UNIVERSITY OF SOUTHAMPTON

Faculty of Physical Sciences and Engineering

School of Electronics and Computer Science

GPSP: Graph Partition and Space Projection based approach for Heterogeneous Network Embedding

by

Wenyu DU

Completed at Sep 2017

Supervisor: **Dr. Markus Brede**

Second Examiner: **Dr. Sarvapali (Gopal) Ramchurn**

A dissertation submitted in partial fulfillment of the degree of
MSc Data Science.

Abstract

We study the problem of embedding different types of vertices in heterogeneous networks. Previous network embedding approaches mostly focus on homogeneous networks and only a few heterogeneous network embedding methods still learn the embeddings of different types of nodes in the same latent space. We develop one scalable heterogeneous network embedding model, namely GPSP. The GPSP model uses edge-type based Graph Partition to divide the heterogeneous network into two types of subnetworks, homogeneous networks and bipartite networks. Next homogeneous subnetworks will be fed into conventional homogeneous network embedding models in separate spaces. Then the projective relation hidden in bipartite subnetworks will be extracted to learn projective embeddings in the Space Projection step. Finally, GPSP combines homogeneous and projective embeddings to generate the complete heterogeneous embeddings. Experiments show that GPSP model outperforms all state-of-the-art baseline embedding models in various network mining tasks, like classification and clustering for vertices in the heterogeneous network.

Acknowledgements

I feel the need to express my gratitude to my supervisor Dr. Markus Brede for offering me the chance to work on such a novel topic. During our meeting, he had many nutritious talks for the project and beyond. I still remembered he pointed out it was crucial to learn the knowledge thoroughly for doing research. Superficial comprehension might be okay for taught modules but not enough for doing any research, as I summarized after the meeting.

In addition, I need to thank Dr. Tao Wang who had the project idea and introduced me the field of Network Embedding. During our extensive conversations, we came up many basic ideas for the paper, like Graph Partition. And he has shown himself as an excellent example of being a true young academic.

Also, I want to thank the second examiner of the project Dr. Sarvapali (Gopal) Ramchurn for motivation and appreciation he gave me in the meeting.

Last but not least, I want to thank my parents and all my friends for their understanding and supporting.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

Human social activities, academic activities and biology networks, are examples of real-world complex systems, wherein enormous multi-typed components interact with each other [11]. In such systems, these interacting components compose the interconnected networks, named as **information networks** [30]. From Facebook network to Wikipedia network, it is evident that the information network is ubiquitous and takes an important part in modern information infrastructure [29]. Figure 1.1 shows one example of information networks.

In the preceding decades, analysis of these information networks attracts attentions from a wide range of disciplines, such as Physics, Computer Science, Social Science, and so on. Particularly, in the field of data mining and information retrieval, information network analysis becomes one of the main research topics. The basic paradigm is to mine implicit information from interaction links in these information networks, and related work covers scopes from link mining and analysis [9, 15, 42], graph mining [40] to social network analysis [26, 41] and network science [18].

However, most outputs of traditional information network analysis cannot be directly fed into multiple downstream machine learning algorithms, a cumbersome pre-processing step is required. Inspired by recent advances of word embedding in the neutral language processing (NLP) field [3], i.e. word2vec embedding model [22, 23]. **Information net-**

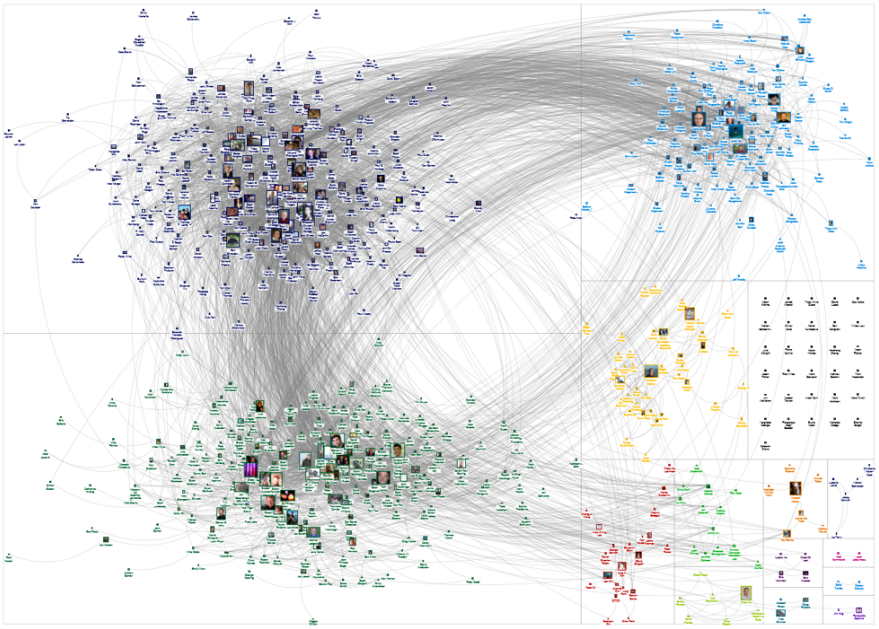


Figure 1.1: One example of information networks: Facebook friendship network

work embedding (**network embedding** for short) or called **information network latent representation learning** (**network representation learning** for short) was formally introduced to learn general hidden pattern from large-scale information networks containing millions of **vertices** (**nodes**) and **edges** (**links**). The learned outputs of network embedding are usually vectorized **representations** (**embeddings**) for all vertices in these networks.

The output then can be used straightforwardly for various data mining applications such as node classification [16, 28], clustering [32], recommendation [38], network visualization [20], missing link prediction [19] and anomaly detection [5]. In stead of building handcrafted feature vectors, network embedding models such as DeepWalk [27], LINE [34] and node2vec [10] can learn meaningful latent vector representations from network data directly.

1.2 Motivation

Most of the network embedding methods are built upon an underlying hypothesis: there is only one type of nodes and links in the network. That is, the network is homogeneous. Like a large-scale friendship network [17], the kind of millions of objects can only be

human, and tens of millions of links can only represent friendship relation.

However, in practical cases, information networks usually exist in a heterogeneous fashion; there are multiple types of vertices and edges in an information network. Figure 1.2 [14] shows an example that demonstrates a heterogeneous academic network containing two distinct types of vertices, author and paper, and three kinds of links between objects. A connection between two papers should represent a citation relationship while a link between an author and a paper should mean the author writes the paper.

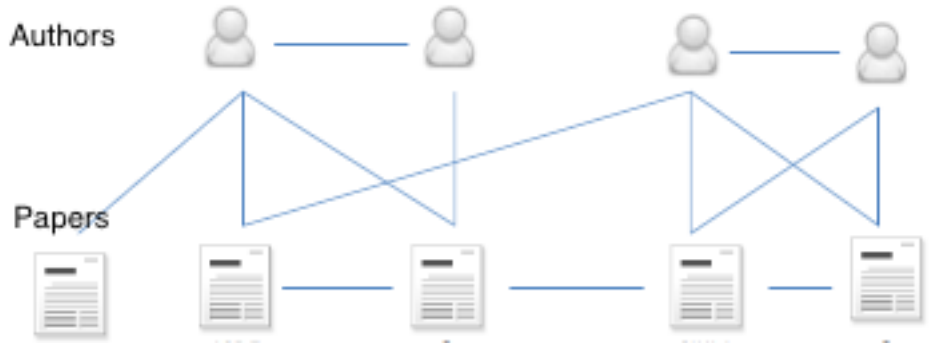


Figure 1.2: One example of heterogeneous information network: academic network[14]

Compared to homogeneous network embedding, heterogeneous network embedding requires the algorithms to take objects and links' type information into account. Since the concept of meta path was proposed in 2011 [31], it became the dominant methodology in the heterogeneous network analysis field. And in only a few existing heterogeneous network embedding algorithms [7, 12], most of them are designed based on the meta path concept. However, the meta path based heterogeneous network embedding algorithms still focus on learning representations for different types of vertices in the same latent space. Apparently, different types of vertices and their links have entirely different meanings.

1.3 Aims and Objectives

So, for a heterogeneous network, can we try to learn representations separately concerning vertices' types? Can we apply homogeneous network-oriented embedding algorithms like LINE or DeepWalk to heterogeneous networks with simple modification?

Aiming to solve the above questions, the project explores the potential solutions to parti-

tion heterogeneous networks into subnetworks, perform network embedding on homogeneous subnetworks, analyze relations between subnetworks and combine their outputs as the final output. Moreover, the model should also be compared with previous network embedding models for various network mining tasks, such as node classification, clustering, and visualization.

Based on the aims above, the objectives can be summarized as:

- Familiar with previous successful network embedding models by running or building them.
- Familiar with heterogeneous information networks and related work.
- Implement the main network embedding model for heterogeneous networks.
- Compare the performance of the model with other previous models by using models' output for different network mining tasks.
- Evaluate the experiment and sum up as a conclusion.

1.4 Contribution

This project formalizes the edge-based Graph Partition problem, where heterogeneous network will be divided into homogeneous and bipartite networks. We also introduce the latent Space Projection concept that a vertex can learn different representation vectors in different spaces. Building on the above ideas, GPSP, a heterogeneous network embedding model is presented. The goal of GPSP is to learn a vertex' representations from different perspectives (spaces). We first perform edge-based Graph Partition to get homogeneous and bipartite subnetworks. Then we apply conventional embedding algorithms on homogeneous subnetworks and use Space Projection techniques to learn projective presentations for vertices. The homogeneous and projective presentations will be concatenated as the final heterogeneous embedding.

The proposed GPSP is different from previous network embedding models, which mainly focus on homogeneous network [34, 27, 10]. Specifically, the ignorance of type information

of vertices and edges in training causes the conventional embedding models to produce type indistinguishable embeddings for heterogeneous vertices. The model differs from PTE [33] and metapath2vec [7] as well. PTE is a semi-supervised heterogeneous text network embedding model, but GPSP and metapath2vec are unsupervised heterogeneous network embedding models. However, metapath2vec and PTE still project different types of vertices into the same embedding space while GPSP solves this problem by Space Projection.

To summarize, our work makes the following contribution:

1. Formalizes the edge-based Graph Partition problem and identifies its relation to different types of networks.

2. Introduces the Space Projection concept into the network embedding domain and formalizes the projection process.

3. Develops GPSP, a novel heterogeneous network embedding model to preserve both structural and semantic (type) information in the heterogeneous network.

4. Through experiments, demonstrates the efficacy of GPSP in various network mining tasks such as node classification and clustering.

Chapter 2

Related Work

2.1 History of Network Embedding

2.1.1 1977 - 2013

Although network embedding only witnesses gradual popularity in the recent five years, its pioneer research can date back to 1977 [8] when graph embedding problem was first introduced. In the first few years of the new millennium, a family of new general graph embedding algorithms was developed [25, 37]. These algorithms aim to construct low dimensional manifolds to model the nonlinear geometric data, such as Laplacian Eigenmaps [2], Isomap [37] and spectral based approaches [6, 39].

Up till 2013, a large number of network embedding methods had been proposed. Iwata et al. [13] performed probabilistic latent semantic analysis on document networks. Temporal information was employed by Tang et al. [36] for analyzing dynamic multi-mode networks. Mei et al. [21] implemented a harmonic regularization based embedding framework for topic modeling problem in the network structure.

However, like Laplacian Eigenmap [2] and MDS [4], embedding models mentioned above fail to handle a large scale network. The reason is that these algorithms focus on factorizing a network into a matrix or tensor format and generate latent features for vertices or edges in this matrix or tensor, but the computational cost of decomposing such a large scale matrix or tensor is usually too high. Moreover, this approach also suffers

from statistical performance drawback [10], making it impractical to embed large scale networks.

2.1.2 2013 - Now

With the prevalence of deep learning methods, researchers tried to design neural network based embedding algorithms. In 2013, a famous word embedding model word2vec [23, 22] was first proposed in the natural language processing (NLP) domain. It was designed as a two-layer neural network and to learn the distributed representations for words in a size-fixed shifting window. The goal of the framework is to optimize the distributed probabilities for all words in their "context". If one word appears in one window, the rest words in this window will be regarded as the "context" of this word. This maximizing probability task could be categorized as a maximum likelihood estimation problem.
Building on the concept of word2vec, DeepWalk [27] was introduced in the network embedding field. But unlike the word document, which can naturally provide sentences as paths for the window to slide, networked data do not have this property. So DeepWalk performs random walking within the network and record the walking paths as the "sentences" in which the window can slide across vertices to get "context".
In 2015, Tang et al. proposed a large-scale information network embedding method LINE [34], wherein they introduced the concept of the first order and the second order of proximity in the network. The model LINE can preserve these two orders of proximity in the learned representations.
More recently, Yang et al. proved DeepWalk is equivalent to a matrix factorization TADW [43]. Grover and Leskovec proposed node2vec [10], a biased random walk model, which is a mixture of breadth-first and width-first search approach. Followed the idea in LINE, Tang et al. implemented PTE [33], a semi-supervised model for embedding heterogeneous text network. In 2017, metapath2vec [7], a meta path based heterogeneous network embedding model was introduced.

2.2 Introduction of Network Embedding Models

This section will introduce four successful network embedding models in detail, two homogeneous network embedding models, DeepWalk and LINE, and two heterogeneous network embedding ones, PTE and methpath2vec.
DeepWalk and LINE will be adopted in the project while DeepWalk, LINE and a modified version of PTE will be used as the benchmarks in the paper.

2.2.1 Homogeneous Model 1: DeepWalk

This DeepWalk [27] algorithm contains two parts; first a random walk generator and second an update procedure, which will be introduced respectively.

Random walk generator

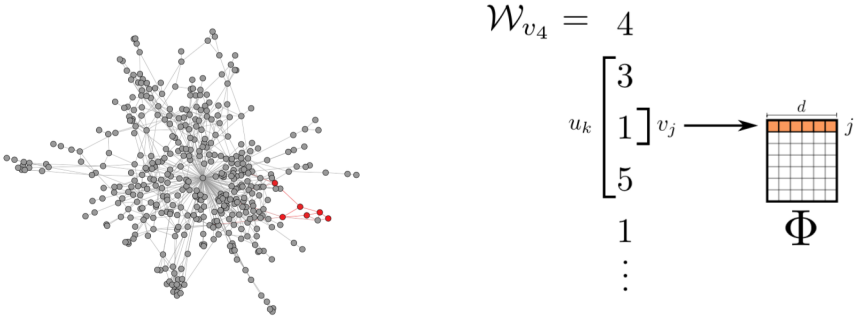
Starting from each vertex in the network, the walk generator will randomly visit one of the neighbors of the last visited vertex and iterate until reach a maximum walking length t. The neighborhood relation is defined by whether there is an edge between two vertices. Then for each vertex, the above random walking procedure will perform γ times to generate γ random walk paths, as shown in Figure 2.1 (a).

Representation mapping and Skip-Gram

Then every generated random walk path W_{v_i} will be applied to Skip-Gram, a language model introduced in word2vec [22], where the joint probabilities among the vertices that co-occur within a window will be maximized. In practice, when the window shifts across the path, each vertex v_j within the window will be mapped to its current representation vector $\Phi(v_j) \in R^d$ (See Figure 2.1 (b)). Then the mapped representation will be updated based on the "context" in the window.

Updating rule in Skip-Gram

Here is the updating rule: for every vertex v_i in the walk path W_{v_i}, its context is the neighbor vertices within the size of the slide window w, which is defined as $C =$



(a) Random walked paths

(b) Representation mapping: W_{v_4} is the random path, u_k is the window and v_j is the mapping vertex

Figure 2.1: Two procedures in DeepWalk [27]

$\{v_{i-w}, v_{i-w+1}, \ldots, v_{i+w-1}, v_{i+w}\}$. The goal of updating is to maximize the average log probability of all node pairs in W_{v_i}:

$$O = \frac{1}{|W_{v_i}|} \sum_{i=1}^{|W_{v_i}|} \sum_{-w \leq j \leq w, j \neq 0} \log p(v_{i+j}|v_i) \tag{2.1}$$

Posterior distribution p

In terms of probability $p(v_j|v_i)$, we can model it as the posterior distribution of logistic regression or softmax classifier [27]. Formula 2.2 defines a softmax classifier version,

$$p(v_j|v_i) = \frac{exp(c_{v_j}^T d_{v_i})}{\sum_{v \in V} exp(c_v^T d_{v_i})} \tag{2.2}$$

where vector c_v represents the contexts of vertex v while vector d_v represents vertex v. In the DeepWalk's paper, one type of softmax classifiers, Hierarchical Softmax [24] is used for speedup optimization.

2.2.2 Homogeneous Model 2: LINE

LINE [34], a large-scale network embedding model, is another homogeneous network embedding model, aiming to preserve both first and second orders of proximity in the networks. Here is the introduction of two orders of proximity.

First order proximity

First order proximity refers to the local pairwise proximity between the vertices in the network [34]. It describes a phenomenon that similar vertices are most likely to be linked directly in a network [1]. In Figure 2.2, vertex 6 and 7 are directly connected, according to first order proximity, these two vertices should have similar representations after performing network embedding.

Second order proximity

Second order proximity is based on another phenomenon observed in the network that "even two vertices are not directly linked but many neighbors of them are the same, so they two also should have similar vector representations after embedding". The relation between vertices 5 and 6 in Figure 2.2 is such an example. The second order proximity fills the blank that the first order proximity leaves, because first order proximity fails to represent global structure in the network but second order proximity can. Moreover, massive legitimate connections are not observed in a real-world network [34].

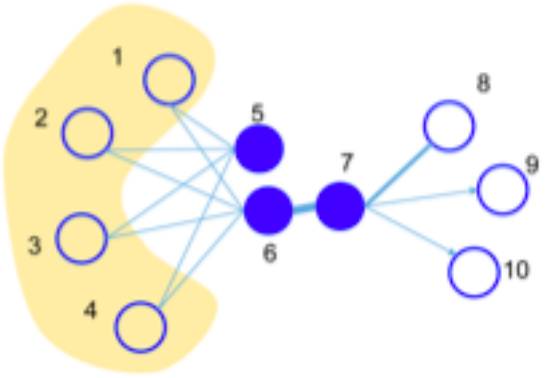


Figure 2.2: An illustration of two orders of proximity[34]

Model two orders of proximity as similarity distribution using K-L Divergence

In LINE, they try to preserve the first and second order proximity by maximizing similarities between the distribution in the embedded model (joint probability) and in observation model (empirical probability). In terms of joint probability, this is also modeled as a posterior distribution of softmax classifier or its binary class version, sigmoid function.

For the first order of proximity, the joint probability between vertices v_i and v_j are defined in Formula 2.3, where $\vec{u}_i$ and $\vec{u}_j$ are vector representations for two vertices respectively. And empirical probability for vertices v_i and v_j is the weight between two vertices $w_{i,j}$ over all weights W in the network in Formula 2.4.

$$p_1(v_i, v_j) = \frac{1}{1 + exp(-\vec{u}_i^T \cdot \vec{u}_j)} \tag{2.3}$$

$$\hat{p}_1(v_i, v_j) = \frac{w_{ij}}{W} \tag{2.4}$$

The goal is to make the similarities between two distribution probabilities p_1 and $\hat{p}_1$ as much as possible. KullbackLeibler (KL) divergence is adopted to minimize differences between two distributions as shown in Formula 2.5.

$$O_1 = -\sum_{(i,j)\in E} w_{ij} log p_1(v_i, v_j) \tag{2.5}$$

Regarding the second order proximity, each vertex is mapped to two types of representations, the original one, denoted as $\vec{u}_i$ and the "context" one, denoted as $\vec{u}'_i$. The original one only represents the vertex itself, and the "context" one represents the vertex' "context", defined as the vertex's all neighbor vertices.

The joint probability of the second order proximity $p_2(v_j|v_i)$ for vertices v_i and v_j is defined as the "context" of vertex v_j generated by vertex v_i in Formula 2.6, where $|V|$ is the number of all vertices in the network.

$$p_2(v_j|v_i) = \frac{exp(\vec{u}_j'^T \cdot \vec{u}_i)}{\sum_{k=1}^{|V|} exp(\vec{u}_k'^T \cdot \vec{u}_i)} \tag{2.6}$$

The empirical probability for vertices v_i and v_j is the weight between two vertices over the overall weights between vertex v_j and all v_i's neighbors (the out-neighbors of vertex i) as shown in Formula 2.7. This KL-divergence optimization formula is displayed in Formula 2.8.

$$\hat{p}_2(v_i|v_j) = \frac{w_{ij}}{d_i} \tag{2.7}$$

$$O_2 = -\sum_{(i,j)\in E} w_{ij} \log p_2(v_j|v_i) \tag{2.8}$$

Negative sampling and Optimization

LINE optimizes two orders of proximity O_1 and O_2 separately. But optimizing $p_2(\cdot|v_i)$ in O_2 is computationally expensive, since $|V|$ is usually a large number in the large scale network. Thus, the negative sampling [23] is adopted. For each edge (i, j), the following objective function is used:

$$p_2(v_j|v_i) = \log \sigma(\vec{u}_j'^T \cdot \vec{u}_i) + \sum_{i=1}^{K} E_{v_n \sim P_n(v)}[\log \sigma(-\vec{u}_n'^T \cdot \vec{u}_i)] \tag{2.9}$$

Where σ is the sigmoid function defined in Formula 2.3, the first term on the right side of the equation is the observed edges and the second term models the negative edges drawn from the distribution.

2.2.3 Heterogeneous Model 1: PTE

PTE [33] is short for predictive text embedding. It is designed for embedding words in a heterogeneous text network. As shown in Figure 2.3, the model first takes partially labeled text corpora to generate a heterogeneous text network containing three levels of sub-networks, word-word, word-document and word-label.

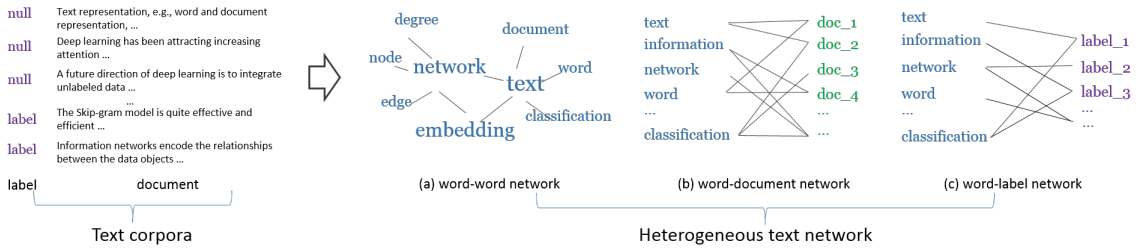


Figure 2.3: Convert a text corpora into a heterogeneous text network[33]

The first two contain unsupervised information while the word-label network encodes the

supervised information; hence PTE is a semi-supervised text embedding model.

For three networks, PTE performs a joint embedding training approaches using LINE. The only difference between PTE and LINE in the training phase is that PTE only performs negative sampling within subnetworks. For example, the negative samples for a word-word edge are also from the word-word network. Besides, PTE only uses the second order proximity information in the network, comparing to LINE.

2.2.4 Heterogeneous Model 2: Metapath2vec

Metapath2vec [7] represents the latest meta path based heterogeneous embedding model. Similar to DeepWalk, the generated paths in the Metapath2vec will also be fed into Skip-Gram for learning representations to maximize the distributed probability. But unlike DeepWalk, the paths constructed in metapath2vec are not random but follow a predefined schema, which is called meta path. As shown as an example in Figure 2.4, there three meta path candidates. Take the first one (APA) for instance, the walk generator will choose to visit a paper vertex after visiting an author vertex, and the next vertex to a paper vertex will be an author again.

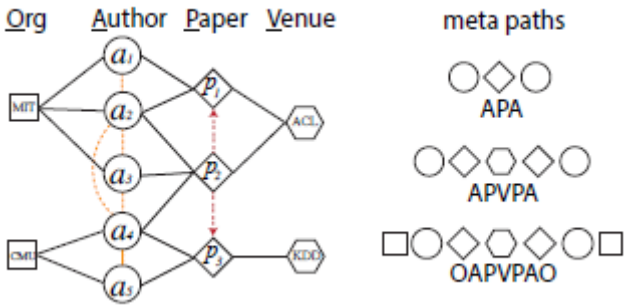


Figure 2.4: Academic network and three meta paths

Unfortunately, the model was released in late Aug. 2017. It is not included as the benchmark, but it will be put into future work.

Chapter 3

Problem Definition

We formally define the problem of heterogeneous network embedding using edge-type based Graph Partition and Space Projection. Firstly, we follow the idea in [34] to formulate homogeneous network:

Definition 3.1. ***Homogeneous Network*** *A homogeneous network is defined as a graph G=(V,E) in which V is the set of vertices, each representing an object and E is the set of edges, each representing a relationship between two objects. Each edge $e \in E$ is a pair of two vertices $u, v \in V$, denoted as $e = (u, v)$. If G is a weighted graph, e will be also associated with a weight $w_{uv} > 0$. If G is directed, pair (u, v) will be ordered, which means $(u, v) \not\equiv (v, u)$ and $w_{uv} \not\equiv w_{vu}$; if G is undirected, pair (u, v) will be unordered, which means $(u, v) \equiv (v, u)$ and $w_{uv} \equiv w_{vu}$.*

A network could either be weighted or unweighted, directed or undirected. For example, paper citation network is a typical unweighted but directed network while coauthor network is a weighted network with undirected edges.

In practice, to regulate the format of networks, four types of networks are unified to one type of weighted and directed networks. An unweighted network could be regarded as a weighted network where weight value is fixed, and an undirected network could be treated as a directed network where every edge can find its reverse one.

All vertices and edges in homogeneous network are hypothesized to be the same type. However, real networks are usually heterogeneous, vertices represent different types and

edges indicate various relations. We first define the simplest heterogeneous network, Bipartite Network, which contains two types of vertices and one type of edges:

Definition 3.2. ***Bipartite Network*** *A bipartite network is defined as a graph* $G = (V, E)$ *where* $V = V_1 \cup V_2$ *and* $E = E_{V_1V_2}$. V_1 *and* V_2 *are two types of vertex sets, and in the network there is only one type of edges* $e_{v_1v_2} \in E_{V_1V_2}$ *connecting two vertices* $v_1 \in V_1$ *and* $v_2 \in V_2$.

In terms of the definition for general Heterogeneous Network, we adopt the one in [7] as follows:

Definition 3.3. ***Heterogeneous Network*** *A heterogeneous network is defined as a graph G=(V,E,T) where each vertex* $v \in V$ *and each edge* $e \in E$ *are associated with their mapping functions* $\phi(v) : V \to T_V$ *and* $\phi(e) : E \to T_E$. T_V *and* T_E *denote the sets of object type and relation type respectively, where* $|T_V| + |T_E| > 2$.

For example, Figure 3.1 shows a typical academic network containing three types of vertices, author(A), paper(P), organization(O), wherein edges could represent four relations, coauthor(A-A), publish(A-P), citation(P-P), affiliation(A-O).

According to the above definitions, it is clear that bipartite network is one type of heterogeneous network where $|T_V| = 2$ and $|T_E| = 1$.

Since vertices and edges represent different types of objects and relations in the heterogeneous network, it is natural to consider methods which can divide the complex heterogeneous network into simpler networks. This process is defined as follows:

Definition 3.4. ***Graph Partition*** *For a network* $G = (V, E)$, *graph partition is to partition G into smaller components with specific properties.*

Here, in our graph partition approach, we define the specific property in smaller components is only containing one type of edges, named as Edge-type Based Graph Partition:

Definition 3.5. ***Edge-type Based Graph Partition*** *For a heterogeneous network G, partition it into minimum number of subnetworks* $(G_1, ..., G_N)$,*where* $G_i \subseteqq G$ *and* $G \subseteqq \bigcup_{i=1}^{N} G_i$. *In each subnetwork* $G_i = (V, E, T)$, $|T_E| = 1$, *i.e. each subnetwork is a homogeneous or bipartite network.*

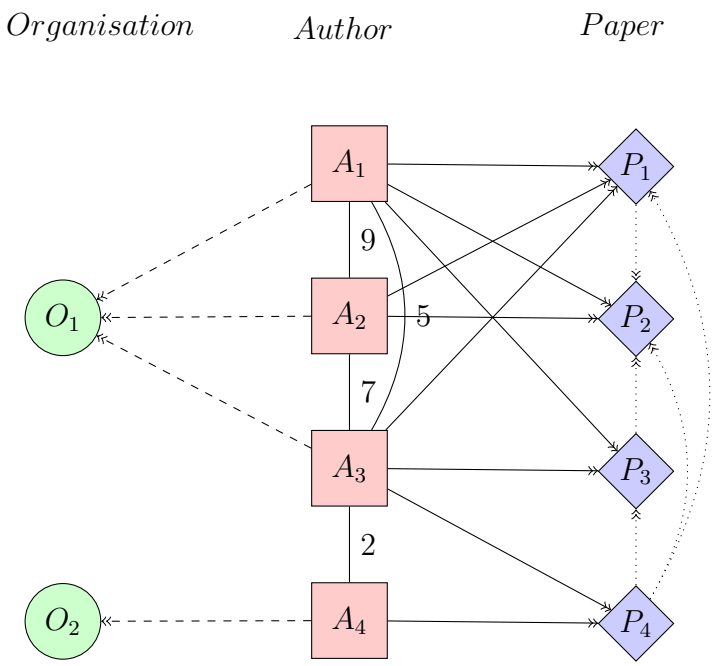


Figure 3.1: One example of heterogeneous network: academic network Dashed arrows denote unweighted but directed affiliation relationships, solid lines denote weighted but undirected coauthor relationships, solid arrows denote unweighted but directed publish relationships and dotted arrows denote unweighted but directed citation relationships.

The idea behind is to divide a network containing N types of edges into N subnetworks, each fully containing one unique type of edges and relevant vertices in the original network. Figure 3.2 shows subnetworks of the heterogeneous network in Figure 3.1 using edge-type based Graph Partition. Affiliation and publish subnetworks are bipartite networks while coauthor and citation subnetworks are homogeneous networks.

The process of taking the heterogeneous network as input and using edge-type based Graph Partition for network embedding, is formalized as follows:

Definition 3.6. ***Heterogeneous Network Embedding*** *Given a heterogeneous network $G = (V, E, T)$, the task of heterogeneous network embedding is to represent each vertex $v \in V$ in space $R^{|V|}$ into a low-dimensional space R^d, where $d \ll |V|$. These vertex representations in R^d can preserve the structural and semantic relations in $R^{|V|}$.*

The output of this heterogeneous network embedding is the representation matrix $X \in R^{|V| \times d}$, where i^{th} row is the d-dimensional numeric vector representation for vertex $v_i \in V$.

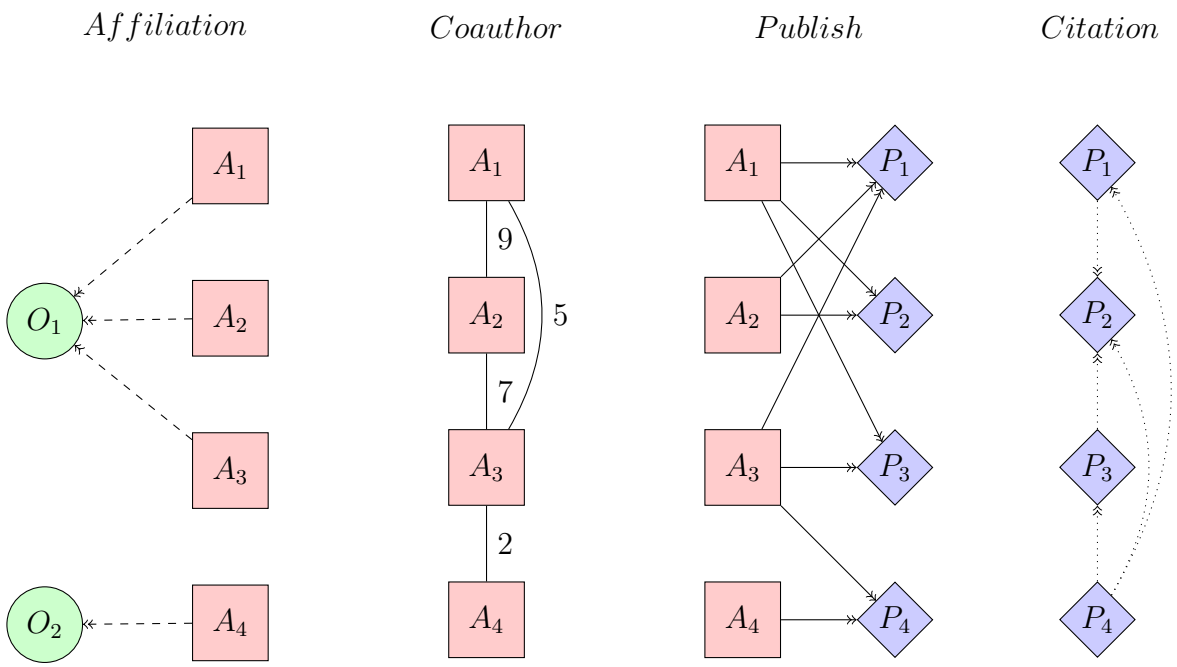


Figure 3.2: Four subnetworks after using edge-type based Graph Partition in Figure 3.1

Chapter 4

The GPSP Framework

The embedding model we propose for heterogeneous network, GPSP should satisfy several requirements: first, it is able to preserve desirable structural and semantic relations among original vertices; second, the model should keep vertices' type information; third, it must be scalable for very large networks, say millions of vertices and tens of millions of edges, fourth, it should be able to deal with directed, undirected, weighted and unweighted networks. The objective of GPSP is to embed heterogeneous network using related homogeneous and bipartite networks.

4.1 Edge-type Based Graph Partition

Since heterogeneous networks contain various types of vertices and edges, it would be suspicious to learn heterogeneous vertices representations in the same low-dimensional latent space. We proposed one graph partition method, AKA edge-type based Graph Partition, as previously illustrated in Figure 3.1 and 3.2. This approach will be elaborated in this section.

4.1.1 Building Type Table

We firstly sketch all types of vertices and edges in a heterogeneous network $G = (V, E, T)$ as shown in Table 4.1 where there are M types of vertices and N types of edges. In this type table, each edge type E_i is defined by source vertex type V_j and target vertex

type V_k. The table is diagonally asymmetric if the network contains any unidirectional relation, like we can say people eat fruit but we cannot say it reversely. Based on this, two edges with the same pair of source and target vertices could be different types. For example in the table, edge $\{V_{M-2} \to V_2\}$ leads to type E_2 while $\{V_2 \to V_{M-2}\}$ creates a different edge type E_{N-2}.

		Source Vertex Type						
		V_1	V_2	V_3	$\cdots$	V_{M-2}	V_{M-1}	V_M
Target Vertex Type	V_1	E_1			$\cdots$			
	V_2				$\cdots$	E_2	E_3	
	V_3				$\cdots$			
	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
	V_{M-2}		E_{N-2}		$\cdots$			
	V_{M-1}			E_{N-1}	$\cdots$			
	V_M	E_N			$\cdots$			

Table 4.1: Edge and Vertex Type Table

4.1.2 Graph Partition

Based on the type table, perform edge-type based Graph Partition to divide network G into N subnetworks $G_1, G_2, \cdots, G_i, \cdots, G_{N-1}, G_N$ where $G_i = (V, E_{T_i})$, E_{T_i} is the compete set of T_i type edges in G. Subnetworks like $G_i = (V_1, E_{T_1})$ in Table 4.1 are homogeneous networks while the ones such as $G_j = (V_{M-2} \cup V_2, E_{T_2})$ are categorized as bipartite networks. For homogeneous networks, perform homogeneous network embedding as introduced in the following section.

4.2 For Homogeneous Network: Embedding

For homogeneous network embedding, we could adopt any existing embedding model. In this paper, we use two famous homogeneous network embedding models as introduced in chapter two, LINE and DeepWalk.

4.2.1 Model One: LINE

For each homogeneous subnetwork, perform LINE model to learn representations. The first and second orders of proximity in the network are preserved via minimizing the KL-divergence between joint probability and empirical probability. LINE was designed only to accept directed and weighted network, here we modified it to accept undirected or unweighted network.

4.2.2 Model Two: DeepWalk

Similarly, perform DeepWalk on each homogeneous subnetwork. The walk generator randomly walks around vertices in the network to generate paths for Skip-Gram to learn representations for each vertex. The model was slightly changed to be acceptable for weighted network since the original one cannot.

4.3 For Bipartite Network: Latent Space Projection

After learning representations from homogeneous subnetworks, how to utilize the information hidden in the bipartite subnetworks? Unlike homogeneous network, vertices are not the same types in the bipartite network, but previous heterogeneous network embedding algorithms like PTE ignore this point. Here, instead of embedding bipartite network directly, we decided to choose an alternative way, latent Space Projection.

4.3.1 Projective Relation

Figure 4.1 is a subset of a bipartite network. Since we have learned representations of O_i and P_i in two different homogeneous networks in Section 4.2, or say in two different low-dimensional spaces. We could treat the relation between objects O_i and P_i in this bipartite network as the implicit projection between two spaces, as demonstrated in Figure 4.2.

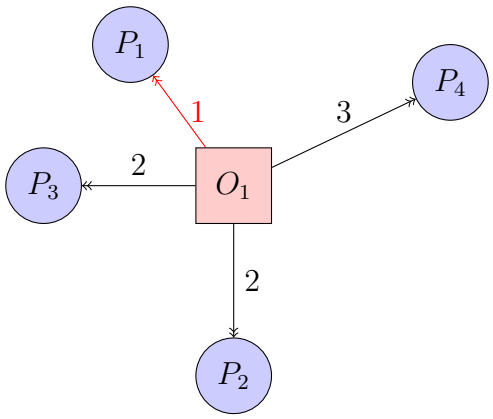


Figure 4.1: Subset of a bipartite network

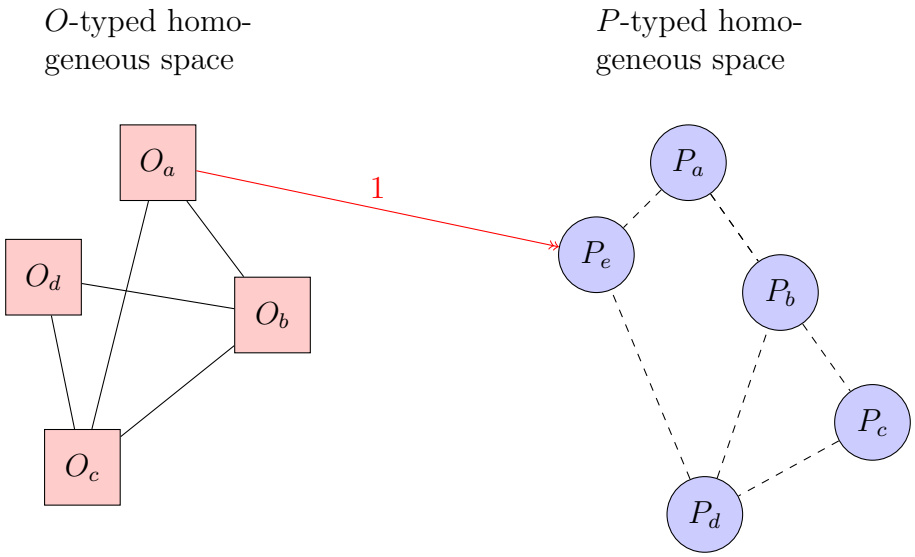


Figure 4.2: An illustration of Space Projection

4.3.2 Space Projection

Based upon this projective relation between two types of vertices, we could learn representations of one type vertices in another type space via all projections between these two spaces. Formula 4.1 formulates the representation learning process. In two homogeneous networks O and P, each vertex O_i could learn a projective representation $E_{O_i \to P}$, where

$$E_{O_i \to P} = \frac{1}{N} \sum_{j=1}^{N} (E_{P_j} * w_{O_i P_j}) \tag{4.1}$$

Where $\to$ denotes the projection relation between two vertices in two spaces,$\{P_N\}$ is the complete set of objects in P that has $P_j \in P_N$ and $O_i \to P_j$, E_{P_j} is the learned representation of object P_j in the homogeneous network, $w_{O_i P_j}$ is the weight of the edge

between O_i and P_j.

Exemplify this formula using Figure 4.2, O_1 connects four P-type objects P_1, P_2, P_3 and P_4 in this bipartite network. The projective representation of O_1 in P network would be the weighted mean of the embedding vectors for P_1, P_2, P_3 and P_4, the equation is shown as follows:

$$E_{O_1} = \frac{1}{4} * (E_{P_1} + 2 * E_{P_2} + 2 * E_{P_3} + 3 * E_{P_4})$$

4.3.3 Discussion

This projective relation should be reversible. The embedding of P-type vertices could also be learned in O-type network via the same projective relation. Moreover, it is possible to have two projective relations between two spaces. For example, a group of people write textbooks for a larger group of people. There are two projective relations between "people" and "books" types. The first group of people write books, and textbooks educate the second group of people, as illustrated in Figure 4.3. The "write" and "educate" relations could be both used to learn projective representations for people and textbooks networks.

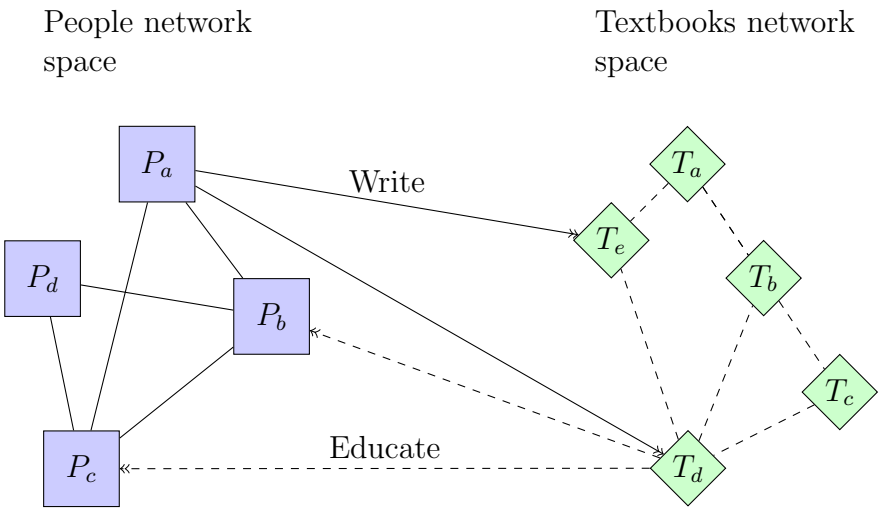


Figure 4.3: Two reverse projective relations

4.4 Representations Concatenation

In section 4.2, we learned the representation of a vertex in its homogeneous network and in section 4.3, we learned the projective representation of a vertex in the bipartite network. As shown in Figure 4.5, our final heterogeneous embedding for each vertex concatenates potentially one homogeneous embedding and potentially several projective embeddings. For example, in Table 4.1, the heterogeneous embedding for V_1-typed vertices contains one homogeneous embedding and another projective embedding in $\{V_1 \to V_M\}$ bipartite network, while V_2-typed vertices' heterogeneous embedding only consists of three projective embeddings from $\{V_2 \to V_{M-2}\}$, $\{V_{M-2} \to V_2\}$ and $\{V_{M-1} \to V_2\}$ bipartite networks.

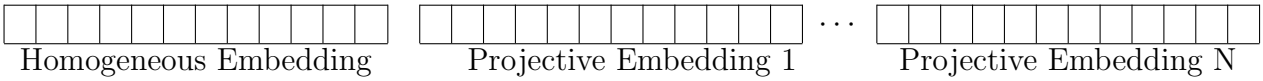


Figure 4.4: Concatenate representations

4.5 Discussion

If each embedding is regarded as multi-dimensional coordinates in the relevant latent space, homogeneous embedding indicate the vertex's position in its own type space, and the projective embedding implicitly represent the vertex's hidden position in another type space. So the final heterogeneous embedding is just an ensemble of coordinates for a single vertex in different spaces. Since different spaces contain different information, the ensemble embeddings should represent the vertex better than the single embedding. Comparing to the previous embedding approaches, this approach both preserve vertices' type property, and structural information via homogeneous embedding. The following is the algorithm of the GPSP framework.

Algorithm 1: The GPSP algorithm

Data: The heterogeneous information network $G = (V, E, T)$, number of negative samples n, number of walks per node w, walk length l, embedding dimension d, neighborhood size k.

Result: The latent vertex embeddings $X \in R^{|V| \times d}$

subnetworkList = EdgeTypedBasedGraphPartition(G);

homogeneousEmbeddingList;

projectiveEmbeddingList;

for *i = 1 to len(subnetworkList)* **do**

  if *subnetworkList[i] is homogeneous network* **then**

    add LINE($subnetworkList[i], n, d$) or DeepWalk($subnetworkList[i], w, l, d, k$) into homogeneousEmbeddingList;

  else

    add SpaceProjection(homogeneousEmbeddingList,subnetworkList[i]) into projectiveEmbeddingList

  end

end

X;

for *i = 1 to len(homogeneousEmbeddingList)* **do**

  a = vertex type in homogeneousEmbeddingList[i] idx = index row of type a in X X = concatenation of X and homogeneousEmbeddingList[i] in row idx

end

for *i = 1 to len(projectiveEmbeddingList)* **do**

  a = vertex type in projectiveEmbeddingList[i] idx = index row of type a in X X = concatenation of X and projectiveEmbeddingList[i] in row idx

end

Algorithm 2: The EdgeTypedBasedGraphPartition algorithm

Data: The heterogeneous information network $G = (V, E, T)$

Result: *subnetworkList*

edgeTypeList;

for $i = 1 \rightarrow len(E)$ **do**

  if E_i *not the type in edgeTypeList* **then**

    add E_i's type into edgeTypeList;

  end

end

subnetworkList;

for $i = 1 \rightarrow len(E)$ **do**

  j = the index of the element in subnetworkList to contain E_i type;

  add E_i to subnetworkList[j];

end

Algorithm 3: The SpaceProjection algorithm

Data: *homogeneousEmbeddingList*, bipartite network *bn*

Result: projectiveEmbedding

projectiveEmbedding;

[*a*,*b*] = two vertex types in *bn*;

if *type a in homogeneousEmbeddingList* **then**

  for *each vertex in type b* **do**
    add result of Eq. 4.1 into projectiveEmbedding

  end

end

if *type b in homogeneousEmbeddingList* **then**

  for *each vertex in type a* **do**
    add result of Eq. 4.1 into projectiveEmbedding

  end

end

Chapter 5

Experiments

In this section, we evaluate the efficiency and efficacy of the GPSP for heterogeneous network embedding via multiple machine learning problems. We built two GPSP models, GPSPL which uses LINE as the homogeneous embedding model and GPSPD that adopts DeepWalk to embed homogeneous subnetworks.

5.1 Experimental setup

5.1.1 Data Set

We constructed an academic heterogeneous network, based on the dataset from AMiner Computer Science[1] [35]. The AMiner dataset contains $9,323,739$ computer scientists and $3,194,405$ papers from $3,883$ computer science venues. The constructed network consists of two types of vertices: authors and papers, and three types of edges: authors coauthor with each other, authors write papers, papers cite other papers. After edge-based Graph Partition, two homogeneous subnetwork coauthor (A-A) network and citation (P-P) network, and one bipartite network, writing (A-P) network will be generated.

[1] https://aminer.org/aminernetwork

5.1.2 Benchmark Algorithms

We compared GPSP with several recent successful homogeneous or heterogeneous network embedding algorithms as benchmarks:

1. Deepwalk[27]: We directly fed the heterogeneous network as input into the DeepWalk model.

2. LINE[34]: We used three LINE models in the experiment, LINE with first order proximity (LINE-1st), LINE with second order proximity (LINE-2nd) and LINE with both first and second orders of proximity (LINE-1st+2nd).

3. PTE[33]: PTE is initially designed for semi-supervised embedding for heterogeneous text network by preserving the second order proximity in LINE. Here we modified the model to embed our heterogeneous network in an unsupervised fashion.

5.1.3 Parameter Settings

For GPSP and all benchmark models listed above, we try to use the same parameters.

1. The homogeneous and projective embeddings dimension is 128. (The dimension in LINE-1st+2nd is $128 + 128$)

2. The size of negative samples is 5.

3. The number of random walks to start at each node in deepwalk is 10.

4. The walk length in deepwalk is 40.

We evaluated the performance of the embeddings learned via different algorithms over two network mining tasks, multi-label classification and node clustering. Besides, we also visualized the vertices representations into a 2-D space via T-SNE [20].

5.1.4 Configuration

All the models are run on a single Linux machine with 32G memory, Xeon E5-1650 v3 CPU at 3.5GHZ using 12 threads. The codes of LINE[2] and DeepWalk[3] are collected from Github and PTE is implemented upon the code of LINE in C++. The model GPSP and network mining tasks are mainly implemented in Python code with libraries like Pandas, Numpy and Sklearn.

5.2 Multi-label classification

5.2.1 Procedure

For the multi-label classification, we use the labeled dataset generated by [7] which groups authors into 8 categories based on authors' research fields[4]. We try to match this label set with our author embeddings, and we get 103,024 successfully matching author embeddings with their labels.

A linear SVM is used to classify these author embeddings while the ratio of the training set is set from 10% to 90% and the rest is all for testing. We evaluate the average performance of each embedding algorithm for 10 random runs in Micro-F1 and Macro-F1 scores. The variances of the performance between different runs are usually within 2%. The formula of F1 score is shown below. Precision and recall in Micro-F1 are calculated based on values of true positive, false positive or false negative in each class while precision and recall in Macro-F1 are just the average values of precision and recall in all classes.

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{5.1}$$

Moreover, to separately explore the information hidden in homogeneous and projective embeddings, homogeneous author embedding in the author network is denoted as "GPSP-

[2]https://github.com/tangjianpku/LINE

[3]https://github.com/phanein/deepwalk

[4]8 groups are Computing Systems, Theoretical Computer Science, Computer Networks & Wireless Communication, Computer Graphics, Human Computer Interaction, Computational Linguistics, Computer Vision & Pattern Recognition, Databases & Information Systems.

author", and projective author embedding using the paper network and the "writing-relation" bipartite network is denoted as "GPSP-paper".

5.2.2 Result and evaluation

Table 5.1 and 5.2 report the results of multi-label classification using different models measured by two metrics, Micro-F1 and Macro-F1. Overall, the two proposed models GPSPL and GPSPD outperform all benchmarks for both metrics. When varying the training ratio, the performances of models are quite stable. The constant gains achieved by both proposed models are around 3%-5% over the related baselines.

When investigating the contributions of homogeneous part and projective part in the complete GPSP embedding by evaluating these two parts respectively, it turns out the projective part, i.e. the embedding in method GPSP-paper, performs better than the embedding the GPSP-author for both LINE and DeepWalk related models, indicating that projective embedding offers a larger contribution to the embedding in the complete model GPSP.

It is also noticeable that LINE-related models outperform DeepWalk-related models. As proposed in [34], they take DeepWalk as a network embedding algorithm which only considers the structural information (second order proximity) by randomly walking. However, DeepWalk still performs better than LINE-2nd model which might be because higher orders of proximity are recorded in the random walking than the just second order proximity in LINE-2nd.

In summary, GPSP generates more appropriate embeddings for heterogeneous networks than the current state-of-the-art baselines, by evaluating the performance in the multi-label classification problem. This indicates the proposed model can preserve better vertices' information for embedding heterogeneous networks.

Metric	Method	10%	20%	30%	40%	50%	60%	70%	80%	90%
Micro-F1	LINE-1st	0.7003	0.7069	0.7081	0.7087	0.7087	0.7084	0.7079	0.7087	0.7079
	LINE-2nd	0.6436	0.6446	0.6457	0.6462	0.6463	0.6458	0.6456	0.6450	0.6470
	LINE-1st+2nd	0.7062	0.7064	0.7067	0.7075	0.7074	0.7077	0.7062	0.7072	0.7075
	PTE	0.7122	0.7125	0.7129	0.7135	0.7133	0.7138	0.7140	0.7135	0.7138
	GPSPL-author 1st	0.6390	0.6420	0.6430	0.6436	0.6439	0.6432	0.6426	0.6448	0.6455
	GPSPL-author 2nd	0.6162	0.6179	0.6184	0.6186	0.6181	0.6181	0.6183	0.6199	0.6212
	GPSPL-author 1st+2nd	0.6487	0.6509	0.6515	0.6519	0.6522	0.6515	0.6519	0.6534	0.6540
	GPSPL-paper 1st	0.7118	0.7148	0.7136	0.7156	0.7167	0.7127	0.7219	0.7206	0.7227
	GPSPL-paper 2nd	0.6532	0.6546	0.6553	0.6554	0.6546	0.6540	0.6552	0.6521	0.6565
	GPSPL-paper 1st+2nd	0.7235	0.7247	0.7247	0.7252	0.7256	0.7250	0.7262	0.7256	0.7267
	GPSPL 1st	0.7344	0.7378	0.7397	0.7396	0.7391	0.7401	0.7410	0.7425	0.7388
	GPSPL 2nd	0.7121	0.7128	0.7141	0.7130	0.7148	0.7146	0.7137	0.7145	0.7159
	GPSPL 1st+2nd	**0.7512**	**0.7540**	**0.7557**	**0.7564**	**0.7564**	**0.7558**	**0.7554**	**0.7574**	**0.7552**
Macro-F1	LINE 1st	0.6996	0.7050	0.7061	0.7069	0.7067	0.7062	0.7056	0.7063	0.7059
	LINE 2nd	0.6389	0.6400	0.6413	0.6417	0.6419	0.6415	0.6409	0.6403	0.6426
	LINE 1st+2nd	0.7032	0.7034	0.7036	0.7046	0.7043	0.7049	0.7035	0.7044	0.7036
	PTE	0.7089	0.7093	0.7094	0.7098	0.7101	0.7104	0.7090	0.7099	0.7094
	GPSPL-author 1st	0.6399	0.6427	0.6434	0.6439	0.6438	0.6436	0.6424	0.6451	0.6451
	GPSPL-author 2nd	0.6119	0.6136	0.6141	0.6143	0.6140	0.6138	0.6138	0.6162	0.6169
	GPSPL-author 1st+2nd	0.6477	0.6498	0.6506	0.6507	0.6508	0.6501	0.6506	0.6529	0.6528
	GPSPL-paper 1st	0.7087	0.7112	0.7099	0.7120	0.7130	0.7083	0.7198	0.7177	0.7211
	GPSPL-paper 2nd	0.6557	0.6574	0.6580	0.6582	0.6571	0.6570	0.6578	0.6550	0.6591
	GPSPL-paper 1st+2nd	0.7212	0.7226	0.7226	0.7230	0.7231	0.7229	0.7243	0.7232	0.7251
	GPSPL 1st	0.7318	0.7356	0.7369	0.7369	0.7361	0.7374	0.7388	0.7402	0.7364
	GPSPL 2nd	0.7111	0.7117	0.7132	0.7119	0.7139	0.7137	0.7130	0.7136	0.7155
	GPSPL 1st+2nd	**0.7482**	**0.7513**	**0.7527**	**0.7534**	**0.7534**	**0.7529**	**0.7526**	**0.7544**	**0.7522**

Table 5.1: Multi-label classification results for author embeddings in LINE-related algorithms

Metric	Method	10%	20%	30%	40%	50%	60%	70%	80%	90%
Micro-F1	Deepwalk	0.6992	0.6998	0.7010	0.7008	0.6992	0.6988	0.6986	0.6964	0.6988
	GPSPD-author	0.5919	0.5936	0.5950	0.5968	0.5963	0.5993	0.5974	0.5995	0.5980
	GPSPD-paper	0.7010	0.7011	0.7016	0.7019	0.7021	0.7020	0.7018	0.7023	0.7020
	GPSPD	**0.7275**	**0.7304**	**0.7318**	**0.7330**	**0.7324**	**0.7328**	**0.7320**	**0.7331**	**0.7318**
Macro-F1	Deepwalk	0.6964	0.6969	0.6982	0.6981	0.6965	0.6964	0.6963	0.6937	0.6961
	GPSPD-author	0.5872	0.5887	0.5912	0.5922	0.5912	0.5977	0.5941	0.5971	0.5944
	GPSPD-paper	0.7012	0.7015	0.7018	0.7020	0.7022	0.7021	0.7018	0.7023	0.7016
	GPSPD	**0.7253**	**0.7280**	**0.7290**	**0.7300**	**0.7298**	**0.7302**	**0.7295**	**0.7306**	**0.7289**

Table 5.2: Multi-label classification results for author embeddings in DeepWalk-related algorithms

5.3 Clustering

5.3.1 Procedure

We also illustrate how the latent representations learned by different embedding models using another network mining task, node clustering. Similar to multi-label classification, we ran the each model 10 times and recorded the average score. The clustering algorithm is K-means clustering where k is assigned to 8, and the evaluator for clusters is normalized mutual information (NMI) [31], which measures the mutual information contained between the generated cluster and the label cluster introduced in the multi-label

classification task.

5.3.2 Result and evaluation

Table 5.3 and 5.4 list the result of both LINE-related and DeepWalk-related models using NMI. Overall, two GPSP models GPSPL and GPSPD outperform their baselines around 5%-7%, and GPSPD performs better than GPSPL to 4%.

Unlike the performance between 1-st and 2-nd order proximity models in the multi-label classification, the 2-nd order proximity model performs much better in node clustering task, even better than the complete LINE-based model in several cases. We could make a hypothesis based on this finding, that structural information (2nd order proximity) matters more for node clustering task. This assumption is supported by the performance in the DeepWalk-based models. Since DeepWalk preserves even higher order proximity in the network, it , in general, performs better than LINE-based models.

The poor performance for GPSPL-paper 1st model might suggest that in citation network, the first order proximity might not be able to preserve the cluster information. On the contrary, the poor performance for GPSPD-author model could mean that high order proximity for coauthor network is not suitable for finding representation for clustering.

Method	LINE	PTE	GPSPL-author	GPSPL-paper	GPSPL
1-st order	0.3015	NA	0.2609	0.0447	0.1049
2-nd order	0.2529	0.2634	0.2505	0.2403	**0.3118**
1st2nd order	0.2516	NA	0.2607	0.1738	0.1894

Table 5.3: Node results for author embeddings in LINE-related algorithms

Method	Deepwalk	GPSPD-author	GPSPD-paper	GPSPD
performance	0.2873	0.1681	0.3392	**0.3555**

Table 5.4: Node results for author embeddings in DeepWalk-related algorithms

5.4 Parameter Sensitivity

In the GPSPL and GPSPD models, they share one parameter, the number of dimensions of the homogeneous embedding. We conduct a sensitivity analysis of the method GPSPL

1st+2nd on this parameter for multi-label classification. Table 5.5 shows the result of Mirco-F1 and Macro-F1 scores using embeddings with different sizes of dimension, where the default dimension for both GPSPL-author and GPSPL-paper model is 256. The training ratio is set to 50%.

From the table, we can learn the computational cost (dimensions of the embedding) and the efficacy (Micro-F1 or Marco-F1 score) can be balanced at the dimension size as 256 for each part of embedding. When the dimension is larger than 256, the performance stops to improve and becomes stable. But from 128 to 256, we could still clearly see an improvement as shown in the table. Furthermore, only expanding the size of one part embedding will not help the model to reach the optimal performance. For example, the performance of model (Paper-128+Author-1024) is still worse than the model (Paper-256+Author-256), indicating the information contained in each part of embedding is limited comparing to the embedding for the full network.

		Paper				
		128	256	512	768	1024
Author	128	0.7356	0.7542	0.7523	0.7520	0.7525
	256	0.7387	0.7556	0.7567	0.7562	0.7568
	512	0.7364	0.7545	0.7563	0.7559	0.7569
	768	0.7368	0.7554	0.7564	0.7568	0.7572
	1024	0.7370	0.7557	0.7562	0.7572	0.7578

(a) Parameter sensitivity: Micro-F1

		Paper				
		128	256	512	768	1024
Author	128	0.7289	0.7467	0.7484	0.7513	0.7504
	256	0.7312	0.7479	0.7501	0.7503	0.7509
	512	0.7314	0.7475	0.7498	0.7508	0.7502
	768	0.7309	0.7480	0.7501	0.7508	0.7513
	1024	0.7316	0.7476	0.7498	0.7503	0.7501

(b) Parameter sensitivity: Macro-F1

Table 5.5: Parameter sensitivity in multi-label classification

5.5 Scalability

As said before, the network which the embedding models need to tackle usually consists of millions of vertices and tens of millions of edges. So it is necessary to demonstrate the scalability of the embedding model GPSP. The scale mechanism for LINE and DeepWalk can be adopted to GPSP as well. All experiments are run on a single machine with the same configuration introduced above, but with different threads, 1,2,4,6,12, each of them

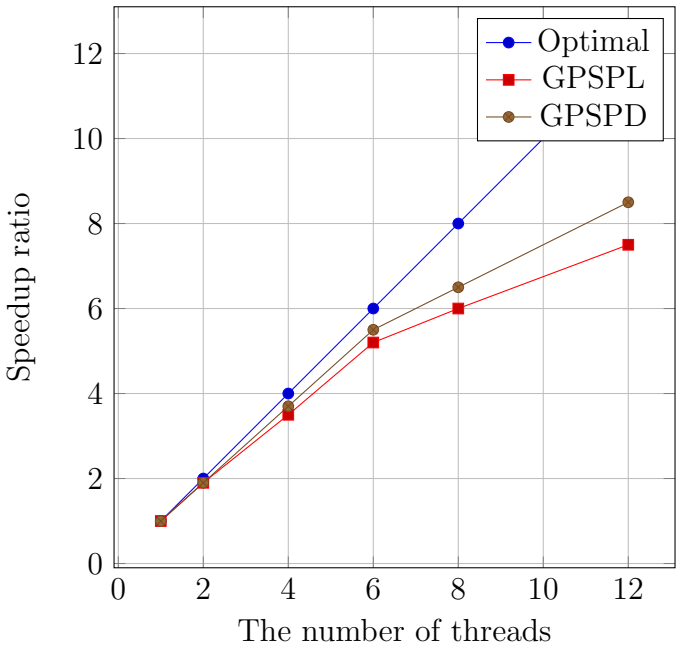


Figure 5.1: Scalability of GPSP

utilizing one CPU logical core.

Figure 5.1 shows the speedup ratio of GPSPL and GPSPD with respect to the number of threads in the model. Theoretically, the optimal scalable model should perform as denoted as the blue line. But generally, two models still show their ability to achieve sub-linear speedup ratio, which is acceptable. Specifically, GPSPD is slightly better than GPSPL because generating random paths in GPSPD can be easily parallelized. By using 12 cores, GPSPL takes 73 minutes to run GPSPL-author 1st and 76 minutes to run GPSPL-author 2nd while it costs 86 minutes to run GPSPD-author model. $156,0640$ vertices and $851,7892$ directed edges consist of the author homogeneous network. Overall, the proposed model GPSP has shown its ability to embed large-scale heterogeneous networks.

5.6 Visualization

TensorFlow provides an Embedding Projector[5] to visualize the high-dimensional embeddings into low-dimensional space using PCA or T-SNE. We try to feed the 4 DeepWalk-related embeddings into the projector using T-SNE to train the 2 dimensional projections of all authors.

[5]http://projector.tensorflow.org/

T-SNE [20] is short for t-distributed Stochastic Neighbor Embedding. It projects high dimension data into 2 or 3-dimensional space for visualization. It measures similarities between data points using joint probabilities and adopts Kullback-Leibler (KL) divergence to minimize the probabilities between the low-dimensional projection and the high-dimensional data.

The Projector will randomly sample the large input down to $10,000$ data points. Figure 5.2 is the 2D projections of embeddings of GPSPD. The numeric labels in the projection match eight subfields in Computer Science, as shown in Table 5.6. (If the label is blurry in the printed version, please refer to the original images stored in Google Drive [6].)

From this figure, we can clearly see the GPSPD model roughly groups authors in the same subfields together. Besides, we can find more hidden information from this projection. For example, cluster 4 (light green) and 7 (light blue) are located quite closely in the projection, while their representing subfields Computer Graphics and Computer Vision & Pattern Recognition are both related to handling images-related problems.

Computing Systems	Theoretical Computer Science	Computer Networks & Wireless Communication
1	2	3
Computer Graphics	Human Computer Interaction	Computational Linguistics
4	5	6
Computer Vision & Pattern Recognition	Databases & Information Systems	
7	8	

Table 5.6: Eight subfields in CS and their numerical representations in the projection

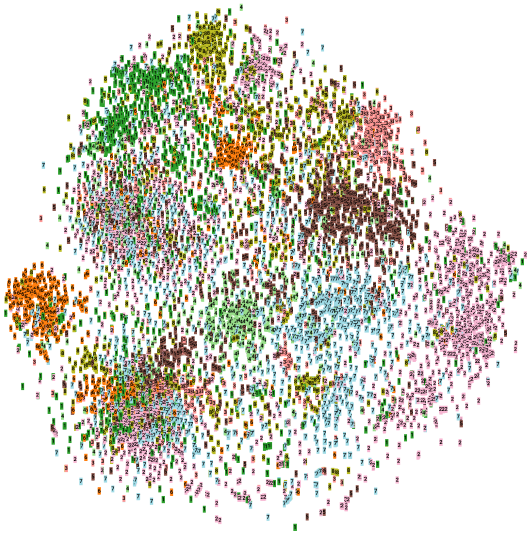
Figure 5.3 summarizes four projections generated by four DeepWalk-based models, DeepWalk, GPSPD-author, GPSPD-paper and GPSPD. As shown in node clusteing part, (Table 5.4), it is clear that the projection of the embedding in GPSPD-author is much worse than the rest three in grouping authors. Comparing to the one in DeepWalk, our proposed GPSPD model can generate better embedding that is able to group author geographically more clearly in the T-SNE projection format.

Thus, these visualizations can intuitively demonstrate GPSP's ability to discover the latent information among multiple types of vertices in the heterogeneous network.

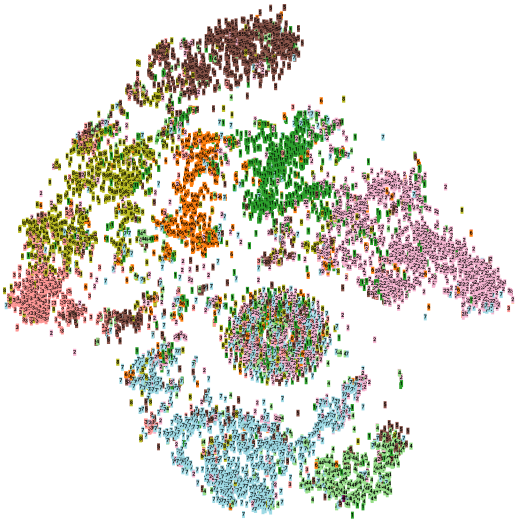
[6] https://drive.google.com/open?id=0B3XilEbNpq1ycG0tR3pBMHhwX3c

Figure 5.2: 2D T-SNE projections of embeddings in GPSPD

(a) 2D T-SNE projections of embeddings in DeepWalk



(b) 2D T-SNE projections of embeddings in GPSPD-author



(c) 2D T-SNE projections of embeddings in GPSPD-paper

(d) 2D T-SNE projections of embeddings in GPSPD

Figure 5.3: 2D T-SNE projections of four DeepWalk-related embeddings

Chapter 6

Conclusion and Future work

6.1 Conclusion

In this work, we formally define the edge-based Graph Partition approach to divide a complex network into atomic subnetworks, where an atomic subnetwork is defined as a subnetwork only containing one type of edges. We also introduce the Space Projection concept which can learn a vertex's different representations from multiple homogeneous networks. To implement the above concepts, we propose the GPSP, a heterogeneous network embedding model. In GPSP, edge-based Graph Partition is applied to the heterogeneous network to generate two types of atomic subnetworks, homogeneous ones and bipartite ones. Then, we use LINE and DeepWalk to embed each homogeneous networks to generate homogeneous embeddings. Next, for each bipartite network, we use the projective relation in the network to generate projective embeddings for the related types of vertices. Finally, GPSP concatenates learned homogeneous embeddings and projective embeddings as the output of the model. Extensive experiments have shown the advantages of GPSP comparing to all benchmarks in various heterogeneous network mining tasks, such as node classification and clustering, or visualization.

6.2 Reflection on Project Plan

As shown in the Appendix, the plan is vastly different from the actual execution. The planned idea was to use the heterogeneity classifier in HSCA [14] to separate the heterogeneous network into homogeneous networks. But the further research shows that this approach neither considers the relation between different types of objects nor is scalable for large networks. Also, because the network embedding is a quite new and fast-changing field, there were no models for embedding general heterogeneous networks when the plan was written.

6.3 Future Work

There are a couple of optimizations and improvements in the future.
Since the data and code for the latest meta path based heterogeneous network embedding model metapath2vec was only available one week before the deadline, there was not enough time to perform experiment on it. In the future, metapath2vec should be included as one benchmark.
From the classification and clustering result, it seems like coauthor network contains more information in the first order proximity form while high order proximity is better for citation network. How about making an ensemble model to mix the proposed LINE-based and DeepWalk-based model?
What will happen if we iterate the embedding process in the model multiple times? Specifically, how about feeding the learned embedding back to the model to learn the reverse projective embedding via bipartite subnetworks and retrain the embedding via homogeneous subnetworks? These questions may lead to interesting answers.

Bibliography

[1] Aiello, L. M., Barrat, A., Schifanella, R., Cattuto, C., Markines, B., and Menczer, F. Friendship prediction and homophily in social media. *ACM Transactions on the Web (TWEB) 6*, 2 (2012), 9.

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

[3] Bengio, Y., Courville, A., and Vincent, P. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence 35*, 8 (2013), 1798–1828.

[4] Borg, I., and Groenen, P. J. *Modern multidimensional scaling: Theory and applications*. Springer Science & Business Media, 2005.

[5] Chandola, V., Banerjee, A., and Kumar, V. Anomaly detection: A survey. *ACM computing surveys (CSUR) 41*, 3 (2009), 15.

[6] Chung, F. R. *Spectral graph theory*. No. 92. American Mathematical Soc., 1997.

[7] Dong, Y., Chawla, N. V., and Swami, A. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of KDD* (2017).

[8] Filotti, I., Miller, G. L., and Reif, J. On determining the genus of a graph in o (v o (g)) steps (preliminary report). In *Proceedings of the eleventh annual ACM symposium on Theory of computing* (1979), ACM, pp. 27–37.

[9] Getoor, L., and Diehl, C. P. Link mining: a survey. *Acm Sigkdd Explorations Newsletter 7*, 2 (2005), 3–12.

[10] Grover, A., and Leskovec, J. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining* (2016), ACM, pp. 855–864.

[11] Han, J. Mining heterogeneous information networks by exploring the power of links. In *Discovery Science* (2009), Springer, pp. 13–30.

[12] Huang, Z., and Mamoulis, N. Heterogeneous information network embedding for meta path based proximity. *arXiv preprint arXiv:1701.05291* (2017).

[13] Iwata, T., Yamada, T., and Ueda, N. Probabilistic latent semantic visualization: topic model for visualizing documents. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining* (2008), ACM, pp. 363–371.

[14] Jacob, Y., Denoyer, L., and Gallinari, P. Learning latent representations of nodes for classifying in heterogeneous social networks. In *Proceedings of the 7th ACM international conference on Web search and data mining* (2014), ACM, pp. 373–382.

[15] Jensen, D., and Goldberg, H. Aaai fall symposium on ai and link analysis, 1998.

[16] Ji, M., Han, J., and Danilevsky, M. Ranking-based classification of heterogeneous information networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining* (2011), ACM, pp. 1298–1306.

[17] Leroy, V., Cambazoglu, B. B., and Bonchi, F. Cold start link prediction. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining* (2010), ACM, pp. 393–402.

[18] Lewis, T. G. *Network science: Theory and applications*. John Wiley & Sons, 2011.

[19] Liben-Nowell, D., and Kleinberg, J. The link-prediction problem for social networks. *journal of the Association for Information Science and Technology 58*, 7 (2007), 1019–1031.

[20] Maaten, L. v. d., and Hinton, G. Visualizing data using t-sne. *Journal of Machine Learning Research 9*, Nov (2008), 2579–2605.

[21] Mei, Q., Cai, D., Zhang, D., and Zhai, C. Topic modeling with network regularization. In *Proceedings of the 17th international conference on World Wide Web* (2008), ACM, pp. 101–110.

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

[23] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (2013), pp. 3111–3119.

[24] Morin, F., and Bengio, Y. Hierarchical probabilistic neural network language model. In *Aistats* (2005), vol. 5, Citeseer, pp. 246–252.

[25] Ng, A. Y., Jordan, M. I., and Weiss, Y. On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems* (2002), pp. 849–856.

[26] Otte, E., and Rousseau, R. Social network analysis: a powerful strategy, also for the information sciences. *Journal of information Science 28*, 6 (2002), 441–453.

[27] Perozzi, B., Al-Rfou, R., and Skiena, S. Deepwalk: Online learning of social representations. In *Proceedings of KDD* (2014), pp. 701–710.

[28] Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. Collective classification in network data. *AI magazine 29*, 3 (2008), 93.

[29] Shi, C., Li, Y., Zhang, J., Sun, Y., and Philip, S. Y. A survey of heterogeneous information network analysis. *IEEE Transactions on Knowledge and Data Engineering 29*, 1 (2017), 17–37.

[30] Sun, Y., and Han, J. Mining heterogeneous information networks: a structural analysis approach. *ACM SIGKDD Explorations Newsletter 14*, 2 (2013), 20–28.

[31] Sun, Y., Han, J., Yan, X., Yu, P. S., and Wu, T. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *Proceedings of the VLDB Endowment 4*, 11 (2011), 992–1003.

[32] Sun, Y., Norick, B., Han, J., Yan, X., Yu, P. S., and Yu, X. Pathselclus: Integrating meta-path selection with user-guided object clustering in heterogeneous information networks. *ACM Transactions on Knowledge Discovery from Data (TKDD) 7*, 3 (2013), 11.

[33] Tang, J., Qu, M., and Mei, Q. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of KDD* (2015), pp. 1165–1174.

[34] Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., and Mei, Q. Line: Large-scale information network embedding. In *Proceedings of WWW* (2015), pp. 1067–1077.

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

[36] Tang, L., Liu, H., Zhang, J., and Nazeri, Z. Community evolution in dynamic multi-mode networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining* (2008), ACM, pp. 677–685.

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

[38] Tu, C., Liu, Z., and Sun, M. Inferring correspondences from multiple sources for microblog user tags. In *Chinese National Conference on Social Media Processing* (2014), Springer, pp. 1–12.

[39] Von Luxburg, U. A tutorial on spectral clustering. *Statistics and computing 17*, 4 (2007), 395–416.

[40] Washio, T., and Motoda, H. State of the art of graph-based data mining. *Acm Sigkdd Explorations Newsletter 5*, 1 (2003), 59–68.

[41] Wasserman, S., and Faust, K. *Social network analysis: Methods and applications*, vol. 8. Cambridge university press, 1994.

[42] Wassermann, S., and Faust, K. Social network analysis: Methods and applications, 1994.

[43] Yang, C., Liu, Z., Zhao, D., Sun, M., and Chang, E. Y. Network representation learning with rich text information. In *IJCAI* (2015), pp. 2111–2117.

Appendix A

Project Plan

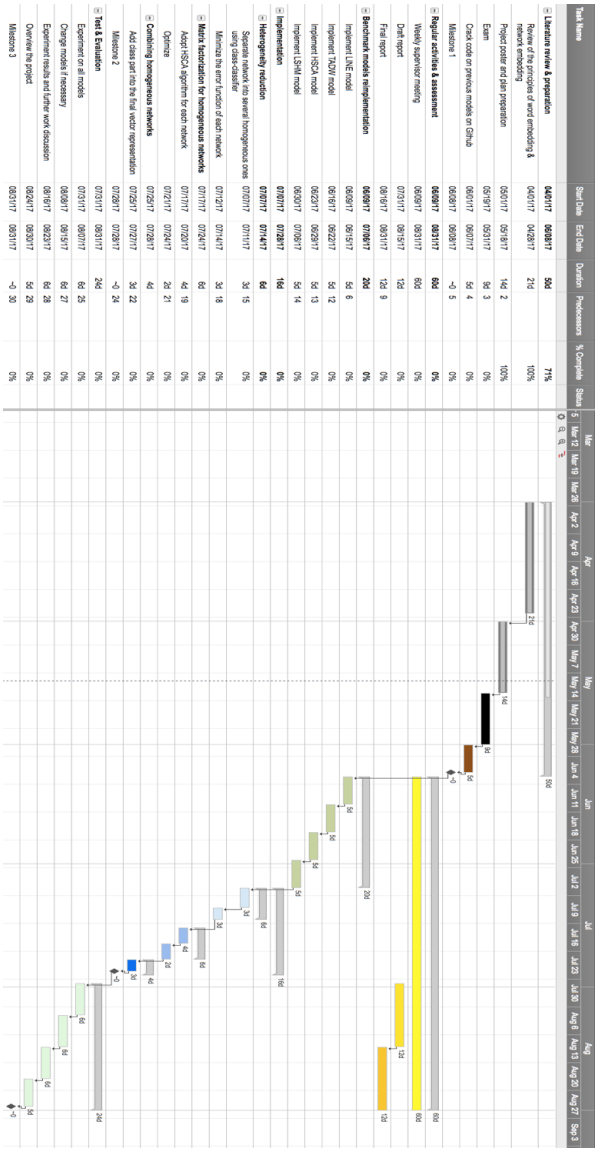


Figure A.1: Gantt Chart