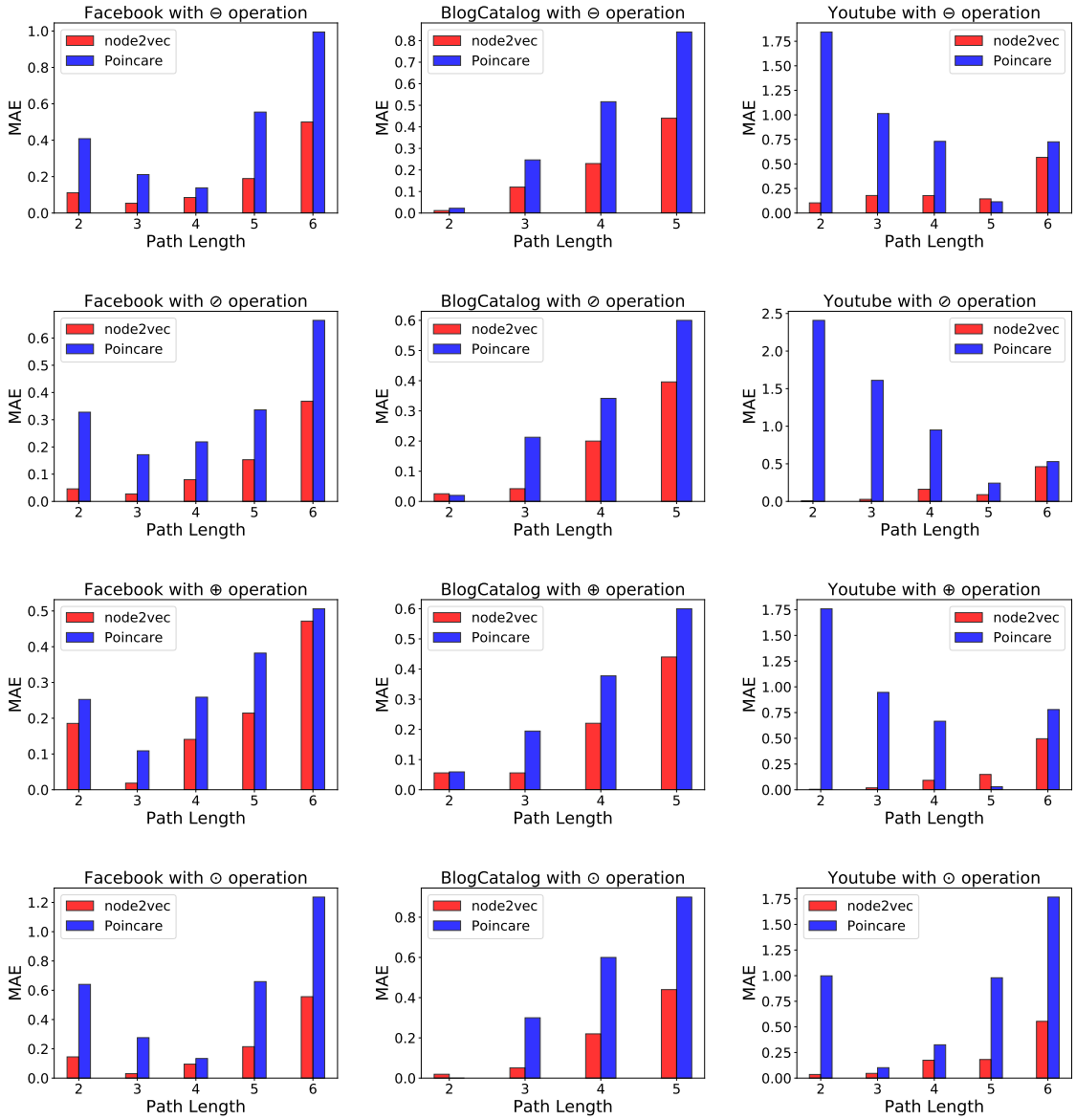



Figure 6.3: Mean Absolute Error (MAE) reported on different path lengths.

by longer paths. This can be explained by the fact that node2vec and Poincarè do not capture the property of nodes very far from a source node. The random walk strategy of node2vec usually captures the semantic of surrounding nodes which benefits short distance preserving. Poincaré differently maps leaf nodes on the boundary of the hyperbolic ball that causes local neighbors to place far from each other.

**Comparing computation time:** We record the average time for preprocessing and distance computation in our method, Orion, and Rigel. As shown in Table 6.6, our proposed method is significantly faster than Orion and Rigel in both preprocessing and test phases. Orion and Rigel incur prohibitively high cost for preprocessing caused by running too many BFS traversals from landmarks to others. However, node2vec relies on the Skip-gram model which was already proved to be fairly fast

Table 6.6: Comparing computation time across different datasets.

| Time | Facebook | BlogCatalog | YouTube | Flickr |
|---|---|---|---|---|
| node2vec Embedding | 12s | 37s | 1.5h | 1.6h |
| Orion Preprocessing | 35m | 1.4h | 8h | 12h |
| Rigel Preprocessing | 8m | 42m | 6.2h | 9.7h |
| FNN Train and Test | 5.3s | 6.5s | 2m | 2.1m |
| Orion Test | 21.9s | 17.6s | 15.3m | 9.4m |
| Rigel Test | 41s | 1.7m | 36.8m | 22.5m |

for word representation learning [MSC+13]. Our test phase conducts distance prediction using a feedforward network which is 5-18 times faster than vector-based distance calculation in Orion and Rigel.

## 6.5 Conclusion

Traditional algorithms for computing shortest path distance no longer scale up to nowadays massive social networks with millions of nodes. Inspired by landmark-based methods, we suggested gathering a set of node pairs as training samples through running multiple BFS traversals. We then made use of graph embedding techniques namely node2vec and Poincaré to calculate the node distances via neural networks. The key idea benefits from representations to reduce the time complexity of transnational methods for shortest distance approximation. Our experimental results indicated the proposed method computes the distance effectively and efficiently over example social network data.

Our analysis, however, suffers from few drawbacks: 1) the distance of the most collected pairs fall in the range of 2-4, yet longer distances are not fairly presented in the training set, 2) both node2vec and Poincaré fail in longer distance approximation which needs further investigations over parameter tuning, 3) the computation time is still high involving those many steps of prepossessing, training pair collecting, and the final test. A possible future work is to design a graph coordinate system which records the node distance for homogeneous and heterogeneous networks while running multiple random walks over the graph. The next part of the thesis will go for special types of networks with heterogeneity or additional attached information.

# Part III

# Graph Embedding for Special Networks

# Chapter 7

# Signed Heterogeneous Networks

## 7.1 Introduction

Heterogeneous networks have attracted increasing attention in the past few years due to the inherent rich types of information [SLZ+17,SH13]. More recently, a line of research for heterogeneous network embedding has been started [TQM15, CHT+15, FLL17, DCS17, HM17, SZG+18] which aims to solve practical applications such as author identification [CS17], name disambiguation [ZSG16, ZAH17], movie recommendation [SHZP18], and proximity search [LZZ+17]. However, the vast majority of these existing algorithms have been devoted to heterogeneous networks with positive or unsigned links.

Nowadays, users post and comment on social networks to express feelings and attitudes towards others, e.g. politicians, which forms sentiment links besides social links. Positive sentiment links express support or like to recipient users [LHK10a], while negative sentiment links indicate dislike or disapproval to others. For instance, a tweet posted on Twitter can express positive or negative sentiment towards Donald Trump. All of these signed sentiment links form a new network topology called sentiment network. Figure 7.1 depicts a snippet of a signed heterogeneous network with two ordinary users and three famous ones. One interesting scenario is to predict the sign of unobserved sentiment links with the aim of simplifying public opinion mining and targeted advertising.

Despite the great importance, there exists little work inferring the sign of sentiment links in heterogeneous social networks. The most recent and relevant work called SHINE [WZH+18] whose model tries to capture the sign, social, and profile information of the network via stacked autoencoders. The autoencoders of SDNE [WCZ16] are combined to learn an aggregated representation extracting the information out of three single-type networks. At the training step, sentiment and social edges are samples to jointly update parameters of three well-defined objective functions. However, SHINE confronts some drawbacks, like, 1) a relatively low performance for signed link prediction and node recommendation, 2) too many parameters to tune, 3) long training time on each epoch, and 4) exploiting profile

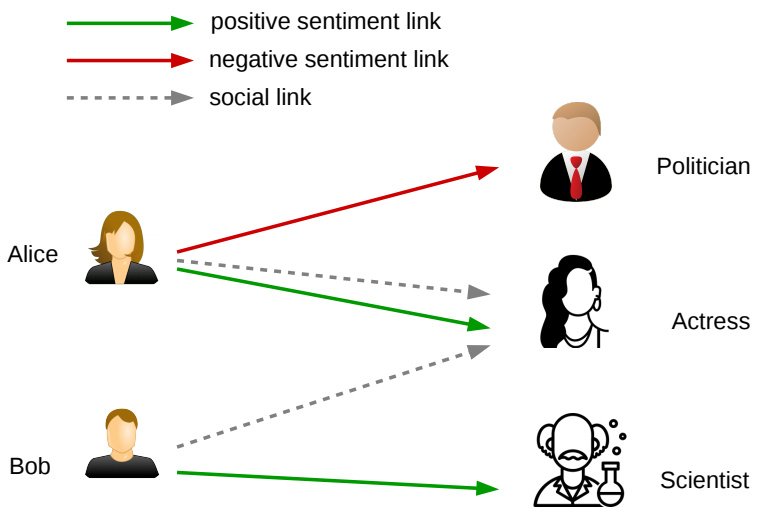



Figure 7.1: A snippet of heterogeneous networks with sentiment and social links.

information which is not always publicly available.

In this chapter, we investigate how to jointly encode semantics of sentiment and social links in linear runtime complexity. It is known that neural graph embedding methods [TQW+15, PARS14, GL16] provide fast and scalable tools for network representation learning. This finding motivates us to propose a Signed Heterogeneous network embedding method utilizing neural networks briefly referred to as SiHet. Therefore, the major contributions of this chapter are summarized as follows:

- We design a joint objective function for signed heterogeneous network embedding involving both sentiment and social links.
- We demonstrate the efficiency of the proposed SiHet which jointly learns embeddings in linear time complexity.
- We show SiHet outperforms the state-of-the-art baselines through empirical evaluations on real-world datasets.

The rest of the chapter is organized as follows. Section 7.2 reviews the related works. Section 7.3 provides preliminaries and elaborates on the proposed model. Section 7.4 discusses the empirical evaluations. Section 7.5 concludes the chapter and outlines the possible future works.

## 7.2 Related Work

Link and node heterogeneity in networks brings challenges but also opportunities to support specific applications. PTE [TQM15] and HNE [CHT+15] attempt to encode node heterogeneity considering 1-hop neighborhood relationship between heterogeneous nodes. PTE adopts deep learning models to jointly train three objective

functions which yield representations for three single-type networks. HNE aims at embedding networks with various data sources for nodes such as text, image, and video. To do so, deep neural networks and convolutional networks are aggregated to simultaneously handle vectorized data (e.g. text documents) or tensor-based multimedia objects (e.g. images, videos). This joint model coordinates two reinforcing parts by iteratively solving an optimization problem on feature learning and objective minimization.

HIN2Vec [FLL17] and metapath2vec [DCS17] later extend PTE by running metapath-based random walks over heterogeneous networks to gather heterogeneous co-occurred neighbors. The collected pairs are then used to update a negative sampling-based objective function which pushes node pairs to be close geographically and semantically. HINE [HM17] further proposes an objective function which minimizes the distance between two distributions, i.e. metapath-based proximity and embedding space proximity. Another line of work [SZG+18] learns edge representation for multiple edge types in heterogeneous networks. The idea is to maximize the typed closeness similarity for a pair of edges in the same type.

Signed networks usually indicate attitude, trust, friendship, or similarity which are the core for applications like social behavior prediction, homophily analysis, and sociological studies. SiNE [WTA+17] proposes a deep learning model to make a distinction between positive and negative edges. To update the model, every iteration requires extraction of node triples whose corresponding edges consist of a positive and a negative edge. The positive and negative distances are reflected in the embedding space with the help of an additional virtual node. SIDE [KPLK18] suggests a linearly scalable method to learn embeddings preserving both sign and direction. The key idea is to generate multiple truncated random walks and to use the collected co-occurring pairs for optimization of a likelihood objective function. The authors introduce two bias terms to model preferential attachment and direction for incoming and outgoing edges. Another study [IPR18] designs an objective function motivated by classical word2vec approaches [MSC+13] to preserve sign within higher order neighbors (more than 2-hops) in social networks.

All these methods are either focused on signed network apply social balanced theory [CH56] or networks with node heterogeneity but not both information in the same network.

## 7.3 Approach

We first introduce our notations and the mathematical definitions, then elaborate on the proposed model for representation learning.

**Signed Heterogeneous Network:** For better illustration, we split the signed heterogeneous network into two single-type networks:

**Sentiment Network:** The sentiment network is denoted as $G_s = (V, S)$,

where $V$ is a set of nodes and $S = \{(v_i, v_j)|v_i \in V, v_j \in V\}$ is a set of sentiment links. Each sentiment edge $(v_i, v_j)$ has a label $s_{ij}$ which takes the value of $+1$ or $-1$, representing a user $v_i$ has positive or negative sentiment towards another user $v_j$ respectively. $S^+$ denotes the set of positive edges and $S^-$ represents the set of negative edges.

**Social Network:** $G_r = (V, R)$ denotes a social network with $V$ nodes and $R$ edges. $R = \{(v_i, v_k)|v_i \in V, v_k \in V\}$ presents a set of social links in which an edge $(v_i, v_k)$ represents a user $v_i$ follows another user $v_k$ in the social network.

**Signed Heterogeneous Network Embedding:** Given a signed heterogeneous network, representation learning maps all nodes into the vector space in which the structural and signed relations among nodes are preserved. We design our model inspired by LINE [TQW+15] which already showed successful representation learning in linear time complexity. LINE is not directly applicable to heterogeneous networks as weights on various types of links are not comparable. Therefore, we aim at expanding the objective function of LINE involving signed links to the optimization steps. The essential idea is to map nodes with positive connections nearby in the low-dimensional space. We later describe graph embedding for sentiment and social networks separately, then explain the merged embedding through a custom objective function.

### 7.3.1 Sentiment Network Embedding

Most of the available literature [TQW+15, PARS14, GL16] model the proximity between a pair of nodes $v_i$ and $v_j$ by the following joint probability:

$$P_r(v_i, v_j) = \sigma(\phi_i^T \cdot \phi_j) = \frac{1}{1 + \exp(-\phi_i^T \cdot \phi_j)}, \tag{7.1}$$

where $\sigma$ is the sigmoid function, and $\phi_i \in \mathbb{R}^d$ is the $d$-dimensional representation for node $v_i$. The sigmoid function already showed ability of modeling the likelihood of existing or non-existing of edges between given nodes [TQW+15, PARS14, GL16].

Given a sentiment network $G_s$, each sentiment edge $(v_i, v_j)$ holds a sign $s_{ij} \in \{-1, +1\}$ must be involved in the joint probability. Since the output of sigmoid is restricted between 0 and 1, we derive that large positive numbers become nearby 1 and large negative numbers get close to 0. Therefore, we consider the largest and smallest values for the probability as follows:

$$P_s(v_i, v_j) = \begin{cases} 1 & \text{if} \quad s_{ij} = +1 \\ 0 & \text{if} \quad s_{ij} = -1. \end{cases} \tag{7.2}$$

Intuitively, the joint probability between $v_i$ and $v_j$ increases to 1 when there exists a positive link with $s_{ij} = +1$ . On the other hand, the probability gets smaller

approaching 0 when the sentiment link has a negative sign $s_{ij} = -1$. Therefore, we integrate the sign of each sentiment edge $(v_i, v_j)$ into the joint probability as follows:

$$P_s(v_i, v_j) = \sigma(s_{ij} * (\phi_i^T \cdot \phi_j)). \tag{7.3}$$

This joint probability rises up as the product term $(\phi_i^T \cdot \phi_j)$ increased by positive edges. While the joint probability drops once the negative value is multiplied to the product term. By incorporating the sign of sentiment edges into the probability, the overall objective function becomes:

$$\mathcal{O}_s = -\sum_{(v_i,v_j)\in S} \log P_s(v_i, v_j). \tag{7.4}$$

In practice, directly optimizing the objective in Equation 7.4 is computationally expensive due to the sum over all the sentiment edges. To resolve this issue, we adopt negative sampling of word embedding [MSC+13] to sample a small fraction of negative edges which enhance the influence of positive edges. Formally, our defined objective function randomly samples $K$ negative edges for each positive sentiment edge $(v_i, v_j)$:

$$\mathcal{O}_s = \sum_{(v_i,v_j)\in S^+} \Big[ -\log P_s(v_i, v_j) + \sum_{m=1}^{K} -\log P_s(v_i, v_m) \Big]. \tag{7.5}$$

The ultimate embeddings are learned via minimizing the objective function 7.5 which assigns high similarity values for positively connected nodes and low similarity values for negatively connected nodes. As a result, incident nodes from positive edges get similar embeddings while nodes connected via negative sign are placed far apart from each other.

### 7.3.2 Social Network Embedding

Given a social network $G_r$, the goal is to learn representations by preserving the network topology especially first-order proximity. Formally, we maximize the joint probability [TQW+15] for every social edge $(v_i, v_k)$ as follows:

$$\mathcal{O}_r = -\sum_{(v_i,v_k)\in R} \log P_r(v_i, v_k). \tag{7.6}$$

### 7.3.3 Signed Heterogeneous Network Embedding

We need to generate aggregated representations from two networks: a sentiment network and a social network where nodes are shared across networks. Intuitively, the model could be calibrated by minimizing the summation of loss functions of two

networks:

$$\mathcal{O}_{SiHet} = \mathcal{O}_s + \mathcal{O}_r. \tag{7.7}$$

The final joint objective function is defined via collectively sampling sentiment and social links:

$$\mathcal{O}_{SiHet} = -\log P_s(v_i, v_j) - \log P_r(v_i, v_k) - \sum_{m=1}^{K} \log P_s(v_i, v_m). \tag{7.8}$$

Where $(v_i, v_j)$ is a positive sentiment edge, $(v_i, v_k)$ is a social link, and $(v_i, v_m)$ is a negative sentiment edge. Algorithm 7.1 illustrates the steps of model optimization and edge sampling where edges are sampled with the probability proportional to the constant weight $\alpha > 1$. Typically, such oversampling significantly improves the efficiency of the optimizer in learning embeddings [TQW+15]. At the training step, the algorithm receives a batch of sentiment and social edges and updates the model parameters until convergence.

**Algorithm 7.1** Joint training

Given sentiment network $G_s$, social network $G_r$, sampling weight $\alpha$, number of negative samples $K$.
**for** $iter < |S^+|$ **do**
  sample a positive sentiment edge $(v_i, v_j)$ from $S^+$ and $K$ negative edges from $S^-$
  sample a social edge $(v_i, v_k)$ from $R$
  update $\phi_i$ taking optimization steps according to the objective function 7.8
**end for**

### 7.3.4 Time Complexity

Algorithm 7.1 samples a positive sentiment edge $(v_i, v_j)$ and a social edge $(v_i, v_k)$ for $\alpha$ times. The random edge sampling from the edgelist takes $O(1)$, and optimization with $K$ negative sampling takes $O(\alpha(K+1))$. This is visible that the number of steps used for optimization is proportional to the number of sentiment edges $|S|$. Therefore, the total runtime complexity of the SiHet is attainable in $O(\alpha K|S|)$, which is linear to the number of sentiment edges.

## 7.4 Experiments

We provide more insight into our proposed SiHet via experiments on two real-world datasets. As evaluation tasks, we follow SHINE [WZH+18] whose experiments include signed link prediction and node recommendation. For further analysis, we

investigate the sensitivity and effect of hyperparameters on the performance. All experiments have been run on an Intel(R) CPU machine with a Core(TM) i5 processor and 32GB of RAM. The implementation of SiHet and datasets are publicly available at https://github.com/fatemehsrz/SiHet.

### 7.4.1 Datasets

To benchmark our results, we examine the same datasets used by SHINE [WZH+18], namely Weibo-STC and Wiki-RfA. Weibo is a Chinese social network resembles Twitter in which users can follow and tweet. Wiki-RfA is the network of Wikipedia Requests for Adminship [WPLP14] where edges represent positive or negative votes. A collection of positive votes promotes the individuals to the role of administrator. Since Wiki-RfA does not contain social links, we adopt positively connected neighbors as social friends. The statistics of the datasets is described in Table 7.1.

Table 7.1: Statistics of the datasets.

| **Dataset** | **Nodes** | **Edges** | |
|---|---|---|---|
| | | **Social** | **Sentiment** |
| Weibo-STC | 12,814 | 71,268 | 18,950 |
| Wiki-RfA | 10,835 | 140,661 | 180,692 |

### 7.4.2 Baselines

The main relevant baseline is SHINE [WZH+18] which merges node embedding of social, sentiment, and profile networks. We additionally compare SiHet to the following baselines which integrate sign information into the final representations:

- **SIDE:** [KPLK18] This model involves both sign and direction of edges into existing models by extending a word2vec [MSC+13] based objective function.
- **SiNE:** [WTA+17] This end-to-end model optimizes an objective function which is built on the structural balanced theory assuming nodes with positive links are more similar than those with negative ones.
- **StEM:** [IR18] This method utilizes a feedforward network to map nodes in different classes via applying decision boundaries between opposing groups (e.g. friends and foes). The objective is to not only incorporate local information but also the global structure.

### 7.4.3 Settings and Environment

In our SiHet model, we set the sampling weight $\alpha = 5$, negative samples $K = 5$, and the dimension $d = 100$ same as SHINE [WZH+18]. Section 7.4.6 additionally studies the effect of parameter sensitivity on the performance. To obtain a fair comparison, we follow the hyperparameter settings suggested by the authors in the original papers. The final setting for SHINE manages the balancing parameters $\lambda_1 = 1$, $\lambda_2 = 1$, $\lambda_3 = 20$, $\lambda_3 = 0.01$, and reconstruction weight is $\alpha = 5$. For node sequence generation in SIDE, we set walk length $l = 40$, number of walk $\gamma = 10$, and window size $w = 10$. In SiNE, we fix the similarity threshold $\delta = 0.5$, and number of layers $N = 3$. We use the Adam optimizer [KB14] as it converges fast and extremely insensitive to the hyperparameters.

### 7.4.4 Signed Link Prediction

Signed link prediction is frequently used to evaluate embeddings of signed networks [KPLK18, WTA+17, WATL17, YWX17, MMLDB20]. We are given a network with several unobserved signs, the task is to predict the sign of missing edges via a binary classifier. Our test set is a percentage of unseen sentiment links with the equal number of negative and positive signs. The rest of the network will be used as the training set. We merge pairs of vectors in training and test sets through binary operations presented by Grover et al. [GL16]. Table 7.2 shows the list of binary operators. The constructed feature matrix is then fed into a logistic regression model which predicts the sign of edges between arbitrary nodes.

Table 7.2: Choice of binary operation.

| Operator | Definition |
|---|---|
| Subtraction | $\lvert\phi_i - \phi_j\rvert$ |
| Concatenation | $(\phi_i, \phi_j)$ |
| Average | $\frac{\phi_i + \phi_j}{2}$ |
| Hadamard | $\phi_i * \phi_j$ |

Following SHINE [WZH+18] experiments, we choose concatenation as our binary operator while varying the percentage of the training set from 10% to 90%. For fair comparison over the baselines, the embedding size is fixed to$d = 100$. The results obtained in signed link prediction are depicted in Figure 7.2. In summary, we make the following observations:

- SiHet consistently obtains better performance than the baseline methods in both datasets. In Weibo-STC, SiHet outperforms SIDE, SHINE, StEM, and SiNE by 6.6%, 7.1%, 14.3%, and 18.9% respectively on accuracy, and achieves 6.0%, 6.5%, 14.6%, and 17.9% gains respectively on Micro-F1.

- SIDE performs best among the baselines which caused by a likelihood formulation objective from word2vec [MSC+13]. StEM and SiNE are particularly designed for signed networks, however cannot include social edges into representation learning. SiNE only focuses on the direct neighbors of nodes rather than the global balance structure.

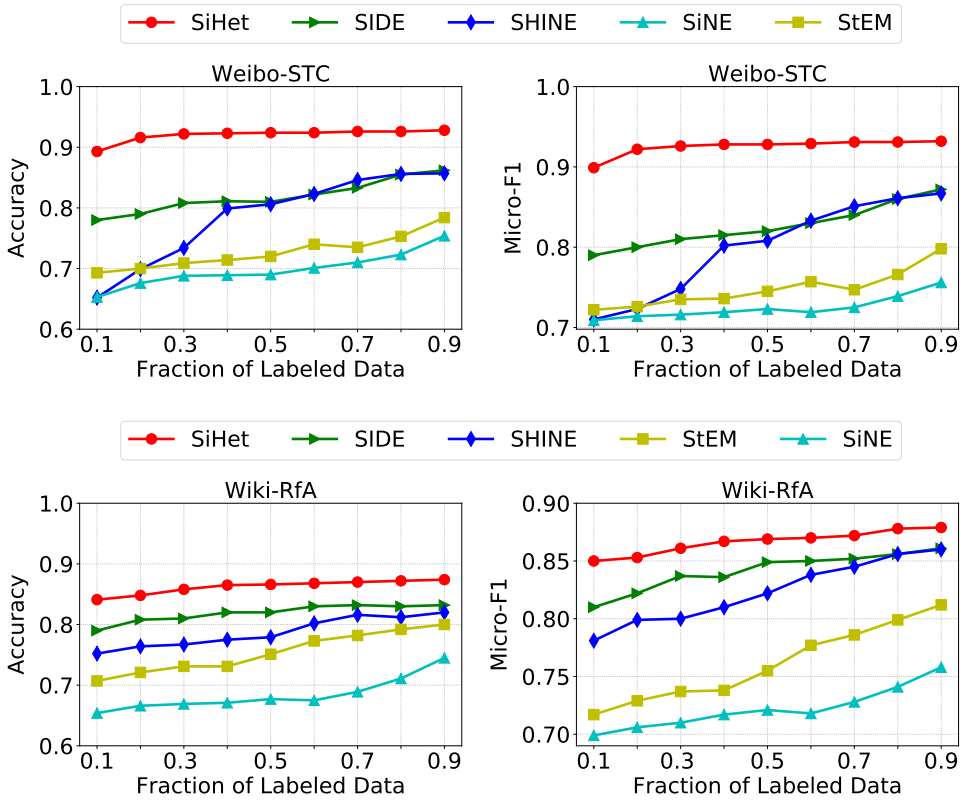



Figure 7.2: Signed link prediction reported by accuracy and Micro-F1.

### 7.4.5 Node Recommendation

The node recommendation task aims to return top-$k$ similar nodes for a given node. The criteria of the similarity is based on node proximity in the network. We conduct node recommendation task following Wang et al. [WZH+18] whose model recommends a set of users emit positive sentiment. The performance of the node recommendation task can verify the quality of the latent representations. We then consider $k$ neighbors with positive sentiment from 1-hop or 2-hop away as ground-truth for each user. Given final representations, we calculate cosine similarity from a user towards the others, then select $k$ users with the largest similarity scores as recommendations. Figure 7.3 shows the results measured by precision@k and recall@k. We then make the following observations:

- The precision curve of SiHet stands consistently above the curves of baselines

which means nodes with higher similarities in the embedding space are mostly the nodes appeared in the same neighborhood containing positive sentiment.

- The overall performance on Weibo-STC are better than Wiki-RfA, which is in accordance with the results in signed link prediction. The reason lies at Weibo-STC structure which provides well separated sentiment and social links which significantly improves the quality of learned representations.

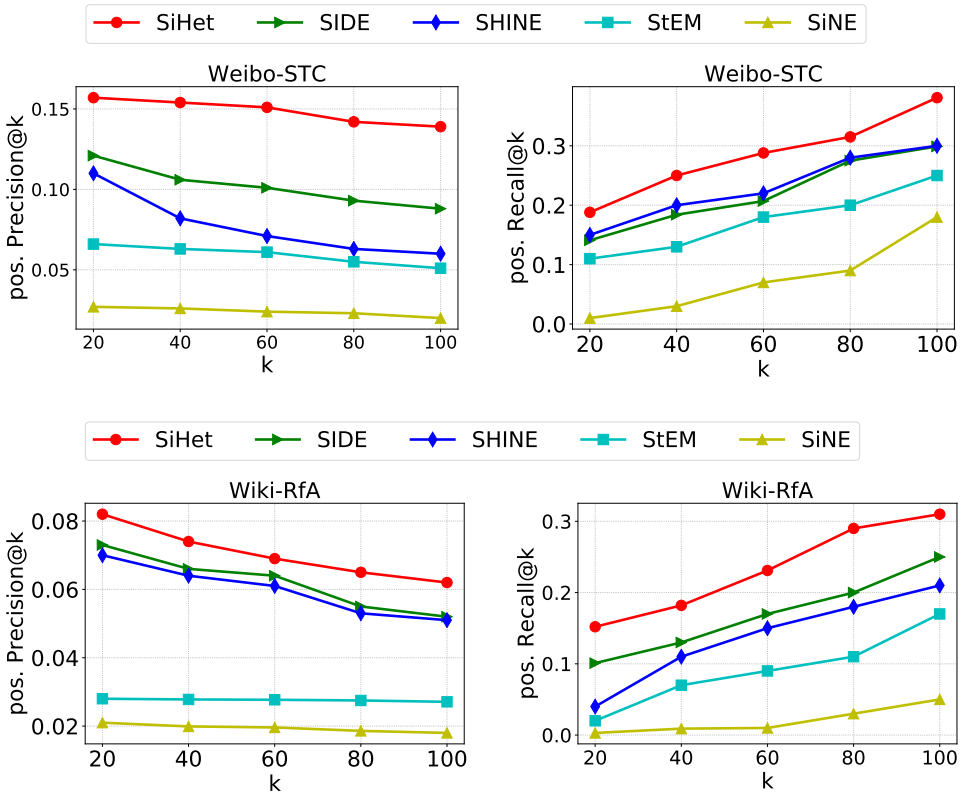



Figure 7.3: Node recommendation reported by positive precision@k and recall@k.

### 7.4.6 Parameter Sensitivity

We additionally examine the impact of embedding dimension $d$, sampling weight $\alpha$, and binary operations on the performance of the link prediction. Throughout the experiments, we randomly select 80% sentiment links as the training set and the remaining 20% as the test set. We also investigate the efficiency of SiHet on the Weibo network increasing number of nodes.

**Impact of $d$ and $\alpha$:** We tune the dimension size whilst sample edges with different $\alpha$ from Weibo. As shown in Figure 7.4, sampling edges more than one time is a quite effective approach to improve the quality of learned embeddings. Once $\alpha > 1$, accuracy rises around 92% which validates the impact of oversampling to learn more

meaningful representations. Another observation is the stability of the performance for $\alpha > 1$ at any dimension size. This explains SiHet can strongly encode the heterogeneous structure even in few dimensions.

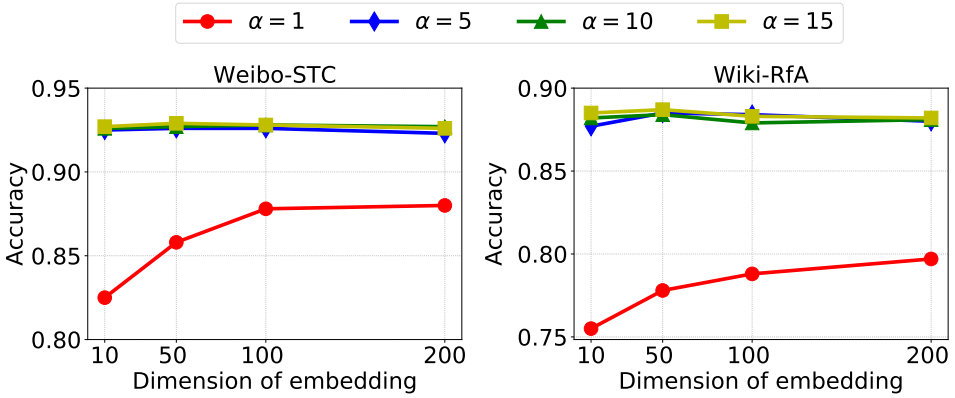



Figure 7.4: Correlation of dimension size $d$ and sampling weight $\alpha$.

**Impact of binary operations:** We examine the effect of binary operations with the fixed sampling weight $\alpha = 5$. In Figure 7.5, we can see concatenation and average prove to be the best performers across two datasets. In specific, concatenation provides a lot of information with $2d$ size. Average also calculates the centroid of two vectors which contains information from both vectors equally.

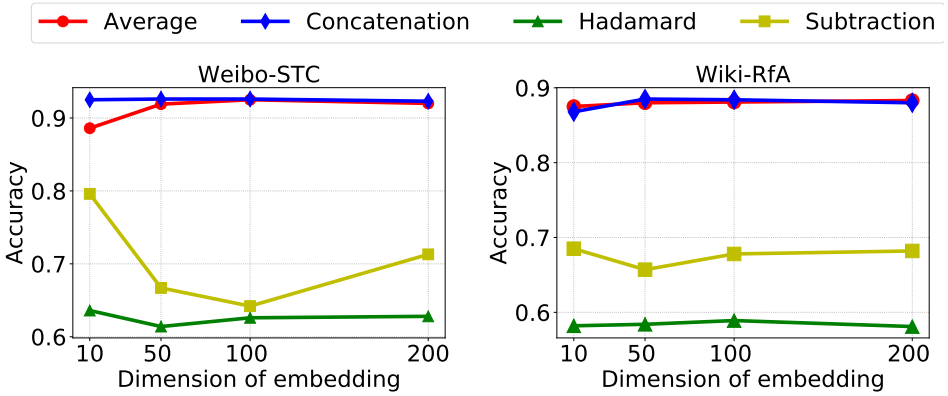



Figure 7.5: Affect of binary operators on accuracy in signed link prediction.

### 7.4.7 Efficiency Evaluation

To expose the computational efficiency of SiHet, we run the baseline methods on Weibo-STC while increasing the number of nodes. As depicted in Figure 7.6, SiHet runs up faster than the baselines and demonstrates a linear scalability with the smallest slope. The superior speed comes from the concise likelihood formulation and a few number of parameters updating at each iteration. However, SIDE spends

more time on random walks, and SHINE updates many parameters of the joint model.

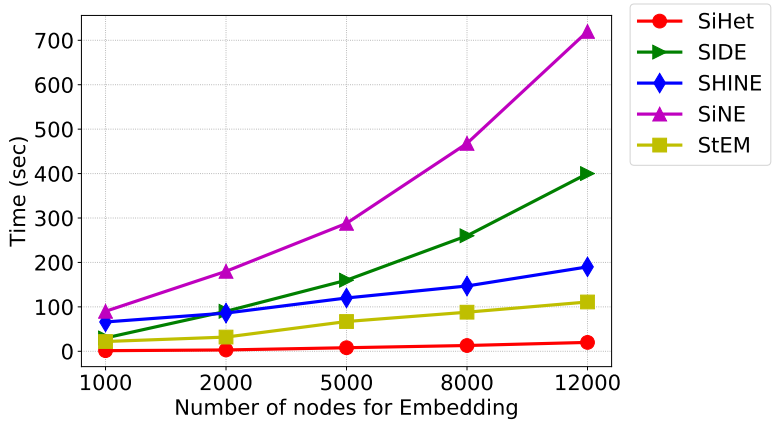



Figure 7.6: Scalability of SiHet on Weibo dataset.

## 7.5 Conclusion

This study focused on representation learning for signed heterogeneous networks in social media. The prior existing work in this direction [WZH+18] used private side information of users and converges with high runtime cost. Therefore, we suggested an extension to the LINE model to learn embeddings considering only sentiment and social links. Via experiments on two real-world datasets, we demonstrated the superior performance of SiHet in link prediction and node recommendation compared to the baselines. We also showed the model scalability in representation learning which yields linear runtime complexity.

However, this work is limited in few directions: 1) there is not a specific formulation for the non-existing edges in the objective function whose terms just push away negatively connected nodes, 2) further evaluation tasks and more datasets are needed to validate the method, 3) a conflict case when two social friends express negative feelings towards each other is not investigated, 4) the direction of negative or positive attitudes towards friends is not formulated.

There are several possibilities to expand this work. First, we can design an objective function to consider meta-data on edges such as text or image. Second, we can investigate how these embeddings benefit other network mining tasks such as node classification and multi-task learning [LHR+15]. Finally, we can extend the work for node and link heterogeneity within a directed network. In the next chapter, we aim to design a model which encodes the additional information attached to the network.

# Chapter 8

# Attributed Networks with Labels

## 8.1 Introduction

Network embedding techniques are typically based on structural proximity, mapping nodes in the local neighborhood nearby in the vector space. In real-world networks, however, nodes are often associated with a set of side information such as contents (e.g. profile attributes, textual features) or labels (e.g. community, affiliation group). These labels are strongly influenced by and inherently correlated to both network structure and attribute information [HLH17b]. Therefore, modeling and incorporating attributes and labels simultaneously into network embedding results more effective vector representations.

Recently, various efforts have been made to map networks with attributes (attributed networks) into the low-dimensional latent representations [GH18, HLH17a, MLBZ19, YPZ+18, ZYZZ19]. All of these methods include network structure and attributes but ignore abundant labels which potentially benefit network embedding and subsequent analytic tasks. Among the baselines, LANE [HLH17b] is the most relevant work which smoothly incorporates labels and attributes into network embedding while preserving their correlation. First, network affinity matrices are constructed from the attribute matrix, adjacency matrix, and the label matrix, then decomposition techniques are applied on affinity matrices to gain final embeddings. This means LANE conducts three embedding tasks to jointly encode the network information into a shared representation through optimizing three objective functions. Despite the strong model, LANE encounters several drawbacks: 1) a high computational cost for matrix decomposition, 2) manually tuning weights for multiple objectives, and 3) relatively low performance in node classification and link prediction.

To overcome the existing drawbacks, we need to design a multi-task learning framework which adapts weights of objectives during training. There has been a vast amount of research showed multi-task learning improves the overall performance of each task relative to learn them separately [Car97, CTLY09, CZY11]. Specifically, stacked autoencoders as a unified framework demonstrated promising

performance on conducting multiple classifications and regression tasks simultaneously [MM+17, BKWG17, CDR16, ZLJ+15]. The architecture of the joint autoencoders facilitates the integration of multiple information sources towards an effective representation learning [MM+17, BKWG17]. In this study, we propose a Joint Autoencoders model for Multi-task network Embedding, abbreviated as JAME, which learns shared representation through optimizing multiple objectives. Overall, our end-to-end model composed of three autoencoders: a network structure autoencoder, an attribute autoencoder, and a label autoencoder.

At the training phase, autoencoders reconstruct the input data by minimizing the respective reconstruction losses. The overall performance of the multi-task learning highly depends on the optimal choice of weighting between each task's loss function [CBLR17, BH17, KGC18]. However, manually tunning loss weights is tedious specifically for a large number of learning tasks. We thus define an additional loss weighting layer whose weights are based on task difficulty. This layer dynamically assigns weights to each task where difficulty is the loss ratio between the initial loss and the batch loss. We want higher weights for poorly trained tasks (ratio close to 1) to contribute more to overall loss and gradient. In summary, the main contributions of this paper are as follows:

- We propose JAME, an end-to-end model which encodes shared representations for the networks through multi-task learning.
- We define a loss weighting layer to automatically adapt loss weights based on task difficulty.
- We empirically evaluate our JAME on real-world datasets showing its superior performance over the baseline methods.

The rest of the chapter is organized as follows. Section 8.2 outlines the recent related works. Section 8.4 presents notations and the proposed model. Section 8.5 describes empirical evaluation. Finally, Section 8.6, concludes the chapter and discusses future works.

## 8.2 Related Work

Network embedding via deep neural networks has become an efficient tool to deal with complex networks. Recently, there have been numerous embedding methods which focused on preserving the topological structure of plain networks [GF18] which yield to solve various real-world applications [JS18, Zho19, RG19, NMV17, SRG17, RGZ17, RSG18]. However, today's massive networks are often associated with a rich set of attributes and abundant label information which are typically encoded within joint embedding frameworks. DANE [GH18] proposes joint autoencoders which input graph adjacency and attribute matrices to return shared representation

via optimizing four objectives, i.e. first-order proximity loss, higher-order proximity loss, semantic proximity loss, and complementary loss. The final embedding preserves first-order and higher-order proximity of the network which results superior performance in typical graph mining tasks.

Another work by [MLBZ19] named CAN utilizes two variational autoencoders containing an inference model and a generative one. The inference model encodes attributes into Gaussian distributions while the generative model reconstructs the original edges and attributes. A reparameterization method is applied to transform embeddings from Gaussian random variables to deterministic variables. This process helps to measure affinities between nodes and attributes. CAN finally learns separate embeddings for attributes and nodes in the same semantic space, mapping nodes nearby their relevant attributes.

Some recent works like BANE [YPZ+18] and LANE [HLH17b] jointly apply matrix factorization to attributed networks to earn shared representations. BANE first applies the Weisfeiler-Lehman proximity to aggregate adjacency and attribute matrices into a unified proximity matrix. Then, the proximity matrix is factorized by the Weisfeiler-Lehman decomposition approach to obtain the final binary representations. LANE first integrates attributes and edge connectivity into a unified proximity matrix via spectral transformations. Then, graph Laplacian [CG97] is used to smoothly embed both label matrix and proximity matrix into final representations. Although LANE observes abundant labels during training, the performance is relatively low with an expensive runtime complexity.

Multi-task learning conducts multiple relevant tasks simultaneously such that useful information can be shared. The joint model usually optimizes multiple objectives to improve efficiency and accuracy of all tasks. The main challenge is how to automatically obtain optimal weights for these objectives during model training. Kendall et al. [KGC18] propose a multi-task loss function based on maximizing the Gaussian likelihood with homoscedastic uncertainty which optimizes the combination of objectives. The model conducts multi-task weighting and outperforms separate models trained individually on each task. Similarly, Bentaieb et al. [BH17] define a multi-loss objective function integrating uncertainty in each loss term which is trainable during model optimization. The proposed convolutional autoencoders classify medical images meanwhile learn how to optimally combine multiple objectives. Another method is that of GradNorm [CBLR17] which adaptively balances the loss weights based on the norm of gradients. The internal gradients are used to manipulate gradient norms over time.

Our JAME model for multi-task learning can be viewed as a generalization of GradNorm with two important differences: 1) we measure task difficulty in each iteration to contribute difficult tasks more to the overall loss and gradient, 2) instead of assigning task-weights by gradient norm, we add extra-capacity to the network in the form of a loss weighting layer. This layer is placed after the decoders and learns optimal weights in reference to task difficulty. To our best knowledge, we are the first employing multi-task loss weighting into graph embedding to improve some

graph analysis tasks.

## 8.3 Problem Definition and Preliminaries

Let $G = (V, E)$ be an attributed network with abundant labels, where $V$ is a set of nodes and $E$ is the set of edges. The network is represented by an adjacency matrix $A \in \{0,1\}^{n \times n}$ with $n = |V|$, an attribute matrix $X \in \{0,1\}^{n \times k}$ with $k$-dimensional attribute vector, and a label matrix $Y \in \{0,1\}^{n \times m}$ with $m$ categories.

Each row $x_i$ in matrix $X$ describes the attributes associated with node $v_i$, and each element $Y_{ij} = 1$ in matrix $Y$ indicates node $v_i$ belongs to category $j$. The final shared embedding is denoted as $U \in \mathbb{R}^{n \times d}$ where $d \ll n$ is the dimension size.

Based on the terminologies discussed above, we formally define the problem of multi-task network embedding as follows. Given an attributed network $G$, we aim to learn a mapping $f : \{A, X, Y\} \mapsto U$ in an unsupervised manner by preserving the network structure $A$, attribute $X$, and label information $Y$ via operating the following learning tasks:

- **Network structure embedding** maps network topological structure into latent representations:
$$f : \{A\} \mapsto U \tag{8.1}$$
- **Attribute embedding** incorporates attributes of the nodes into the vector space:
$$f : \{X\} \mapsto U \tag{8.2}$$
- **Label embedding** integrates node labels into the vector space:
$$f : \{Y\} \mapsto U \tag{8.3}$$

## 8.4 Approach

### 8.4.1 Framework

Figure 8.1 depicts our JAME which handles joint autoencoders aim at encoding the input data while assigning optimal weights to parallel tasks. JAME composed of three autoencoders: network structure autoencoder, attribute autoencoder, and label autoencoder. These autoencoders are jointly trained by sharing the learned structure $U$ across three embedding tasks. JAME receives the adjacency matrix $A$, the attribute $X$, and the binary label $Y$ as input and reconstructs the outputs $\hat{A}$, $\hat{X}$, and $\hat{Y}$. Additionally, JAME offers the loss weighting layer $L$ capable of learning optimal weights for all objectives during training. We provide more detail on our proposed JAME model in the following sections.

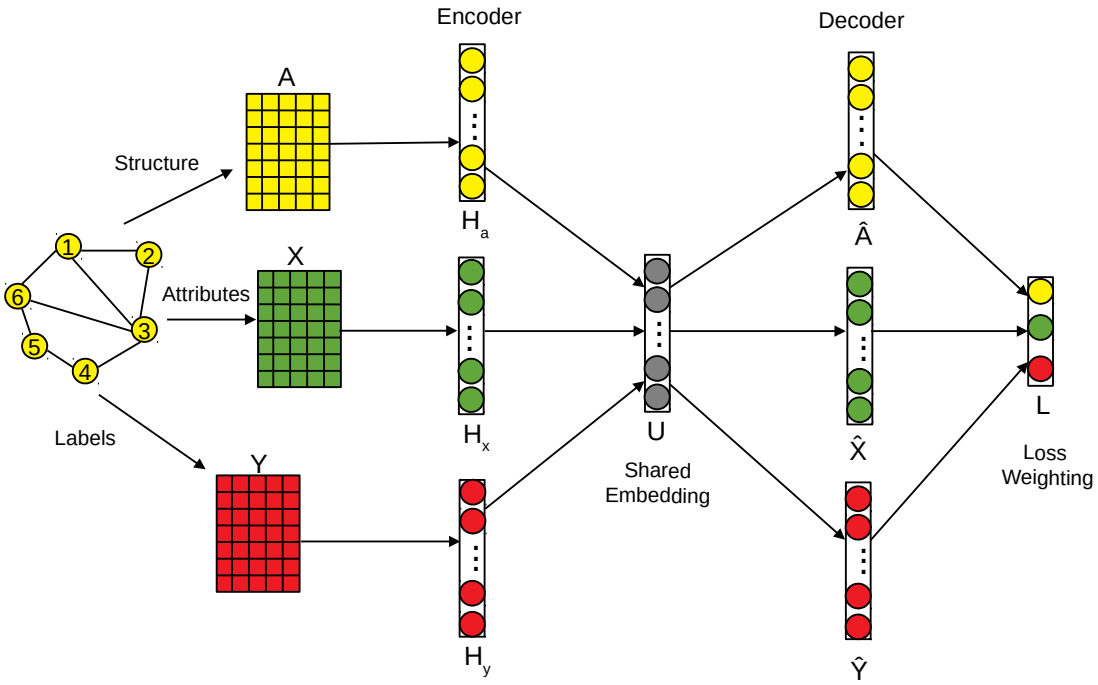



Figure 8.1: The framework of our proposed JAME model. The model inputs the adjacency matrix $A$, attribute matrix $X$, and the label matrix $Y$ to reconstruct $\hat{A}$, $\hat{X}$, and $\hat{Y}$ as output. The shared embedding layer $U$ aggregates information from the structure, attribute and labels while loss weighting layer $L$ learns optimal weights for loss values based on task difficulty.

### 8.4.2 Network Structure Embedding

We build our first autoencoder by employing nonlinear activation functions to encapsulate the nonlinear structure of the input network. As shown in Figure 8.1, the encoder gets latent representations for each node while the decoder computes the pairwise distance between node representations generated by the encoder. More formally, the structure encoder is defined as:

$$H_a = \text{ReLU}(AW_1^{(1)} + b_1^{(1)}), \tag{8.4}$$

here $H_a \in \mathbb{R}^{n \times s}$ is the hidden representation by the structure encoder with dimension size $s$. $W_r^{(l)}$ is the trainable weight matrix of the $r^{th}$ autoencoder and $b_r^{(l)}$ is the bias there which both placed in the $l^{th}$ layer. Sigmoid $\sigma(.)$ increases the nonlinear property of neural networks and ReLU is a widely used element-wise activation function. In Figure 8.1, the bottleneck features in $U$ constitute an informative compact representation of the data. Our shared layer is the summation of representations from the structure encoder $H_a$, attribute encoder $H_x$, and label encoder $H_y$:

$$U = \sigma(H_a + H_x + H_y)W^{(2)} + b^{(2)}), \tag{8.5}$$

where $W^{(2)}$ is the shared weight within stacked autoencoders. The structure

decoder reconstructs the network adjacency matrix:

$$\hat{A} = \sigma(UW_1^{(3)} + b_1^{(3)}), \tag{8.6}$$

where $\hat{A}$ is the reconstruction of the adjacency matrix, and $U \in \mathbb{R}^{n \times d}$ is the shared representation with dimension size $d$.

According to the definitions in [Tra18], we update the model parameters by minimizing a masked cross-entropy loss formulation:

$$\mathcal{L}_{ce} = -a_i \log(\sigma(\hat{a}_i)) - (1 - a_i) \log(1 - \sigma(\hat{a}_i)), \tag{8.7}$$

$$\mathcal{L}_a = \frac{\sum_i m_i \odot \mathcal{L}_{ce}}{\sum_i m_i}. \tag{8.8}$$

Where $a_i \in \mathbb{R}^n$ is an adjacency vector of $A$, and $\hat{a}_i \in \mathbb{R}^n$ is the reconstruction output vector of $\hat{A}$. $\odot$ refers to the element-wise Hadamard product, and $m_i$ is a boolean mask, i.e. $m_i = 1$ if $a_i = 1$, else $m_i = 0$.

### 8.4.3 Attribute Embedding

We simply employ another standard autoencoder to capture vertex-wise attribute proximity. The encoder creates latent representations while the decoder rebuilds the attributes. The hidden layer of the attribute encoder is defined as:

$$H_x = \text{ReLU}(XW_2^{(1)} + b_2^{(1)}), \tag{8.9}$$

here $X$ is the attribute matrix, and $H_x \in \mathbb{R}^{n \times s}$ is the hidden representation by the attribute encoder with dimension size $s$. The attribute decoder reconstructs the attributes associated with nodes:

$$\hat{X} = \sigma(UW_2^{(3)} + b_2^{(3)}), \tag{8.10}$$

where $\hat{X}$ is the rebuild attribute matrix, and $U \in \mathbb{R}^{n \times d}$ is the final shared representation with dimension size $d$. We optimize the binary cross-entropy to update parameters owing to the binary representation of the input matrix:

$$\mathcal{L}_x = -x_i \log(\sigma(\hat{x}_i)) - (1 - x_i) \log(1 - \sigma(\hat{x}_i)), \tag{8.11}$$

where $x_i \in \mathbb{R}^n$ is an attribute vector of $X$, and $\hat{x}_i \in \mathbb{R}^n$ is a reconstruction output vector of $\hat{X}$.

### 8.4.4 Label Embedding

Labels play an essential role in determining the status and mutual impact of nodes with strong intrinsic correlations to network structure. We model network label

information via a standard autoencoder. The label encoder transformation is defined as:

$$H_y = \text{ReLU}(YW_3^{(1)} + b_3^{(1)}), \tag{8.12}$$

where $H_y \in \mathbb{R}^{n \times s}$ is the hidden representation from label encoder with dimension size $s$. The label decoder rebuild the class labels of nodes:

$$\hat{Y} = \sigma(UW_3^{(3)} + b_3^{(3)}), \tag{8.13}$$

where $\hat{Y}$ is the reconstruction of the label matrix, and $U \in \mathbb{R}^{n \times d}$ is the shared representation with dimension size $d$. Intuitively, we minimize the categorical cross entropy loss to update model parameters in a multi-class setting:

$$\mathcal{L}_y = \sum_{i=1}^{m} -y_i \log(\sigma(\hat{y}_i)), \tag{8.14}$$

where $y_i \in \mathbb{R}^n$ is a label vector of $Y$, $\hat{y}_i \in \mathbb{R}^n$ is a reconstruction vector in $\hat{Y}$, and $m$ is the number of class labels.

### 8.4.5 Adaptive Loss Weighting

In the loss weighting layer $L$, we calculate the final loss $\mathcal{L}_f$ which is weighted combination of multiple losses:

$$\mathcal{L}_f = \text{ReLU}(w_a\mathcal{L}_a + w_x\mathcal{L}_x + w_y\mathcal{L}_y), \tag{8.15}$$

where $w_a$, $w_x$, and $w_y$ are the weights for network structure, attribute, and label embedding tasks respectively. Our weighting approach is based on task difficulty which measures the rate of loss changes for each task during training. The lower ratio between the current loss and the initial loss means the more difficult task. In this way, a poorly trained task needs higher weights to contribute more to the final loss and gradient. To control the softness of task weighting, we use the softmax operator inspired by distillation [HVD15]. Algorithm 8.1 receives a batch of nodes and calculates the task difficulty to weight loss terms. The final loss is calculated as a combination of loss values to obtain gradients and update model weights. The model iterates on several training steps until convergence and results embeddings.

### 8.4.6 Training Complexity

The time complexity of JAME is proportional to $O(cdI|V|)$, where $c$ is the average number of non-zero entries in each row of adjacency matrix (average degree of the network). $c$ is a constant in scale-free networks [BB03]. $d$ stands for the maximum dimension of the hidden layers, and $I$ is the number of iterations, thus $cdI$ stays a

**Algorithm 8.1** JAME Training with Adaptive Loss Weighting.

Given a set of tasks $\{1, 2, \cdots, T\}$
Initialize task weights $w_t = 1\ \forall t$
Initialize model weights $W^{(l)}\ \forall l$
**for** each iteration $i$ **do**
  **for** each input batch $B$ **do**
    Compute each task loss $\mathcal{L}_{(B)} \in \mathbb{R}^T$
    Get the first batch loss $\mathcal{L}_{(0,i)} \in \mathbb{R}^T$
    Compute each task difficulty $\gamma_t = \frac{\mathcal{L}_{(B)}}{\mathcal{L}_{(0,i)}} \in \mathbb{R}^T$
    **for** each task $t$ **do**
      Update the task weight $w_t = \frac{\exp(\gamma_t)}{\sum_k \exp(\gamma_k)}$
    **end for**
    Compute final loss $\mathcal{L}_f = \sum_t w_t \mathcal{L}_{(B,t)}$
    Update $W^{(l)}$ with respect to $\mathcal{L}_f$
  **end for**
**end for**

constant and independent of $|V|$. Therefore, the training complexity is bounded to $O(|V|)$ which signifies the total time is proportional to the number of nodes in the network.

## 8.5 Experiments

This section first introduces datasets, baseline methods and experimental design, then track the impact of loss weights on the performance over several iterations. Lastly, we compare the efficiency of JAME to the baseline methods in terms of computational time. We conduct experiments on real-world datasets to evaluate the effectiveness of JAME on node classification, link prediction, and attribute inference. These analytical tasks can give insights to understand whether shared embeddings retain certain types of information. We repeat 5-fold cross validation and report the mean result. Our source code and data are available at `https://github.com/fatemehsrz/JAME`.

### 8.5.1 Datasets

We test our proposed model on three widely used benchmark datasets: Cora, Citeseer, and PubMed [YPZ+18, MLBZ19, GH18, ZYZZ19, CSG+19, GPH19, BKBM18]. These datasets are citation networks [SNB+08] in which nodes present publications, edges indicate citations, and labels stand for research topics. Each node is associated with a set of attributes whose presentation is formed by Bag-of-Words. The statistics of the datasets is summarized in Table 8.1.

Table 8.1: Statistics of the datasets.

| Dataset | Nodes | Edges | Attributes | Labels |
|---|---|---|---|---|
| Cora | 2,708 | 5,429 | 1,433 | 7 |
| Citeseer | 3,312 | 4,660 | 3,703 | 6 |
| PubMed | 19,717 | 44,338 | 500 | 3 |

### 8.5.2 Baselines

We compare our proposed JAME with the following relevant baselines methods:

- **LANE:** This method integrates attributes and labels into the shared representations through eigen-decomposition of the graph affinity, attribute affinity, and label affinity matrices [HLH17b].
- **CAN:** This embedding model employs a variational autoencoder with an inference model for encoding attributes into Gaussian distribution and a generative model for reconstructing both edges and attributes [MLBZ19].
- **attri2vec:** It is to unify network structure and attributes together seamlessly via a transformation function [ZYZZ19]. DeepWalk model [PARS14] is used to preserve network structure by forcing similar neighbors being mapped closely in the attribute subspace.
- **DANE:** This model learns representation preserving proximity of both topological structure and node attributes via a joint autoencoder model [GH18].

### 8.5.3 Parameter Settings

We follow the authors' suggested settings available in the original baseline papers as follows. DANE builds a higher-order proximity matrix with a window size of $w = 10$, the walk length of $l = 80$, and the number of walks of $\gamma = 10$ over 500 iterations. CAN sets the hyperparameters of prior distributions, $\sigma_{\mathcal{V}}, \sigma_{\mathcal{A}}$ to 1, and training iterations to 200. In attri2vec, parameters are set as $w = 10$, $l = 100$, $\gamma = 40$, and the number of iterations up to a million. LANE optimizes three objective functions which are manually weighted to 1, $\alpha_1 > 1$ and $\alpha_2 > 10$. In JAME, we set the dimension size for shared embeddings to $d = 128$ and for hidden layers to $s = 300$. We optimize the model parameters using Adam optimizer [KB14] with the learning rate of 0.001.

### 8.5.4 Multi-task Weighting

The initial loss weights are set to 1.0, then the model is expected to update these weights during training. Table 8.2 shows the change of loss weights in which the

Table 8.2: Multi-task weighting based on task difficulty across different datasets.

| Iteration | Cora | | | Citeseer | | | PubMed | | |
|---|---|---|---|---|---|---|---|---|---|
| | $w_a$ | $w_x$ | $w_y$ | $w_a$ | $w_x$ | $w_y$ | $w_a$ | $w_x$ | $w_y$ |
| 1 | 0.933 | 0.942 | **0.946** | 0.921 | 0.932 | **0.935** | 0.913 | 0.926 | **0.933** |
| 2 | 0.876 | 0.889 | **0.922** | 0.842 | 0.857 | **0.925** | 0.868 | 0.886 | **0.928** |
| 3 | 0.801 | 0.823 | **0.919** | 0.762 | 0.783 | **0.926** | 0.799 | 0.875 | **0.922** |
| 4 | 0.735 | 0.762 | **0.917** | 0.673 | 0.711 | **0.924** | 0.660 | 0.764 | **0.920** |
| 5 | 0.672 | 0.708 | **0.919** | 0.596 | 0.645 | **0.923** | 0.520 | 0.567 | **0.917** |
| 6 | 0.615 | 0.634 | **0.916** | 0.516 | 0.568 | **0.921** | 0.483 | 0.553 | **0.911** |
| 7 | 0.533 | 0.583 | **0.913** | 0.432 | 0.482 | **0.922** | 0.455 | 0.542 | **0.908** |
| 8 | 0.464 | 0.535 | **0.914** | 0.358 | 0.436 | **0.919** | 0.423 | 0.517 | **0.901** |
| 9 | 0.403 | 0.477 | **0.912** | 0.264 | 0.348 | **0.918** | 0.342 | 0.432 | **0.898** |
| 10 | 0.335 | 0.418 | **0.910** | 0.221 | 0.283 | **0.916** | 0.313 | 0.415 | **0.897** |

hardest task gets the higher weights. This is visible that reconstructing the group labels is the hardest task in our datasets and gains consistently the highest weight over 10 iterations. As shown in Figure 8.2, the final loss naturally drops and takes a few iterations to converge for all datasets.

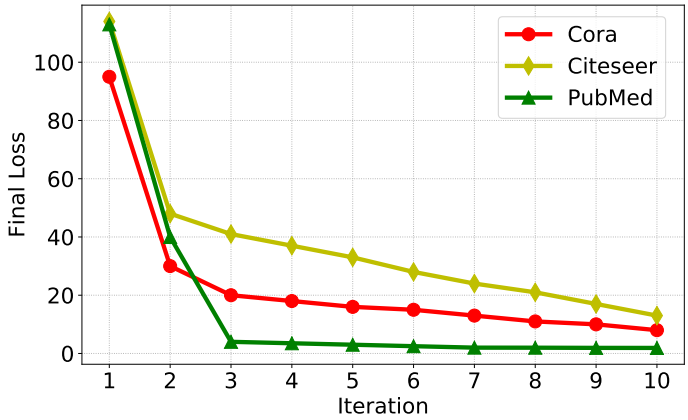



Figure 8.2: Change of final loss during training on all datasets.

### 8.5.5 Node Classification

Node classification has been widely adopted to validate the quality of network embedding [GF18]. Similar to LANE [HLH17b], we manage to divide $n$ nodes randomly into a training $(A_{train}, X_{train}, Y_{train})$ and a test set $(A_{test}, X_{test}, Y_{test})$. The goal is to predict the label of each node in the test set, given the network structure and attributes. We use the JAME architecture itself for node classification observing 10% to 90% of labels at random and predicting the remaining. In the other baselines, we directly apply logistic regression as the main classifier and Macro-F1 score as the performance measurement [GF18]. Table 8.3 shows our proposed JAME consistently outperforms the baselines which confirms the shared layer and optimal

weighting tailored for better embedding. Our multi-task setting enforces tasks to compensate each other by learning from the shared information leveraged from the network. Due to optimal task weighting, label embedding can be trained as effective as others which ends at better classification results. DANE is the closest competitor as employs autoencoders to jointly encode link and attribute information in an effective manner. Overall, the presented results reveal the effectiveness and robustness of JAME to capture and encode the label information associated with the network.

Table 8.3: Node classification performance observing different percentages of labeled data.

| **Dataset** | **Method** | **Macro-F1** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | 10% | 20% | 30% | 40% | 50% | 60% | 70% | 80% | 90% |
| Cora | JAME | **0.816** | **0.824** | **0.839** | **0.847** | **0.855** | **0.868** | **0.879** | **0.885** | **0.891** |
| | LANE | 0.663 | 0.671 | 0.684 | 0.715 | 0.726 | 0.739 | 0.753 | 0.785 | 0.799 |
| | CAN | 0.733 | 0.752 | 0.768 | 0.773 | 0.788 | 0.794 | 0.806 | 0.814 | 0.822 |
| | DANE | 0.778 | 0.795 | 0.812 | 0.822 | 0.837 | 0.854 | 0.861 | 0.869 | 0.877 |
| | attri2vec | 0.695 | 0.713 | 0.729 | 0.732 | 0.746 | 0.767 | 0.788 | 0.792 | 0.806 |
| Citeseer | JAME | **0.731** | **0.739** | **0.755** | **0.778** | **0.786** | **0.790** | **0.798** | **0.804** | **0.810** |
| | LANE | 0.503 | 0.549 | 0.570 | 0.579 | 0.581 | 0.591 | 0.606 | 0.624 | 0.636 |
| | CAN | 0.577 | 0.606 | 0.613 | 0.619 | 0.628 | 0.632 | 0.638 | 0.641 | 0.642 |
| | DANE | 0.604 | 0.633 | 0.671 | 0.678 | 0.696 | 0.705 | 0.723 | 0.735 | 0.745 |
| | attri2vec | 0.556 | 0.571 | 0.614 | 0.650 | 0.656 | 0.662 | 0.670 | 0.666 | 0.682 |
| PubMed | JAME | **0.868** | **0.873** | **0.879** | **0.881** | **0.884** | **0.886** | **0.888** | **0.894** | **0.898** |
| | LANE | 0.787 | 0.792 | 0.795 | 0.799 | 0.820 | 0.824 | 0.828 | 0.833 | 0.837 |
| | CAN | 0.793 | 0.796 | 0.801 | 0.809 | 0.812 | 0.816 | 0.820 | 0.825 | 0.829 |
| | DANE | 0.857 | 0.864 | 0.870 | 0.873 | 0.874 | 0.876 | 0.878 | 0.880 | 0.882 |
| | attri2vec | 0.843 | 0.850 | 0.853 | 0.858 | 0.860 | 0.862 | 0.863 | 0.865 | 0.866 |

### 8.5.6 Link Prediction

Link prediction is a well-studied analytic task that exposes the meaningfulness of vector representations for preserving local connections [GF18]. We follow Grover et al. [GL16] instructions for link prediction which gathers the equal percentage of positive and negative edges. The positive examples are obtained by randomly removing 50% of real edges from the original network. Whereas negative examples are node pairs without any connection (non-existing edges). We first construct edge features by applying Hadamard product [GL16] on vectors of incident nodes, then feed a logistic regression. Table 8.4 shows the performance measured by area under ROC curve (AUC) and average precision (AP). As can be seen, JAME obtains highly competitive performance with CAN to predict the missing links due to the effect of labels and attributes. The architecture of JAME allows shared embeddings to encode more meaningful vectors by observing links, attribute, and labels simultaneously. CAN is in high competition with JAME as it exploits variational autoencoders along with a GCN layer [KW16a] which effectively seizes nonlinearity

and local connections of the graph structure.

Table 8.4: Comparison of the baseline methods on link prediction.

| **Method** | **Cora** | | **Citeseer** | | **PubMed** | |
|---|---|---|---|---|---|---|
| | AUC | AP | AUC | AP | AUC | AP |
| JAME | **0.987** | **0.972** | **0.985** | **0.979** | **0.983** | **0.980** |
| LANE | 0.860 | 0.857 | 0.771 | 0.764 | 0.883 | 0.879 |
| CAN | 0.985 | 0.971 | 0.984 | 0.977 | 0.980 | 0.977 |
| DANE | 0.897 | 0.886 | 0.938 | 0.923 | 0.969 | 0.951 |
| attri2vec | 0.872 | 0.867 | 0.796 | 0.791 | 0.890 | 0.885 |

### 8.5.7 Attribute Inference

Attribute inference is to predict attributes associated with each node applying logistic regression to the learned representations. We follow CAN [MLBZ19] experimental setting for attribute inference which randomly divides nodes into a training (80%), and a test set (20%). AUC and AP are used to measure the performance due to 0/1-valued presentation of data. As shown in Table 8.5, JAME achieves improvements in both AUC and AP over the baseline methods. This can be explained by the fact that attributes and labels are correlated in our datasets.

Table 8.5: Comparison of the baseline methods on attribute inference.

| **Method** | **Cora** | | **Citeseer** | | **PubMed** | |
|---|---|---|---|---|---|---|
| | AUC | AP | AUC | AP | AUC | AP |
| JAME | **0.968** | **0.961** | **0.971** | **0.963** | **0.681** | **0.677** |
| LANE | 0.872 | 0.867 | 0.880 | 0.875 | 0.590 | 0.587 |
| CAN | 0.932 | 0.916 | 0.954 | 0.939 | 0.670 | 0.652 |
| DANE | 0.917 | 0.910 | 0.928 | 0.921 | 0.639 | 0.635 |
| attri2vec | 0.881 | 0.878 | 0.889 | 0.883 | 0.610 | 0.598 |

### 8.5.8 Efficiency Evaluation

We record the runtime (in seconds) of all methods increasing the number of nodes on the PubMed dataset. Our experiments are performed on an Intel(R) CPU machine with a Core(TM) i5 processor and 32GB of RAM. Due to the high computation time in attri2vec (around 6.6 hours), we limit our focus on the other works with more reasonable runtime. Figure 8.3 shows JAME consistently learns embeddings faster than LANE, DANE, and CAN for the given number of nodes. As the number of nodes increases the performance difference in time also raises up. DANE and CAN

both employ autoencoders, however more iterations are needed until convergence. Overall, the obtained results demonstrates efficiency and scalability of the proposed model.

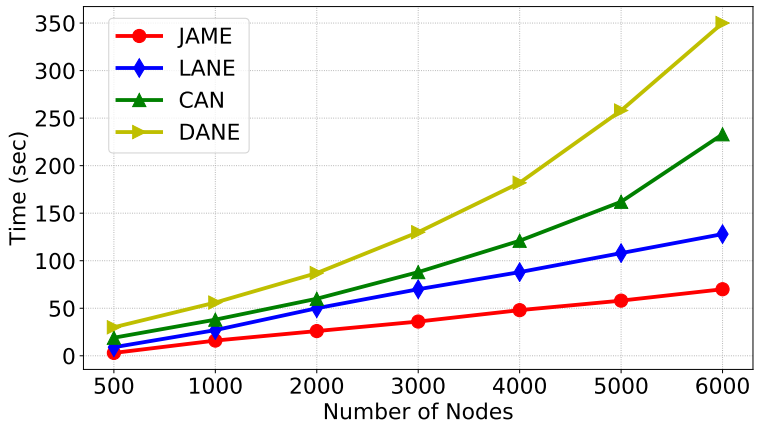



Figure 8.3: Runtime comparison on the PubMed dataset.

## 8.6 Conclusion

Labeled data and content are critical information available in today's social networks. Multi-task learning architectures provide a shared structure across embedding tasks which integrate network structure and attributes. In this chapter, we designed a joint model called JAME in which a shared layer and a loss weighting layer solicit for better representations. Our extensive experiments revealed that JAME incorporates data from different sources within linear runtime complexity.

Despite the strong experimental results, our method suffers from a few limitations. Foremost, autoencoders tend to be data-specific, in the sense that their utility totally depends on input data. Our experiments are limited to citation networks, more diverse datasets with different sizes are needed to validate the effectiveness of our method. Also, it is not clear how the experimental results change if DANE and CAN involve labels in their autoencoders. This is particularly needed to examine how link prediction and attribute inference change once observing node labels.

For future work, we aim at extending JAME to the heterogeneous one, where the network contains multiple types of nodes and edges. We can work further on analytic theories that quantitatively measure how different tasks can be combined in a multi-task setting other than autoencoders.

# Chapter 9

# Conclusion

In this thesis, we studied network embedding for processing social network data, primarily, to propose several contributions to this nascent but fast evolving field. Throughout the thesis, we have seen that many core concepts (e.g. deep learning models, graph types) can be taken over to solve social network analytical tasks. We have touched upon different types of social networks (e.g. signed, attributed) exploring current models or drawing new models. The young field offered many unexplored paths to tread where some of them had a dead end, but some others led us to effective experimental results for network property valuation and data integration. However, our proposed methods are hardly perfect and we tried to remain critical in stating their shortcomings in each chapter.

Deep learning is a booming topic and there are often numerous lines of work proposed independently. Particularly in deep learning on graphs, the research work usually comes from several separate communities working in natural language processing, information retrieval, signal processing, computer vision, and in biomedical computing. Therefore, we attempted to gather a somewhat fused but not exhaustive picture which equips the reader with a core understanding of the context of our work.

The rest of the chapter recapitulates our contributions and concludes with an outlook over possible future directions.

The first contribution was to learn embeddings for fine-grained structures so-called ego networks. The key idea was to deploy Paragraph Vector in graphs which results a compact representation for an ego network. Throughout the experimental tasks, namely social circle prediction and event attendance inference, we showed the importance of ego network embedding for solving analytic tasks in a fast and precise manner.

The second contribution stands for exploring the content of vector embeddings to reveal what they retain in terms of network topological properties. The major insight was the formulation of the problem through a learning to rank approach in which model weights indicate the importance of topological features in subgraph

level. Important ingredients were the partitioning input graph into proper test and training sets and efficient pairwise structural similarity measure to build the feature matrix for the ranking algorithm. We also tried to retrieve centrality values in vertex level directly by a linear regression. This was the first time that network analytic problems like centrality and distance are scrutinized by neural graph embedding models.

Finally, the third contribution lies at the core of deep learning usage in specific scenarios such as modeling link heterogeneity and side information in social networks. The major contribution was to adjust the objective function of LINE [TQW+15] or stack autoencoders to capture certain information stashed in the network. Our motivation was to optimize and speed up data integration for the heterogeneous source of information and eventually facilitate subsequent analytic tasks. We believe our work has helped to spark initial interest in multi-task learning on networked data specifically with adapting optimal weights for learning tasks. The shared representations fairly improved the performance compared to the state-of-the-art methods over widely used datasets.

In a nutshell, we have ventured a journey from social graphs to their embeddings and potential applications. We attempted to develop this research with the hope that the context provides new insights about some of the challenges in the emergent field of deep learning on social graphs. That brings one more step closer to solving real-world problems analyzing social media.

## 9.1 Future Directions

Although many techniques have been developed to facilitate social network analysis, there still remain several promising directions which need further exploration.

### 9.1.1 Scalability

The neural based network embedding has achieved substantial performances due to high capacity, however they still suffer from the problem of efficiency. This problem specifically becomes more severe as dealing with real-world massive networks with billions of nodes. Designing scalable representation learning models is another driving factor to proceed in this domain [BBL+17]. Furthermore, developing computational paradigms like using GPUs for network processing can be an alternative way toward efficiency improvement.

### 9.1.2 Interpretability

Although we attempted to explore the content of embeddings, lack of interpretability [LHLH18] in representation learning is still a pending issue. The meaning of

different dimensions in vectors is not clear, hence the underlying factors hidden in the latent space is difficult to perceive. Therefore, more works are needed on how to understand the model output, how to design interpretable embedding models, and how to utilize interpretations. In specific, interpretability will help to encode more meaningful and task-specific representations toward various social network analysis problems.

### 9.1.3 Hierarchical Network Structure

Social networks can present a hierarchical structure composed by individuals who play important roles in a society such as managers, politicians, celebrities, or decision makers. The existing network embedding techniques mainly focus on capturing node pairwise relationships not the global hierarchical structure available in complex networks [BGL16]. Therefore, how to build effective embedding models which preserve global hierarchical structures is a promising direction as for further work.

### 9.1.4 Dynamic Networks

Social networks often show a highly dynamic nature evolving over time by changing the set of nodes, the underlying network structure, or attribute information. Therefore, the existing static methods can fail in encoding these networks properly. The current techniques often rely on certain assumptions, such as fixed set of nodes with dynamics on edge deletion and addition [LDH+17], but the change on attribute information is ignored. Therefore, how to develop a specific embedding method for real dynamic networks remains an open question.

### 9.1.5 Heterogeneous Networks

Most of the existing heterogeneous embedding models deal with node heterogeneity. However, embedding for networks with link or content heterogeneity is still at the early stage. Thus, more comprehensive techniques are required to fully capture semantics of different types of elements in complex networks.

# Bibliography


[AB02] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002. (cited on page 48.)

[ACKP13] Bruno Abrahao, Flavio Chierichetti, Robert Kleinberg, and Alessandro Panconesi. Trace complexity of network inference. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 491–499, 2013. (cited on page 54.)

[ACLG+16] Valerio Arnaboldi, Marco Conti, Massimiliano La Gala, Andrea Passarella, and Fabio Pezzoni. Ego network structure in online social networks and its impact on information diffusion. *Computer Communications*, 76:26–41, 2016. (cited 3 times on pages 2, 24, and 43.)

[Aga] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*. (cited on page 57.)

[AHZ11] Mohammad Al Hasan and Mohammed J Zaki. A survey of link prediction in social networks. In *Social network data analytics*, pages 243–275. Springer, 2011. (cited on page 20.)

[AIY13] Takuya Akiba, Yoichi Iwata, and Yuichi Yoshida. Fast exact shortest-path distance queries on large networks by pruned landmark labeling. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 349–360. ACM, 2013. (cited on page 60.)

[ALGPC14] Valerio Arnaboldi, Massimiliano La Gala, Andrea Passarella, and Marco Conti. The role of trusted relationships on content spread in distributed online social networks. In *European Conference on Parallel Processing*, pages 287–298. Springer, 2014. (cited on page 24.)

[ALGPC16] Valerio Arnaboldi, Massimiliano La Gala, Andrea Passarella, and Marco Conti. Information diffusion in distributed osn: The impact of trusted relationships. *Peer-to-Peer Networking and Applications*, 9(6):1195–1208, 2016. (cited on page 24.)

[AYGC16] Qingyao Ai, Liu Yang, Jiafeng Guo, and W Bruce Croft. Analysis of the paragraph vector model for information retrieval. In *Proceedings of the 2016 ACM international conference on the theory of information retrieval*, pages 133–142, 2016. (cited on page 25.)

[AZRP18] Bijaya Adhikari, Yao Zhang, Naren Ramakrishnan, and B Aditya Prakash. Sub2vec: Feature learning for subgraphs. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 170–182. Springer, 2018. (cited on page 32.)

[BB03] Albert-László Barabási and Eric Bonabeau. Scale-free networks. *Scientific american*, 288(5):60–69, 2003. (cited on page 85.)

[BBBG10] Björn Bringmann, Michele Berlingerio, Francesco Bonchi, and Arisitdes Gionis. Learning and predicting the evolution of social networks. *IEEE Intelligent Systems*, 25(4):26–35, 2010. (cited on page 43.)

[BBDT] Vanda Balogh, Gábor Berend, Dimitrios I Diochnos, and György Turán. Understanding the semantic content of sparse word embeddings using a commonsense knowledge base. (cited on page 45.)

[BBL+17] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017. (cited on page 93.)

[BCM11] Smriti Bhagat, Graham Cormode, and S Muthukrishnan. Node classification in social networks. In *Social network data analytics*, pages 115–148. Springer, 2011. (cited 2 times on pages 1 and 7.)

[BDS+10] Reinhard Bauer, Daniel Delling, Peter Sanders, Dennis Schieferdecker, Dominik Schultes, and Dorothea Wagner. Combining hierarchical and goal-directed speed-up techniques for dijkstra's algorithm. *Journal of Experimental Algorithmics (JEA)*, 15:2–1, 2010. (cited on page 56.)

[BFJ+12] Robert M Bond, Christopher J Fariss, Jason J Jones, Adam DI Kramer, Cameron Marlow, Jaime E Settle, and James H Fowler. A 61-million-person experiment in social influence and political mobilization. *Nature*, 489(7415):295, 2012. (cited on page 33.)

[BG17] Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. *arXiv preprint arXiv:1707.03815*, 2017. (cited on page 7.)

[BGL11] Albert-László Barabási, Natali Gulbahce, and Joseph Loscalzo. Network medicine: a network-based approach to human disease. *Nature reviews genetics*, 12(1):56–68, 2011. (cited on page 7.)

[BGL16] Austin R Benson, David F Gleich, and Jure Leskovec. Higher-order organization of complex networks. *Science*, 353(6295):163–166, 2016. (cited on page 94.)

[BH17] Aïcha BenTaieb and Ghassan Hamarneh. Uncertainty driven multi-loss fully convolutional networks for histopathology. In *Intravascular Imaging and Computer Assisted Stenting, and Large-Scale Annotation of Biomedical Data and Expert Label Synthesis*, pages 155–163. Springer, 2017. (cited 2 times on pages 80 and 81.)

[BK09] Joonhyun Bae and Sangwook Kim. A global social graph as a hybrid hypergraph. In *2009 Fifth International Joint Conference on INC, IMS and IDC*, pages 1025–1031. IEEE, 2009. (cited on page 24.)

[BKB+19] Stephen Bonner, Ibad Kureshi, John Brennan, Georgios Theodoropoulos, Andrew Stephen McGough, and Boguslaw Obara. Exploring the semantic content of unsupervised graph embeddings: an empirical study. *Data Science and Engineering*, 4(3):269–289, 2019. (cited 2 times on pages 45 and 51.)

[BKBM18] Sambaran Bandyopadhyay, Harsh Kara, Anirban Biswas, and M Narasimha Murty. Sac2vec: Information network representation with structure and content. *arXiv preprint arXiv:1804.10363*, 2018. (cited on page 86.)

[BKWG17] Ershad Banijamali, Amir-Hossein Karimi, Alexander Wong, and Ali Ghodsi. Jade: Joint autoencoders for dis-entanglement. *arXiv preprint arXiv:1711.09163*, 2017. (cited on page 80.)

[BL07] John A Bullinaria and Joseph P Levy. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior research methods*, 39(3):510–526, 2007. (cited on page 12.)

[BMP15] Federico Botta, Helen Susannah Moat, and Tobias Preis. Quantifying crowd size with mobile phone and twitter data. *Royal Society open science*, 2(5):150162, 2015. (cited on page 35.)

[BN01] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in neural information processing systems*, 14:585–591, 2001. (cited on page 8.)

[BOF+97] Susan H Bland, Erin S O'leary, Eduardo Farinaro, Fabrizio Jossa, Vittorio Krogh, John M Violanti, and Maurizio Trevisan. Social network disturbances and psychological distress following earthquake evacuation. *The Journal of nervous and mental disease*, 185(3):188–195, 1997. (cited on page 1.)

[Bol14] Raja Ashok Bolla. Crime pattern detection using online social media. 2014. (cited on page 1.)

[BSZG18] Aleksandar Bojchevski, Oleksandr Shchur, Daniel Zügner, and Stephan Günnemann. Netgan: Generating graphs via random walks. *arXiv preprint arXiv:1803.00816*, 2018. (cited on page 14.)

[BUGD+13] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795, 2013. (cited on page 16.)

[BV14] Paolo Boldi and Sebastiano Vigna. Axioms for centrality. *Internet Mathematics*, 10(3-4):222–262, 2014. (cited on page 54.)

[BXM+13] Pavel Berkhim, Zhichen Xu, Jianchang Mao, Daniel E Rose, Abe Taha, and Farzin Maghoul. Trust propagation through both explicit and implicit social networks, May 14 2013. US Patent 8,442,978. (cited on page 1.)

[Car97] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997. (cited on page 79.)

[CBLR17] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. *arXiv preprint arXiv:1711.02257*, 2017. (cited 2 times on pages 80 and 81.)

[CCF08] Ethan Cohen-Cole and Jason M Fletcher. Is obesity contagious? social networks vs. environmental factors in the obesity epidemic. *Journal of health economics*, 27(5):1382–1387, 2008. (cited on page 1.)

[CDF+13] Edith Cohen, Daniel Delling, Fabian Fuchs, Andrew V Goldberg, Moises Goldszmidt, and Renato F Werneck. Scalable similarity estimation in social networks: Closeness, node labels, and random edge lengths. In *Proceedings of the first ACM conference on Online social networks*, pages 131–142, 2013. (cited on page 54.)

[CDR16] Cesar Cadena, Anthony R Dick, and Ian D Reid. Multi-modal auto-encoders as joint estimators for robotics scene understanding. In *Robotics: Science and Systems*, volume 5, page 1, 2016. (cited on page 80.)

[CF07] Nicholas A Christakis and James H Fowler. The spread of obesity in a large social network over 32 years. *New England journal of medicine*, 357(4):370–379, 2007. (cited on page 33.)

[CG97] Fan RK Chung and Fan Chung Graham. *Spectral graph theory*. Number 92. American Mathematical Soc., 1997. (cited on page 81.)

[CG16] Harish Chintakunta and Athanasios Gentimis. Influence of topology in information flow in social networks. In *2016 50th Asilomar Conference on Signals, Systems and Computers*, pages 67–71. IEEE, 2016. (cited on page 33.)

[CH56] Dorwin Cartwright and Frank Harary. Structural balance: a generalization of heider's theory. *Psychological review*, 63(5):277, 1956. (cited on page 69.)

[Cha12] Timothy M Chan. All-pairs shortest paths for unweighted undirected graphs in o (mn) time. *ACM Transactions on Algorithms (TALG)*, 8(4):1–17, 2012. (cited on page 54.)

[CHT+15] Shiyu Chang, Wei Han, Jiliang Tang, Guo-Jun Qi, Charu C Aggarwal, and Thomas S Huang. Heterogeneous network embedding via deep architectures. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 119–128, 2015. (cited 3 times on pages 15, 67, and 68.)

[CLX15] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM international on conference on information and knowledge management*, pages 891–900, 2015. (cited 2 times on pages 1 and 21.)

[CLX16] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Deep neural networks for learning graph representations. In *Thirtieth AAAI conference on artificial intelligence*, 2016. (cited on page 12.)

[CMX18] Jie Chen, Tengfei Ma, and Cao Xiao. Fastgcn: fast learning with graph convolutional networks via importance sampling. *arXiv preprint arXiv:1801.10247*, 2018. (cited on page 13.)

[CNM04] Aaron Clauset, Mark EJ Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical review E*, 70(6):066111, 2004. (cited on page 48.)

[CPHS18] Haochen Chen, Bryan Perozzi, Yifan Hu, and Steven Skiena. Harp: Hierarchical representation learning for networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. (cited 4 times on pages 12, 33, 34, and 37.)

[CS17] Ting Chen and Yizhou Sun. Task-guided and path-augmented heterogeneous network embedding for author identification. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 295–304, 2017. (cited 2 times on pages 8 and 67.)

[CSG+19] Keting Cen, Huawei Shen, Jinhua Gao, Qi Cao, Bingbing Xu, and Xueqi Cheng. Node classification framework, 2019. (cited on page 86.)

[CTLY09] Jianhui Chen, Lei Tang, Jun Liu, and Jieping Ye. A convex formulation for learning shared structures from multiple tasks. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 137–144. ACM, 2009. (cited on page 79.)

[CW08] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167, 2008. (cited on page 8.)

[CW17] Liwei Cai and William Yang Wang. Kbgan: Adversarial learning for knowledge graph embeddings. *arXiv preprint arXiv:1711.04071*, 2017. (cited on page 14.)

[CWW10] Wei Chen, Chi Wang, and Yajun Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1029–1038, 2010. (cited on page 20.)

[CZC+17] Sandro Cavallari, Vincent W Zheng, Hongyun Cai, Kevin Chen-Chuan Chang, and Erik Cambria. Learning community embedding with community detection and node embedding on graphs. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 377–386. ACM, 2017. (cited on page 7.)

[CZY11] Jianhui Chen, Jiayu Zhou, and Jieping Ye. Integrating low-rank and group-sparse structures for robust multi-task learning. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 42–50. ACM, 2011. (cited on page 79.)

[DA10] Mari Carmen Domingo Aladrén. Managing healthcare through social networks. *Computer*, 43(7):20–25, 2010. (cited on page 1.)

[DBS07] Laura Dietz, Steffen Bickel, and Tobias Scheffer. Unsupervised prediction of citation influences. In *Proceedings of the 24th international conference on Machine learning*, pages 233–240, 2007. (cited 2 times on pages 1 and 20.)

[DBV16] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852, 2016. (cited on page 13.)

[DCS17] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 135–144, 2017. (cited 5 times on pages 12, 15, 16, 67, and 69.)

[DDS16] Hanjun Dai, Bo Dai, and Le Song. Discriminative embeddings of latent variable models for structured data. In *International conference on machine learning*, pages 2702–2711, 2016. (cited on page 20.)

[DG08] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008. (cited on page 7.)

[DLG15] Karmen Lata Dykstra, Jefrey Lijffijt, and Aristides Gionis. Covering the egonet: A crowdsourcing approach to social circle discovery on twitter. In *Ninth International AAAI Conference on Web and Social Media*, 2015. (cited on page 25.)

[dLMO+17] Vinicius Monteiro de Lira, Craig Macdonald, Iadh Ounis, Raffaele Perego, Chiara Renso, and Valeria Cesario Times. Exploring social media for event attendance. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, pages 447–450. ACM, 2017. (cited 4 times on pages 34, 35, 38, and 39.)

[DLTW18] Quanyu Dai, Qiang Li, Jian Tang, and Dan Wang. Adversarial network embedding. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. (cited on page 12.)

[DOL15] Andrew M Dai, Christopher Olah, and Quoc V Le. Document embedding with paragraph vectors. *arXiv preprint arXiv:1507.07998*, 2015. (cited on page 25.)

[DSRZ13] Nan Du, Le Song, Manuel Gomez Rodriguez, and Hongyuan Zha. Scalable influence estimation in continuous-time diffusion networks. In *Advances in neural information processing systems*, pages 3147–3155, 2013. (cited on page 54.)

[DTSS10] Cora I Dăniasă, Vasile Tomiţă, Dragoş Stuparu, and Marieta Stanciu. The mechanisms of the influence of viral marketing in social media. *Economics, Management & Financial Markets*, 5(3):278–282, 2010. (cited on page 1.)

[DYDA11] George E Dahl, Dong Yu, Li Deng, and Alex Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on audio, speech, and language processing*, 20(1):30–42, 2011. (cited on page 8.)

[DYM+14] Rong Du, Zhiwen Yu, Tao Mei, Zhitao Wang, Zhu Wang, and Bin Guo. Predicting activity attendance in event-based social networks: Content, context and social influence. In *Proceedings of the 2014 ACM international joint conference on pervasive and ubiquitous computing*, pages 425–434, 2014. (cited 2 times on pages 33 and 36.)

[EJS76] Roger C Entringer, Douglas E Jackson, and DA Snyder. Distance in graphs. *Czechoslovak Mathematical Journal*, 26(2):283–296, 1976. (cited on page 2.)

[FDMCF14] Emilio Ferrara, Pasquale De Meo, Salvatore Catanese, and Giacomo Fiumara. Detecting criminal organizations in mobile phone networks. *Expert Systems with Applications*, 41(13):5733–5750, 2014. (cited on page 7.)

[FLL17] Tao-yang Fu, Wang-Chien Lee, and Zhen Lei. Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1797–1806, 2017. (cited 2 times on pages 67 and 69.)

[FMWFM11] Enrique Frias-Martinez, Graham Williamson, and Vanessa Frias-Martinez. An agent-based model of epidemic spread using human mobility and social network information. In *2011 IEEE third international conference on privacy, security, risk and trust and 2011 IEEE third international conference on social computing*, pages 57–64. IEEE, 2011. (cited on page 1.)

[For10] Santo Fortunato. Community detection in graphs. *Physics reports*, 486(3-5):75–174, 2010. (cited on page 2.)

[Fre77] Linton C Freeman. A set of measures of centrality based on betweenness. *Sociometry*, pages 35–41, 1977. (cited on page 54.)

[Fre11] Mark Freeman. Fire, wind and water: Social networks in natural disasters. *Journal of Cases on Information Technology (JCIT)*, 13(2):69–79, 2011. (cited on page 1.)

[FRS17] Daniel R Figueiredo, Leonardo FR Ribeiro, and Pedro HP Saverese. struc2vec: Learning node representations from structural identity. *arXiv preprint arXiv:1704.03165*, 2017. (cited on page 12.)

[FS14] Marina Fiedler and Marko Sarstedt. Influence of community design on user behaviors in online communities. *Journal of Business Research*, 67(11):2258–2268, 2014. (cited on page 35.)

[FT] Berthy Feng and Divya Thuremella. A tale of two encodings: Comparing bag-of-words and word2vec for vqa. (cited on page 36.)

[GBB11] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323, 2011. (cited 2 times on pages 37 and 57.)

[GBH18] Octavian-Eugen Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic entailment cones for learning hierarchical embeddings. *arXiv preprint arXiv:1804.01882*, 2018. (cited on page 40.)

[GBSW10] Andrey Gubichev, Srikanta Bedathur, Stephan Seufert, and Gerhard Weikum. Fast and accurate estimation of shortest paths in large graphs. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 499–508, 2010. (cited 2 times on pages 56 and 59.)

[GER08] Brian Gallagher and Tina Eliassi-Rad. Leveraging label-independent features for classification in sparsely labeled networks: An empirical study. In *International Workshop on Social Network Mining and Analysis*, pages 1–19. Springer, 2008. (cited on page 7.)

[GF17] Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *arXiv preprint arXiv:1705.02801*, 2017. (cited 2 times on pages 34 and 58.)

[GF18] Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94, 2018. (cited 3 times on pages 80, 88, and 89.)

[GGLZ17] Weiwei Gu, Li Gong, Xiaodan Lou, and Jiang Zhang. The hidden flow structure and metric space of network embedding algorithms based on random walks. *Scientific reports*, 7(1):1–12, 2017. (cited 2 times on pages 55 and 62.)

[GH18] Hongchang Gao and Heng Huang. Deep attributed network embedding. In *IJCAI*, volume 18, pages 3364–3370, 2018. (cited 5 times on pages 18, 79, 80, 86, and 87.)

[GK04] A Grabowski and RA Kosiński. Epidemic spreading in a hierarchical social network. *Physical Review E*, 70(3):031908, 2004. (cited on page 1.)

[GL16] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864. ACM, 2016. (cited 18 times on pages 1, 7, 10, 12, 16, 20, 25, 30, 33, 37, 45, 55, 57, 59, 68, 70, 74, and 89.)

[GLO+16] Yanming Guo, Yu Liu, Ard Oerlemans, Songyang Lao, Song Wu, and Michael S Lew. Deep learning for visual understanding: A review. *Neurocomputing*, 187:27–48, 2016. (cited on page 8.)

[GPAM+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014. (cited on page 14.)

[GPH19] Hongchang Gao, Jian Pei, and Heng Huang. Progan: Network embedding via proximity generative adversarial network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1308–1316, 2019. (cited on page 86.)

[Gut04] Ronald J Gutman. Reach-based routing: A new approach to shortest path algorithms optimized for road networks. *ALENEX/ANALC*, 4:100–111, 2004. (cited on page 56.)

[GW05] Andrew V Goldberg and Renato Fonseca F Werneck. Computing point-to-point shortest paths from external memory. In *ALENEX/ANALCO*, pages 26–40. Citeseer, 2005. (cited on page 56.)

[GZC+09] Ido Guy, Naama Zwerdling, David Carmel, Inbal Ronen, Erel Uziel, Sivan Yogev, and Shila Ofek-Koifman. Personalized recommendation of social software items based on social relations. In *Proceedings of the third ACM conference on Recommender systems*, pages 53–60, 2009. (cited on page 1.)

[GZZ+13] Zhen Guo, Zhongfei Mark Zhang, Shenghuo Zhu, Yun Chi, and Yihong Gong. A two-level topic model towards knowledge discovery from citation networks. *IEEE Transactions on Knowledge and Data Engineering*, 26(4):780–794, 2013. (cited on page 20.)

[HAMH16] Renjun Hu, Charu C Aggarwal, Shuai Ma, and Jinpeng Huai. An embedding approach to anomaly detection. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pages 385–396. IEEE, 2016. (cited 2 times on pages 1 and 21.)

[Har54] Zellig S Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954. (cited on page 36.)

[HC14] Mostafa Haghir Chehreghani. Effective co-betweenness centrality computation. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 423–432, 2014. (cited on page 54.)

[HDY+12] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012. (cited on page 8.)

[Hea08] Jeff Heaton. *Introduction to neural networks with Java*. Heaton Research, Inc., 2008. (cited on page 37.)

[HKP12] Julia Heidemann, Mathias Klier, and Florian Probst. Online social networks: A survey of a global phenomenon. *Computer networks*, 56(18):3866–3878, 2012. (cited on page 24.)

[HLH17a] Xiao Huang, Jundong Li, and Xia Hu. Accelerated attributed network embedding. In *Proceedings of the 2017 SIAM international conference on data mining*, pages 633–641. SIAM, 2017. (cited on page 79.)

[HLH17b] Xiao Huang, Jundong Li, and Xia Hu. Label informed attributed network embedding. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 731–739. ACM, 2017. (cited 4 times on pages 79, 81, 87, and 88.)

[HLJ15] Yu-Yueh Huang, Cheng-Te Li, and Shyh-Kang Jeng. Mining location-based social networks for criminal activity prediction. In *2015 24th Wireless and Optical Communication Conference (WOCC)*, pages 185–189. IEEE, 2015. (cited on page 1.)

[HM17] Zhipeng Huang and Nikos Mamoulis. Heterogeneous information network embedding for meta path based proximity. *arXiv preprint arXiv:1701.05291*, 2017. (cited 2 times on pages 67 and 69.)

[HMK04] David Heckerman, Christopher Meek, and Daphne Koller. Probabilistic models for relational data. Technical report, Technical Report MSR-TR-2004-30, Microsoft Research, 2004. (cited on page 1.)

[HNR68] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968. (cited on page 54.)

[HOT06] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006. (cited on page 8.)

[HVD15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. (cited on page 85.)

[HVG11] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011. (cited on page 13.)

[HYL17] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pages 1024–1034, 2017. (cited 2 times on pages 1 and 20.)

[HZL+16] Qinglai He, Min Zhu, Binbin Lu, Hanqing Liu, and Qiaomu Shen. Mena: Visual analysis of multivariate egocentric network evolution. In *2016 International Conference on Virtual Reality and Visualization (ICVRV)*, pages 488–496. IEEE, 2016. (cited on page 24.)

[IK17] Ryota Iijima and Yuichiro Kamada. Social distance and network structures. *Theoretical Economics*, 12(2):655–689, 2017. (cited on page 54.)

[IPR18] Mohammad Raihanul Islam, B Aditya Prakash, and Naren Ramakrishnan. Signet: Scalable embeddings for signed networks. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 157–169. Springer, 2018. (cited on page 69.)

[IR18] Patrick Hosein Inzamam Rahaman. A method for learning representations of signed networks. In *Proceedings of the International Workshop on Mining and Learning on Graphs (MLG2018)*, 2018. (cited on page 73.)

[JE10] Mohsen Jamali and Martin Ester. A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 135–142, 2010. (cited on page 1.)

[JKB15] Yong-Yeon Jo, Sang-Wook Kim, and Duck-Ho Bae. Efficient sparse matrix multiplication on gpu for large social network analysis. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1261–1270, 2015. (cited on page 55.)

[Joa02] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142. ACM, 2002. (cited 2 times on pages 3 and 44.)

[JS18] Haofeng Jia and Erik Saule. Graph embedding for citation recommendation. *arXiv preprint arXiv:1812.03835*, 2018. (cited on page 80.)

[K+12] Jinyoung Kim et al. *Retrieval and evaluation techniques for personal information*. Citeseer, 2012. (cited on page 44.)

[KB14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. (cited 4 times on pages 37, 57, 74, and 87.)

[KGC18] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7482–7491, 2018. (cited on page 81.)

[KK17] Nobuhiro Kaji and Hayato Kobayashi. Incremental skip-gram model with negative sampling. *arXiv preprint arXiv:1704.03956*, 2017. (cited on page 12.)

[KKT03] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146, 2003. (cited 3 times on pages 1, 22, and 33.)

[KL51] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951. (cited on page 47.)

[KNT10] Ravi Kumar, Jasmine Novak, and Andrew Tomkins. Structure and evolution of online social networks. In *Link mining: models, algorithms, and applications*, pages 337–357. Springer, 2010. (cited on page 43.)

[KO11] Bekir Karlik and A Vehbi Olgac. Performance analysis of various activation functions in generalized mlp architectures of neural networks. *International Journal of Artificial Intelligence and Expert Systems*, 1(4):111–122, 2011. (cited on page 37.)

[KPLK18] Junghwan Kim, Haekyu Park, Ji-Eun Lee, and U Kang. Side: representation learning in signed directed networks. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 509–518. International World Wide Web Conferences Steering Committee, 2018. (cited 4 times on pages 16, 69, 73, and 74.)

[KPST11] U Kang, Spiros Papadimitriou, Jimeng Sun, and Hanghang Tong. Centralities in large networks: Algorithms and observations. In *Proceedings of the 2011 SIAM International Conference on Data Mining*, pages 119–130. SIAM, 2011. (cited on page 60.)

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in*

*neural information processing systems*, pages 1097–1105, 2012. (cited on page 8.)

[KSW04] Jon Kleinberg, Aleksandrs Slivkins, and Tom Wexler. Triangulation and embedding using small sets of beacons. In *45th Annual IEEE Symposium on Foundations of Computer Science*, pages 444–453. IEEE, 2004. (cited on page 55.)

[KW06] Gueorgi Kossinets and Duncan J Watts. Empirical analysis of an evolving social network. *science*, 311(5757):88–90, 2006. (cited on page 33.)

[KW16a] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. (cited 2 times on pages 13 and 89.)

[KW16b] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016. (cited on page 13.)

[KZY13] Xiangnan Kong, Jiawei Zhang, and Philip S Yu. Inferring anchor links across multiple heterogeneous social networks. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 179–188, 2013. (cited on page 22.)

[LAH07] Jure Leskovec, Lada A Adamic, and Bernardo A Huberman. The dynamics of viral marketing. *ACM Transactions on the Web (TWEB)*, 1(1):5–es, 2007. (cited on page 20.)

[LBKT08] Jure Leskovec, Lars Backstrom, Ravi Kumar, and Andrew Tomkins. Microscopic evolution of social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 462–470, 2008. (cited on page 43.)

[LDH+17] Jundong Li, Harsh Dani, Xia Hu, Jiliang Tang, Yi Chang, and Huan Liu. Attributed network embedding for learning in a dynamic environment. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 387–396, 2017. (cited on page 94.)

[LGD15] Omer Levy, Yoav Goldberg, and Ido Dagan. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225, 2015. (cited on page 12.)

[LHK10a] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. Predicting positive and negative links in online social networks. In *Proceedings of the 19th international conference on World wide web*, pages 641–650. ACM, 2010. (cited on page 67.)

[LHK10b] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. Signed networks in social media. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 1361–1370, 2010. (cited on page 16.)

[LHLH18] Ninghao Liu, Xiao Huang, Jundong Li, and Xia Hu. On interpretation of network embedding via taxonomy induction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1812–1820, 2018. (cited on page 93.)

[LHR+15] Wenzhao Lian, Ricardo Henao, Vinayak Rao, Joseph Lucas, and Lawrence Carin. A multitask point process predictive model. In *International Conference on Machine Learning*, pages 2030–2038, 2015. (cited on page 78.)

[LHT+12] Xingjie Liu, Qi He, Yuanyuan Tian, Wang-Chien Lee, John McPherson, and Jiawei Han. Event-based social networks: linking the online and offline social worlds. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1032–1040, 2012. (cited on page 35.)

[LHZC18] Lizi Liao, Xiangnan He, Hanwang Zhang, and Tat-Seng Chua. Attributed social network embedding. *IEEE Transactions on Knowledge and Data Engineering*, 30(12):2257–2270, 2018. (cited on page 1.)

[LJSP18] Jiongqian Liang, Peter Jacobs, Jiankai Sun, and Srinivasan Parthasarathy. Semi-supervised embedding in attributed networks with outliers. In *Proceedings of the 2018 SIAM International Conference on Data Mining*, pages 153–161. SIAM, 2018. (cited 2 times on pages 1 and 21.)

[LM12] Jure Leskovec and Julian J Mcauley. Learning to discover social circles in ego networks. In *Advances in neural information processing systems*, pages 539–547, 2012. (cited 2 times on pages 29 and 58.)

[LM14] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196, 2014. (cited 3 times on pages 25, 28, and 37.)

[LMWDO10] Giseli Rabello Lopes, Mirella M Moro, Leandro Krug Wives, and José Palazzo Moreira De Oliveira. Collaboration recommendation on academic social networks. In *International conference on conceptual modeling*, pages 190–199. Springer, 2010. (cited on page 1.)

[LMY+12] Linyuan Lü, Matúš Medo, Chi Ho Yeung, Yi-Cheng Zhang, Zi-Ke Zhang, and Tao Zhou. Recommender systems. *Physics reports*, 519(1):1–49, 2012. (cited on page 7.)

[LNK07] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007. (cited 5 times on pages 1, 2, 7, 20, and 54.)

[Loh11] Wei-Yin Loh. Classification and regression trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1):14–23, 2011. (cited on page 36.)

[LSSS14] Roi Livni, Shai Shalev-Shwartz, and Ohad Shamir. On the computational efficiency of training neural networks. In *Advances in neural information processing systems*, pages 855–863, 2014. (cited on page 58.)

[LWB18] Po-Ming Law, Yanhong Wu, and Rahul C Basole. Segue: Overviewing evolution patterns of egocentric networks by interactive construction of spatial layouts. In *2018 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 72–83. IEEE, 2018. (cited on page 24.)

[LYL+17] Chao Lan, Yuhao Yang, Xiaoli Li, Bo Luo, and Jun Huan. Learning social circles in ego-networks based on multi-view network structure. *IEEE Transactions on Knowledge and Data Engineering*, 29(8):1681–1694, 2017. (cited on page 25.)

[LZT15] Yuchen Li, Dongxiang Zhang, and Kian-Lee Tan. Real-time targeted influence maximization for online advertisements. 2015. (cited on page 20.)

[LZZ+17] Zemin Liu, Vincent W Zheng, Zhou Zhao, Fanwei Zhu, Kevin Chen-Chuan Chang, Minghui Wu, and Jing Ying. Semantic proximity search on heterogeneous graph by proximity embedding. In *AAAI*, pages 154–160, 2017. (cited 4 times on pages 1, 16, 20, and 67.)

[LZZ+18] Zemin Liu, Vincent W Zheng, Zhou Zhao, Fanwei Zhu, Kevin Chen-Chuan Chang, Minghui Wu, and Jing Ying. Distance-aware dag embedding for proximity search on heterogeneous graphs. In *AAAI*, pages 2355–2362, 2018. (cited 2 times on pages 1 and 20.)

[Maa11] Avi Maayan. Introduction to network analysis in systems biology. *Science signaling*, 4(190):tr5–tr5, 2011. (cited on page 7.)

[MAB+10] Grzegorz Malewicz, Matthew H Austern, Aart JC Bik, James C Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski. Pregel: a system for large-scale graph processing. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 135–146, 2010. (cited on page 7.)

[MB09] Pinaki Mitra and Kamal Baid. Targeted advertising for online social networks. In *2009 First International Conference on Networked Digital Technologies*, pages 366–372. IEEE, 2009. (cited on page 1.)

[MBF+19] Carolyn Beth McNabb, Laura Grace Burgess, Amy Fancourt, Nancy Mulligan, Lily FitzGibbon, Patricia Riddell, and Kou Murayama. Similarity in resting-state functional brain connectivity is not influenced by social closeness in schoolchildren. *bioRxiv*, page 788208, 2019. (cited on page 54.)

[MBM+17] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5115–5124, 2017. No citation in the text.

[MBZ19] Lin Meng, Jiyang Bai, and Jiawei Zhang. Latte: Application oriented social network embedding. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 1169–1174. IEEE, 2019. (cited on page 22.)

[MC84] Peter V Marsden and Karen E Campbell. Measuring tie strength. *Social forces*, 63(2):482–501, 1984. (cited on page 24.)

[MC13] Silviu Maniu and Bogdan Cautis. Network-aware search in social tagging applications: instance optimality versus efficiency. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 939–948, 2013. (cited on page 54.)

[MCP+13] Animesh Mukherjee, Monojit Choudhury, Fernando Peruani, Niloy Ganguly, and Bivas Mitra. Dynamics on and of complex networks, volume 2. *AMC*, 10:12, 2013. (cited on page 34.)

[MdLFH19] João Moreira, André Carlos Ponce de Leon Ferreira, and Tomáš Horváth. *A General Introduction to Data Analytics*. Wiley Online Library, 2019. (cited on page 37.)

[MDM15] Axel Magnuson, Vijay Dialani, and Deepa Mallela. Event recommendation using twitter activity. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 331–332, 2015. (cited on page 35.)

[MF93] Peter V Marsden and Noah E Friedkin. Network studies of social influence. *Sociological Methods & Research*, 22(1):127–151, 1993. (cited on page 1.)

[MF94] Peter V Marsden and Noah E Friedkin. Network studies of social influence. *Sage focus editions*, 171:3–3, 1994. (cited on page 1.)

[MGHR] Ayush Maheshwari, Ayush Goyal, Manjesh Kumar Hanawal, and Ganesh Ramakrishnan. Dyngan: Generative adversarial networks for dynamic network embedding. (cited on page 14.)

[MK13] Andriy Mnih and Koray Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in neural information processing systems*, pages 2265–2273, 2013. (cited on page 17.)

[ML10] Rohan Miller and Natalie Lammas. Social media and its implications for viral marketing. *Asia Pacific Public Relations Journal*, 11(1):1–9, 2010. (cited on page 1.)

[ML14] Julian McAuley and Jure Leskovec. Discovering social circles in ego networks. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 8(1):4, 2014. (cited 8 times on pages 2, 24, 25, 26, 30, 31, 43, and 48.)

[MLBZ19] Zaiqiao Meng, Shangsong Liang, Hongyan Bao, and Xiangliang Zhang. Co-embedding attributed networks. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 393–401, 2019. (cited 6 times on pages 20, 79, 81, 86, 87, and 90.)

[MLTB93] George A Miller, Claudia Leacock, Randee Tengi, and Ross T Bunker. A semantic concordance. In *Proceedings of the workshop on Human Language Technology*, pages 303–308. Association for Computational Linguistics, 1993. (cited on page 45.)

[MM+17] Baruch Epstein Meir, Tomer Michaeli, et al. Joint autoencoders: a flexible multi-task learning framework. *arXiv preprint arXiv:1705.10494*, 2017. (cited on page 80.)

[MMG+07] Alan Mislove, Massimiliano Marcon, Krishna P Gummadi, Peter Druschel, and Bobby Bhattacharjee. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 29–42. ACM, 2007. (cited on page 59.)

[MMLDB20] Alexandru Mara, Yoosof Mashayekhi, Jefrey Lijffijt, and Tijl De Bie. Csne: Conditional signed network embedding. *arXiv preprint arXiv:2005.10701*, 2020. (cited on page 74.)

[Moo59] Edward F Moore. The shortest path through a maze. In *Proc. Int. Symp. Switching Theory, 1959*, pages 285–292, 1959. (cited on page 54.)

[MRV16] Fragkiskos D Malliaros, Maria-Evgenia G Rossi, and Michalis Vazirgiannis. Locating influential nodes in complex networks. *Scientific reports*, 6:19307, 2016. (cited on page 7.)

[MSC+13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013. (cited 13 times on pages 3, 5, 10, 11, 12, 25, 27, 33, 65, 69, 71, 73, and 75.)

[MSLH15] Claudio Martella, Roman Shaposhnik, Dionysios Logothetis, and Steve Harenberg. *Practical graph analytics with apache giraph*, volume 1. Springer, 2015. (cited on page 7.)

[MT17] Mamoru Mimura and Hidema Tanaka. Heavy log reader: learning the context of cyber attacks automatically with paragraph vector. In *International Conference on Information Systems Security*, pages 146–163. Springer, 2017. (cited on page 25.)

[MTT+16] Eric Malmi, Pyry Takala, Hannu Toivonen, Tapani Raiko, and Aristides Gionis. Dopelearning: A computational approach to rap lyrics generation. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 195–204. ACM, 2016. (cited on page 44.)

[MVL15] Syed Agha Muhammad and Kristof Van Laerhoven. Duke: A solution for discovering neighborhood patterns in ego networks. In *Ninth International AAAI Conference on Web and Social Media*, 2015. (cited 2 times on pages 2 and 24.)

[MVW14] Winter Mason, Jennifer Wortman Vaughan, and Hanna Wallach. Computational social science and social computing. 2014. (cited on page 7.)

[ND14] Nagarajan Natarajan and Inderjit S Dhillon. Inductive matrix completion for predicting gene–disease associations. *Bioinformatics*, 30(12):i60–i68, 2014. (cited on page 18.)

[New10] Mark Newman. *Networks: An Introduction*. Oxford University Press, 2010. (cited on page 43.)

[NH10] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010. (cited on page 37.)

[NJ] Jennifer Neville and David Jensen. Iterative classification in relational data. (cited on page 7.)

[NK17] Maximillian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. In *Advances in neural information processing systems*, pages 6338–6347, 2017. (cited 7 times on pages 33, 34, 37, 39, 45, 55, and 59.)

[NLR+18] Giang Hoang Nguyen, John Boaz Lee, Ryan A Rossi, Nesreen K Ahmed, Eunyee Koh, and Sungchul Kim. Continuous-time dynamic network embeddings. In *Companion Proceedings of the The Web Conference 2018*, pages 969–976, 2018. (cited on page 12.)

[NMV17] Giannis Nikolentzos, Polykarpos Meladianos, and Michalis Vazirgiannis. Matching node embeddings for graph similarity. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017. (cited on page 80.)

[NSJ12] Seema Nagar, Aaditeshwar Seth, and Anupam Joshi. Characterization of social media response to natural disasters. In *Proceedings of the 21st International Conference on World Wide Web*, pages 671–674, 2012. (cited on page 1.)

[OCP+16] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1105–1114, 2016. (cited 2 times on pages 1 and 7.)

[OE12] Fatih Ozgul and Zeki Erdem. Detecting criminal networks using social similarity. In *2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 581–585. IEEE, 2012. (cited on page 1.)

[PA06] Mihai Preda and Serge Abiteboul. Ranking nodes in a graph, July 11 2006. US Patent 7,076,483. (cited on page 2.)

[PARS14] Bryan Perozzi, Rami Al-Rfou, and Steven 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. (cited 12 times on pages ix, 7, 8, 10, 15, 21, 25, 30, 45, 68, 70, and 87.)

[PBCG09] Michalis Potamias, Francesco Bonchi, Carlos Castillo, and Aristides Gionis. Fast shortest path distance estimation in large networks. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 867–876. ACM, 2009. (cited 3 times on pages 55, 56, and 57.)

[PKS16] Bryan Perozzi, Vivek Kulkarni, and Steven Skiena. Walklets: Multiscale graph embeddings for interpretable network classification. *arXiv preprint arXiv:1605.02115*, 2016. (cited on page 12.)

[PLL+18] Zhen Peng, Minnan Luo, Jundong Li, Huan Liu, and Qinghua Zheng. Anomalous: A joint modeling approach for anomaly detection on attributed networks. In *IJCAI*, pages 3513–3519, 2018. (cited 2 times on pages 1 and 21.)

[PPK15] Georgios Petkos, Symeon Papadopoulos, and Yiannis Kompatsiaris. Social circle discovery in ego-networks by mining the latent structure of user connections and profile attributes. In *2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 880–887. IEEE, 2015. (cited on page 27.)

[PSV01] Romualdo Pastor-Satorras and Alessandro Vespignani. Epidemic spreading in scale-free networks. *Physical review letters*, 86(14):3200, 2001. (cited on page 1.)

[PWX16] Sancheng Peng, Guojun Wang, and Dongqing Xie. Social influence analysis in social networking big data: Opportunities and challenges. *IEEE network*, 31(1):11–17, 2016. (cited on page 20.)

[PWZ+16] Shirui Pan, Jia Wu, Xingquan Zhu, Chengqi Zhang, and Yang Wang. Tri-party deep network representation. *Network*, 11(9):12, 2016. (cited on page 12.)

[QCCY12] Miao Qiao, Hong Cheng, Lijun Chang, and Jeffrey Xu Yu. Approximate shortest distance computing: A query-dependent local landmark scheme. *IEEE Transactions on Knowledge and Data Engineering*, 26(1):55–68, 2012. (cited 2 times on pages 55 and 56.)

[QFZM13] Xueming Qian, He Feng, Guoshuai Zhao, and Tao Mei. Personalized recommendation combining user interest and social circle. *IEEE transactions on knowledge and data engineering*, 26(7):1763–1777, 2013. (cited on page 1.)

[QHW+19] Zhenyu Qiu, Wenbin Hu, Jia Wu, ZhongZheng Tang, and Xiaohua Jia. Noise-resilient similarity preserving network embedding for social networks. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 3282–3288. AAAI Press, 2019. (cited on page 21.)

[QLM12] Hailong Qin, Ting Liu, and Yanjun Ma. Mining user's real social circle in microblog. In *2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 348–352. IEEE, 2012. (cited 2 times on pages 24 and 25.)

[QXSW13] Zichao Qi, Yanghua Xiao, Bin Shao, and Haixun Wang. Toward a distance oracle for billion-node graphs. *Proceedings of the VLDB Endowment*, 7(1):61–72, 2013. (cited on page 56.)

[RB03] Erzsébet Ravasz and Albert-László Barabási. Hierarchical organization in complex networks. *Physical Review E*, 67(2):026112, 2003. (cited on page 62.)

[RBS11] Manuel Gomez Rodriguez, David Balduzzi, and Bernhard Schölkopf. Uncovering the temporal dynamics of diffusion networks. *arXiv preprint arXiv:1105.0697*, 2011. No citation in the text.

[RG19] Fatemeh Salehi Rizi and Michael Granitzer. Predicting event attendance exploring social influence. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, pages 2131–2134. ACM, 2019. (cited on page 80.)

[RGZ17] Fatemeh Salehi Rizi, Michael Granitzer, and Konstantin Ziegler. Global and local feature learning for ego-network analysis. In *2017 28th International Workshop on Database and Expert Systems Applications (DEXA)*, pages 98–102. IEEE, 2017. (cited 8 times on pages 33, 34, 37, 45, 48, 50, 51, and 80.)

[Roj13] Raúl Rojas. *Neural networks: a systematic introduction*. Springer Science & Business Media, 2013. (cited on page 48.)

[RP11] Oliver Richters and Tiago P Peixoto. Trust transitivity in social networks. *PloS one*, 6(4), 2011. (cited on page 1.)

[RR08] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184, 2008. (cited on page 21.)

[RS00] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000. (cited on page 8.)

[ŘS10] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. http://is.muni.cz/publication/884893/en. (cited on page 30.)

[RS18] Benedek Rozemberczki and Rik Sarkar. Fast sequence-based embedding with diffusion graphs. In *International Workshop on Complex Networks*, pages 99–107. Springer, 2018. (cited on page 59.)

[RSG18] Fatemeh Salehi Rizi, Joerg Schloetterer, and Michael Granitzer. Shortest path distance approximation using deep learning techniques. In *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 1007–1014. IEEE, 2018. (cited on page 80.)

[RZA17] Ryan A Rossi, Rong Zhou, and Nesreen K Ahmed. Deep feature learning for graphs. *arXiv preprint arXiv:1704.08829*, 2017. (cited on page 19.)

[Sab66] Gert Sabidussi. The centrality index of a graph. *Psychometrika*, 31(4):581–603, 1966. (cited on page 54.)

[SB18] John Brennan Georgios Theodoropoulos Andrew Stephen McGough Boguslaw Obara Stephen Bonner, Ibad Kureshi. Exploring the semantic content of unsupervised graph embeddings: An empirical study. *arXiv preprint arXiv:1806.07464*, 2018. (cited on page 40.)

[SBM12] Richard Socher, Yoshua Bengio, and Christopher D Manning. Deep learning for nlp (without magic). In *Tutorial Abstracts of ACL 2012*, pages 5–5. Association for Computational Linguistics, 2012. (cited on page 8.)

[Sch07] Satu Elisa Schaeffer. Graph clustering. *Computer science review*, 1(1):27–64, 2007. (cited on page 7.)

[SD14] Kumar Sricharan and Kamalika Das. Localizing anomalous changes in time-evolving graphs. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 1347–1358, 2014. (cited on page 1.)

[SH12] Robert Speer and Catherine Havasi. Representing general relational knowledge in conceptnet 5. In *LREC*, pages 3679–3686, 2012. (cited on page 45.)

[SH13] Yizhou Sun and Jiawei Han. Mining heterogeneous information networks: a structural analysis approach. *Acm Sigkdd Explorations Newsletter*, 14(2):20–28, 2013. (cited 2 times on pages 14 and 67.)

[SHZP18] Chuan Shi, Binbin Hu, Wayne Xin Zhao, and S Yu Philip. Heterogeneous information network embedding for recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 31(2):357–370, 2018. (cited on page 67.)

[SKLD14] Anna Squicciarini, Sushama Karumanchi, Dan Lin, and Nicole Desisto. Identifying hidden social circles for advanced privacy configuration. *Computers & Security*, 41:40–51, 2014. (cited on page 32.)

[SLKD12] Anna Squicciarini, Dan Lin, Sushama Karumanchi, and Nicole DeSisto. Automatic social group organization and privacy management. In *8th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*, pages 89–96. IEEE, 2012. (cited on page 32.)

[SLZ+16] Chuan Shi, Yitong Li, Jiawei Zhang, Yizhou Sun, and S Yu Philip. A survey of heterogeneous information network analysis. *IEEE Transactions on Knowledge and Data Engineering*, 29(1):17–37, 2016. (cited on page 14.)

[SLZ+17] Chuan Shi, Yitong Li, Jiawei Zhang, Yizhou Sun, and S Yu Philip. A survey of heterogeneous information network analysis. *IEEE Transactions on Knowledge and Data Engineering*, 29(1):17–37, 2017. (cited on page 67.)

[SNB+08] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008. (cited on page 86.)

[SRG17] Fatemeh Salehi Rizi and Michael Granitzer. Properties of vector embeddings in social networks. *Algorithms*, 10(4):109, 2017. (cited 2 times on pages 55 and 80.)

[Ste74] Michael A Stephens. Edf statistics for goodness of fit and some comparisons. *Journal of the American statistical Association*, 69(347):730–737, 1974. (cited on page 49.)

[STLS06] Xiaodan Song, Belle L Tseng, Ching-Yung Lin, and Ming-Ting Sun. Personalized recommendation driven by information flow. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 509–516, 2006. (cited on page 20.)

[ŞUY+18] Lütfi Kerem Şenel, Ihsan Utlu, Veysel Yücesoy, Aykut Koc, and Tolga Cukur. Semantic structure and interpretability of word embeddings. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(10):1769–1779, 2018. (cited on page 45.)

[SWRG19] Jörg Schlötterer, Martin Wehking, Fatemeh Salehi Rizi, and Michael Granitzer. Investigating extensions to random walk based graph embedding. In *2019 IEEE International Conference on Cognitive Computing (ICCC)*, pages 81–89. IEEE, 2019. (cited on page 7.)

[SZ08] Aaditeshwar Seth and Jie Zhang. A social network based approach to personalized recommendation of participatory media content. In *ICWSM*, 2008. (cited on page 1.)

[SZG+18] Yu Shi, Qi Zhu, Fang Guo, Chao Zhang, and Jiawei Han. Easing embedding learning by comprehensive transcription of heterogeneous information networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2190–2199. ACM, 2018. (cited 2 times on pages 67 and 69.)

[TACGB+11] Konstantin Tretyakov, Abel Armas-Cervantes, Luciano García-Bañuelos, Jaak Vilo, and Marlon Dumas. Fast fully dynamic landmark-based estimation of shortest path distances in very large graphs. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 1785–1794, 2011. (cited 2 times on pages 56 and 57.)

[TBA06] Ioannis Tsamardinos, Laura E Brown, and Constantin F Aliferis. The max-min hill-climbing bayesian network structure learning algorithm. *Machine learning*, 65(1):31–78, 2006. (cited on page 27.)

[TCAL16] Jiliang Tang, Yi Chang, Charu Aggarwal, and Huan Liu. A survey of signed network mining in social media. *ACM Computing Surveys (CSUR)*, 49(3):1–37, 2016. (cited on page 16.)

[TFL+15] Yulia Tsvetkov, Manaal Faruqui, Wang Ling, Guillaume Lample, and Chris Dyer. Evaluation of word vector representations by subspace alignment. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2049–2054, 2015. (cited on page 45.)

[TG16] Shazia Tabassum and João Gama. Evolution analysis of call ego-networks. In *International conference on discovery science*, pages 213–225. Springer, 2016. (cited on page 24.)

[TGB11] Mohammad A Tayebi, Uwe Glässer, and Patricia L Brantingham. Organized crime detection in co-offending networks. In *IEEE-9th International Conference Proceedings*. Citeseer, 2011. (cited on page 7.)

[TGC+14] Fei Tian, Bin Gao, Qing Cui, Enhong Chen, and Tie-Yan Liu. Learning deep representations for graph clustering. In *Aaai*, volume 14, pages 1293–1299. Citeseer, 2014. (cited 2 times on pages 1 and 21.)

[TH12] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012. (cited on page 30.)

[TK14] Frank W Takes and Walter A Kosters. Adaptive landmark selection strategies for fast shortest path computation in large real-world graphs. In *2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, volume 1, pages 27–34. IEEE, 2014. (cited on page 55.)

[TKOT15] Yukihiro Tagami, Hayato Kobayashi, Shingo Ono, and Akira Tajima. Modeling user activities on the web using paragraph vector. In *Proceedings of the 24th International Conference on World Wide Web*, pages 125–126, 2015. (cited on page 25.)

[TLLS17] Cunchao Tu, Han Liu, Zhiyuan Liu, and Maosong Sun. Cane: Context-aware network embedding for relation modeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1722–1731, 2017. (cited on page 20.)

[TLXW14] Yan Tang, Lili Lin, Zhuoming Xu, and Yu Wang. Effective social circle prediction based on bayesian network. In *2014 11th Web Information System and Application Conference*, pages 131–135. IEEE, 2014. (cited on page 27.)

[TMKM18] Anton Tsitsulin, Davide Mottin, Panagiotis Karras, and Emmanuel Müller. Verse: Versatile graph embeddings from similarity measures. In *Proceedings of the 2018 World Wide Web Conference*, pages 539–548. International World Wide Web Conferences Steering Committee, 2018. (cited on page 7.)

[TNJ16] Mengfan Tang, Feiping Nie, and Ramesh Jain. Capped lp-norm graph embedding for photo clustering. In *Proceedings of the 24th ACM international conference on Multimedia*, pages 431–435, 2016. (cited on page 1.)

[TNS18] Dinh V Tran, Nicolò Navarin, and Alessandro Sperduti. On filter size in graph convolutional networks. In *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1534–1541. IEEE, 2018. (cited on page 13.)

[TQM15] Jian Tang, Meng Qu, and Qiaozhu Mei. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1165–1174, 2015. (cited 3 times on pages 15, 67, and 68.)

[TQW+15] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 1067–1077, Republic and Canton of Geneva, Switzerland, 2015. International World Wide Web Conferences Steering Committee. (cited 14 times on pages 3, 6, 7, 10, 11, 15, 30, 45, 50, 68, 70, 71, 72, and 93.)

[Tra18] Phi Vu Tran. Multi-task graph autoencoders. *arXiv preprint arXiv:1811.02798*, 2018. (cited on page 84.)

[TSWY09] Jie Tang, Jimeng Sun, Chi Wang, and Zi Yang. Social influence analysis in large-scale networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 807–816. ACM, 2009. (cited on page 33.)

[TY12] Xuning Tang and Christopher C Yang. Ranking user influence in healthcare social media. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(4):1–21, 2012. (cited 2 times on pages 1 and 20.)

[TZY+08] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong 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, 2008. (cited on page 1.)

[UKBM11] Johan Ugander, Brian Karrer, Lars Backstrom, and Cameron Marlow. The anatomy of the facebook social graph. *arXiv preprint arXiv:1111.4503*, 2011. (cited on page 24.)

[Uts15] Akira Utsumi. A complex network approach to distributional semantic models. *PloS one*, 10(8), 2015. (cited on page 7.)

[VA13] G Tendayi Viki and Dominic Abrams. The social influence of groups on individuals. 2013. (cited on page 38.)

[VL07] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007. (cited on page 22.)

[VSKB10] S Vichy N Vishwanathan, Nicol N Schraudolph, Risi Kondor, and Karsten M Borgwardt. Graph kernels. *Journal of Machine Learning Research*, 11(Apr):1201–1242, 2010. (cited on page 1.)

[WATL17] Suhang Wang, Charu Aggarwal, Jiliang Tang, and Huan Liu. Attributed signed network embedding. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 137–146, 2017. (cited 2 times on pages 18 and 74.)

[WCW+17] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. Community preserving network embedding. In *Thirty-first AAAI conference on artificial intelligence*, 2017. (cited 2 times on pages 1 and 21.)

[WCWW15] Shitong Wang, Fu-Lai Chung, Jun Wang, and Jun Wu. A fast learning method for feedforward neural networks. *Neurocomputing*, 149:295–307, 2015. (cited on page 29.)

[WCZ16] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1225–1234, 2016. (cited 6 times on pages 1, 7, 9, 12, 45, and 67.)

[WG13] Yu Wang and Lin Gao. An edge-based clustering algorithm to detect social circles in ego networks. *JCP*, 8(10):2575–2582, 2013. (cited on page 25.)

[WLZ12] Jing Wang, Zhijing Liu, and Hui Zhao. Group recommendation based on the pagerank. *JNW*, 7(12):2019–2024, 2012. (cited on page 2.)

[WPLP14] Robert West, Hristo S Paskov, Jure Leskovec, and Christopher Potts. Exploiting social network structure for person-to-person sentiment analysis. *arXiv preprint arXiv:1409.2450*, 2014. (cited on page 73.)

[WPZ+15] Yanhong Wu, Naveen Pitipornvivat, Jian Zhao, Sixiao Yang, Guowei Huang, and Huamin Qu. egoslider: Visual analysis of egocentric network evolution. *IEEE transactions on visualization and computer graphics*, 22(1):260–269, 2015. (cited on page 24.)

[WSP07] Chao Wang, Venu Satuluri, and Srinivasan Parthasarathy. Local probabilistic models for link prediction. In *Seventh IEEE international conference on data mining (ICDM 2007)*, pages 322–331. IEEE, 2007. (cited on page 1.)

[WTA+17] Suhang Wang, Jiliang Tang, Charu Aggarwal, Yi Chang, and Huan Liu. Signed network embedding in social media. In *Proceedings of the 2017 SIAM international conference on data mining*, pages 327–335. SIAM, 2017. (cited 4 times on pages 17, 69, 73, and 74.)

[WWH+14] Ji Wan, Dayong Wang, Steven Chu Hong Hoi, Pengcheng Wu, Jianke Zhu, Yongdong Zhang, and Jintao Li. Deep learning for content-based image retrieval: A comprehensive study. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 157–166, 2014. (cited on page 8.)

[WWW+18] Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. Graphgan: Graph representation learning with generative adversarial nets. In *Thirty-second AAAI conference on artificial intelligence*, 2018. (cited on page 14.)

[WXCY17] Xiaokai Wei, Linchuan Xu, Bokai Cao, and Philip S Yu. Cross view link prediction by learning noise-resilient representation consensus. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1611–1619, 2017. (cited on page 1.)

[WZH+18] Hongwei Wang, Fuzheng Zhang, Min Hou, Xing Xie, Minyi Guo, and Qi Liu. Shine: signed heterogeneous information network embedding for sentiment link prediction. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 592–600. ACM, 2018. (cited 6 times on pages 67, 72, 73, 74, 75, and 78.)

[XGFS13] Reynold S Xin, Joseph E Gonzalez, Michael J Franklin, and Ion Stoica. Graphx: A resilient distributed graph system on spark. In *First international workshop on graph data management experiences and systems*, pages 1–6, 2013. (cited on page 7.)

[YBLS08] Sihem Amer Yahia, Michael Benedikt, Laks VS Lakshmanan, and Julia Stoyanovich. Efficient network aware search in collaborative tagging sites. *Proceedings of the VLDB Endowment*, 1(1):710–721, 2008. (cited on page 54.)

[YCA+18] Wenchao Yu, Wei Cheng, Charu C Aggarwal, Kai Zhang, Haifeng Chen, and Wei Wang. Netwalk: A flexible deep embedding approach for anomaly detection in dynamic networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2672–2681, 2018. (cited on page 1.)

[YDCL06] Wan-Shiou Yang, Jia-Ben Dia, Hung-Chi Cheng, and Hsing-Tzu Lin. Mining social networks for targeted advertising. In *Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06)*, volume 6, pages 137a–137a. IEEE, 2006. (cited on page 1.)

[YL15] Jaewon Yang and Jure Leskovec. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems*, 42(1):181–213, 2015. (cited on page 59.)

[YLL+14] Yuhao Yang, Chao Lan, Xiaoli Li, Bo Luo, and Jun Huan. Automatic social circle detection using multi-view clustering. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1019–1028, 2014. (cited on page 26.)

[YLZ+15] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y Chang. Network representation learning with rich text information. In *IJCAI*, volume 2015, pages 2111–2117, 2015. (cited 2 times on pages 1 and 18.)

[YPZ+18] Hong Yang, Shirui Pan, Peng Zhang, Ling Chen, Defu Lian, and Chengqi Zhang. Binarized attributed network embedding. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 1476–1481. IEEE, 2018. (cited 3 times on pages 79, 81, and 86.)

[YWX17] Shuhan Yuan, Xintao Wu, and Yang Xiang. Sne: signed network embedding. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 183–195. Springer, 2017. (cited 2 times on pages 17 and 74.)

[Zac77] Wayne W Zachary. An information flow model for conflict and fission in small groups. *Journal of anthropological research*, 33(4):452–473, 1977. (cited 2 times on pages ix and 8.)

[ZAH17] Baichuan Zhang and Mohammad Al Hasan. Name disambiguation in anonymized graphs using network embedding. In *Proceedings of the*

*2017 ACM on Conference on Information and Knowledge Management*, pages 1239–1248, 2017. (cited on page 67.)

[ZAL14] Reza Zafarani, Mohammad Ali Abbasi, and Huan Liu. *Social media mining: an introduction*. Cambridge University Press, 2014. (cited on page 46.)

[ZGL03] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pages 912–919, 2003. (cited on page 7.)

[Zho11] T Zhou. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications*, 390(6):1150–1170, 2011. (cited on page 20.)

[Zho19] Sida Zhou. Empirical effect of graph embeddings on fraud detection/risk mitigation. *arXiv preprint arXiv:1903.05976*, 2019. (cited on page 80.)

[ZL09] R. Zafarani and H. Liu. Social computing data repository at ASU, 2009. (cited on page 58.)

[ZL17] Shuo Zhang and Qin Lv. Event organization 101: Understanding latent factors of event popularity. In *Eleventh International AAAI Conference on Web and Social Media*, 2017. (cited 2 times on pages 33 and 36.)

[ZL19] Jason Shuo Zhang and Qin Lv. Understanding event organization at scale in event-based social networks. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):16, 2019. (cited on page 33.)

[ZLG+09] Zhongzhi Zhang, Yuan Lin, Shuyang Gao, Shuigeng Zhou, and Jihong Guan. Average distance in a hierarchical scale-free network: an exact solution. *Journal of Statistical Mechanics: Theory and Experiment*, 2009(10):P10022, 2009. (cited on page 62.)

[ZLJ+15] Fuzhen Zhuang, Dan Luo, Xin Jin, Hui Xiong, Ping Luo, and Qing He. Representation learning via semi-supervised autoencoder for multi-task learning. In *2015 IEEE International Conference on Data Mining*, pages 1141–1146. IEEE, 2015. (cited on page 80.)

[ZLL+17] Chang Zhou, Yuqiong Liu, Xiaofei Liu, Zhongyi Liu, and Jun Gao. Scalable graph embedding for asymmetric proximity. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 2942–2948, 2017. (cited 2 times on pages 1 and 20.)

[ZN13] Jiangchuan Zheng and Lionel M Ni. An unsupervised learning approach to social circles detection in ego bluetooth proximity network. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, pages 721–724. ACM, 2013. (cited on page 25.)

[ZSG16] Stefan Zwicklbauer, Christin Seifert, and Michael Granitzer. Robust and collective entity disambiguation through semantic embeddings. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 425–434. ACM, 2016. (cited on page 67.)

[ZSW+10] Xiaohan Zhao, Alessandra Sala, Christo Wilson, Haitao Zheng, and Ben Y Zhao. Orion: shortest path estimation for large social graphs. *networks*, 1:5, 2010. (cited 4 times on pages 55, 56, 59, and 63.)

[ZSZZ11] Xiaohan Zhao, Alessandra Sala, Haitao Zheng, and Ben Y Zhao. Efficient shortest paths on massive social graphs. In *7th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*, pages 77–86. IEEE, 2011. (cited 4 times on pages 55, 56, 59, and 63.)

[ZVTD+18] Guido Zampieri, Dinh Van Tran, Michele Donini, Nicolo Navarin, Fabio Aiolli, Alessandro Sperduti, and Giorgio Valle. Scuba: scalable kernel-based gene prioritization. *BMC bioinformatics*, 19(1):23, 2018. (cited on page 7.)

[ZXZZ15] Qian-Ming Zhang, Xiao-Ke Xu, Yu-Xiao Zhu, and Tao Zhou. Measuring multiple evolution mechanisms of complex networks. *Scientific reports*, 5:10350, 2015. (cited on page 21.)

[ZYZZ17] Daokun Zhang, Jie Yin, Xingquan Zhu, and Chengqi Zhang. User profile preserving social network embedding. In *IJCAI International Joint Conference on Artificial Intelligence*, 2017. (cited on page 21.)

[ZYZZ19] Daokun Zhang, Jie Yin, Xingquan Zhu, and Chengqi Zhang. Attributed network embedding via subspace discovery. *arXiv preprint arXiv:1901.04095*, 2019. (cited 3 times on pages 79, 86, and 87.)

[ZZC15] Xiaomei Zhang, Jing Zhao, and Guohong Cao. Who will attend?–predicting event attendance in event-based social network. In *2015 16th IEEE International Conference on Mobile Data Management*, volume 1, pages 74–83. IEEE, 2015. (cited on page 35.)