UNIVERSITY OF TECHNOLOGY SYDNEY
CAI, SoS, FEIT

LEARNING AND REPRESENTING ATTRIBUTED GRAPHS

by

Ruiqi Hu

A Thesis Submitted
in Partial Fulfillment of the
Requirements for the Degree

Doctor of Philosophy

Dissertation Directed by
Dr. Guodong Long, Dr. Shirui Pan, and Prof. Chengqi Zhang

Sydney, Australia

2018

Certificate of Authorship/Originality

I certify that the work in this thesis has not been previously submitted for a degree nor has it been submitted as a part of the requirements for other degree except as fully acknowledged within the text.

I also certify that this thesis has been written by me. Any help that I have received in my research and in the preparation of the thesis itself has been fully acknowledged. In addition, I certify that all information sources and literature used are quoted in the thesis.

This research is supported by the Australian Government Research Training Program.

Ruiqi Hu

Production Note:
Signature removed
prior to publication.

ABSTRACT

LEARNING AND REPRESENTING ATTRIBUTED GRAPHS

by

Ruiqi Hu

Information graphs are ubiquitous in many areas, such as medicine, social media and academic engines, and each node in the graph comes with various attributes. For example, in a academic citation graph, we can take each paper a node, then the author(s) and title of each paper can be extracted as the attributes of the node. Moreover, papers, authors as well as venues can be taken as different sources of nodes in one information graph. By doing so, we have got a heterogeneous information graph with more than one sources of nodes, attributes and links.

To implement these applications, such as identifying protein residues and social media marketing, graph representation of homogeneous information graphs has been widely researched and employed. This research, aims to embed and represent homogeneous nodes with low-dimensional and unified vectors, while preserving the contextual information between nodes, and, as a result, classical machine learning methods can be directly applied.

However, existing graph embedding algorithms are facing five major challenges: 1.the graph representation learning and node classification in graphs are separated into two steps, which may result in sub-optimal results because the node representation may not fit the classification model well; 2. existing ones are mostly shallow methods that can only capture the linear and simple relationships in the data; 3.ignoring the data distribution of the latent codes from the graphs, which often results in inferior embedding in real-world graph data; 4. unable to handle the heterogeneous and multi-relational information graph which is the major form that graph data existed in the real-world; and 5. unable to effectively discover functional groups

and understand the roles of detected groups.

To face the aforementioned challenges, the main research objective of the thesis is to study that how to more effectively embed the nodes of a graph into a compact space for the tasks which are most related to the real-world applications.

The main research objective has been studied from four coherently linked perspectives: (1) How to unify the traditional two-step embedding work-flow into one smooth embedding procedure to avoid the inconsistency between the embedding architecture and classifier; (2) How to learn a universal embedding for all sources of nodes in a graph, so one single embedding can be used to represent the entire heterogeneous information graph; (3) How to smoothly regularize the embedding with a certain distribution during the learning procedure for a more robust embedding;(4) How to automatically generate a human-understandable explanation of each cluster of nodes in the graph and applied the algorithm in the real business world.

Specifically, this thesis aims to tackle aforementioned challenges by conducting studies of graph ladder network to unifies both representation and classifier model learning into one framework; developing universal graph representation to represent different types of nodes in heterogeneous information graph in a continuous and common vector space; introducing generative adversarial scheme into graph domain to encode the topological structure and node content in a graph to a compact representation, on which a decoder is trained to reconstruct the graph structure under an adversarial training scheme and carrying out co-clustering on enterprise information graph for functional group discovery and understanding. All works in this thesis are validated with related tasks like graph classification, graph clustering, graph visualization and link prediction respectively.

Acknowledgements

Firstly, I would like to express my sincere gratitude to my superviors Dr. Guodong Long, Dr. Shirui Pan, and Prof. Chengqi Zhang for their continued support and guidance in my Ph.D. study and related projects, specifically for their patience, motivation, and immense knowledge. Their guidance helped me in the entire research and writing of this thesis. I could not have imagined having better supervisors or mentors for my Ph.D study.

A very special gratitude goes to Prof. Mingan Choct, who has provided life-changing support and opportunities. Words cannot express my emotions in this regard.

Thanks to all those who shared my sleepless nights in the laboratory. It was great spending time in the laboratory with all of you.

I am also grateful to all the following people, who have helped and supported me along the way: Dr. Jing Jiang, Prof. Li Xue, Dr. Tony W.T. Chan, Mr. David Xue, Dr. Xueping Peng, and the list goes on.

And finally, last but by no means least, I would like to thank my family: my father Mr. Wenming Hu, who always encourages me to try harder in the business world; my mother Prof. Shuqing Yang, who strongly believes that deep learning is not science; and my fiancée Qian Zhang; without her, it is impossible for me to just focus on what I am really interested in.

Thanks for all your encouragement!

Ruiqi Hu

Sydney, Australia, 2018.

List of Publications

Journal Papers

J-1. [TCYB] **Ruiqi Hu**, Shirui Pan, Sai-fu Fung, Guodong Long, Jing Jiang, and Chengqi Zhang. "Learning Graph Embedding with Adversarial Training Methods", IEEE Transactions on Cybernetics. (Under review) (ERA Rank **A*** journal).

J-2. [PR] **Ruiqi Hu**, Shirui Pan, Sai-fu Fung, Guodong Long. "Clustering Social Audiences in Business Information Networks", Pattern Recognition. (Under review) (ERA Rank **A*** journal).

J-3. [TNNLS] Chun Wang, **Ruiqi Hu**, Shirui Pan, Guodong Long and Jing Jiang. "Unsupervised Deep Neighbor-aware Embedding for Graph Clustering" IEEE Transactions on Neural Networks and Learning Systems. (Under review) (the first and second author contribute equally to this work) (ERA Rank **A*** journal)

Conference Papers

C-1. [IJCAI-19] **Ruiqi Hu**, Shirui Pan, , Guodong Long, Jing Jiang, Lina Yao and Chengqi Zhang. "Going Deep: Graph Convolutional U-Shape Networks for Semi-supervised Node Classification" 2019 International Joint Conference on Artificial Intelligence. (Under review) (CORE Rank **A*** conference)

C-2. [IJCAI-19] Chun Wang, Shirui Pan, **Ruiqi Hu**, Guodong Long, Jing Jiang and Chengqi Zhang. "DAEGC: Unsupervised Deep Attentional Embedding for Graph Clustering" 2019 International Joint Conference on Artificial Intelligence. (Under review) (CORE Rank **A*** conference)

C-3. [IJCAI-18] Shirui Pan, **Ruiqi Hu**, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. "Adversarially Regularized Graph Autoencoder." Interna-

tional Joint Conference on Artificial Intelligence, 2018 (accepted on 17 April 2018) (Shirui Pan and Ruiqi Hu contribute equally to this work) (ERA Rank **A***)

C-4. [CIKM-17] **Ruiqi Hu**, Shirui Pan, Jing Jiang, and Guodong Long. "Graph Ladder Networks for Network Classification." In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, pp. 2103-2106. ACM, 2017 (ERA Rank A).

C-5. [IJCNN-16] **Ruiqi Hu**, Shirui Pan, Guodong Long, Xingquan Zhu, Jing Jiang, and Chengqi Zhang. "Co-clustering enterprise social networks." In Neural Networks (IJCNN), 2016 International Joint Conference on, pp. 107-114. IEEE, 2016 (ERA Rank A).

C-6. [IJCNN-17] **Ruiqi Hu**, Celina Ping Yu, Sai-Fu Fung, Shirui Pan, Haishuai Wang, and Guodong Long. "Universal network representation for heterogeneous information networks." In Neural Networks (IJCNN), 2017 International Joint Conference on, pp. 388-395. IEEE, 2017 (ERA Rank A).

C-7. [IJCNN-19] Di Wu, Ruiqi Hu, Yu Zheng, Jing Jiang and Michael Blumenstein. “Feature-Dependent Graph Convolutional Autoencoders with Adversarial Training Methods” 2017 International Joint Conference on, (accepted on 08 March 2019) (Di Wu and Ruiqi Hu contribute equally to this work) (CORE Rank A conference)

Contents

List of Figures

Abbreviation

EM - Expectation Maximization

GCN - Graph Convolutional Network

NMF - Non-negative Matrix Factorization

CNN - Convolutional Neural Network

MLP - Multi-layer Perceptron

MHIG - Multi-relational Heterogeneous Information Graph

SGD - Stochastic Gradient Descent

SAC - Social Audience Cluster

BIG - Business Information Graph

K-NN - K-Nearest Neighbors

PPMI - Positive Point-wise Mutual Information

Nomenclature and Notation

Capital letters denote matrices.

Lower-case alphabets denote column vectors.

$(.)^T$ denotes the transpose operation.

I_n is the identity matrix of dimension $n \times n$.

0_n is the zero matrix of dimension $n \times n$.

$\mathbb{R}$, $\mathbb{R}^+$ denote the field of real numbers, and the set of positive reals, respectively.

Chapter 1

Introduction

1.1 Background

Graphs are essential tools used to capture complicated relationships among data, and graph models are useful for demonstrating information across many academic or industrial domain. In a variety of graph applications, including protein-protein interaction networks [1, 2, 3], social media [4, 5, 6], citation networks [7, 8], graph data analysis plays an important role in various data mining tasks, including node classification [9], link prediction [10], and node clustering [11]. However, the high computational complexity of, low parallelizability of, and inapplicability of machine learning methods to graph data have made these graph analytical tasks profoundly challenging [12]. Therefore, it is important to learn features of nodes in graphs, considering that actual topological structure and graph embedding have emerged as general approaches to these problems. The aim of graph embedding is to represent each node and its features into low-dimensional vectors while two similar nodes would have a shorter distance between two learned vectors. In this way, the information about each node and the contextual information of the graph are well preserved in the vectors. Figure 1.1 shows an example of information graph embedding which aims to encode each node into a low-dimensional vector, while preserving the majority of the graph information. The graphs can be categorized into two classes: homogeneous information graphs, which only considers a single type of node, and heterogeneous information graphs, which contains more than one type of node in the graph. In this thesis, I consider and research both types of information graphs.

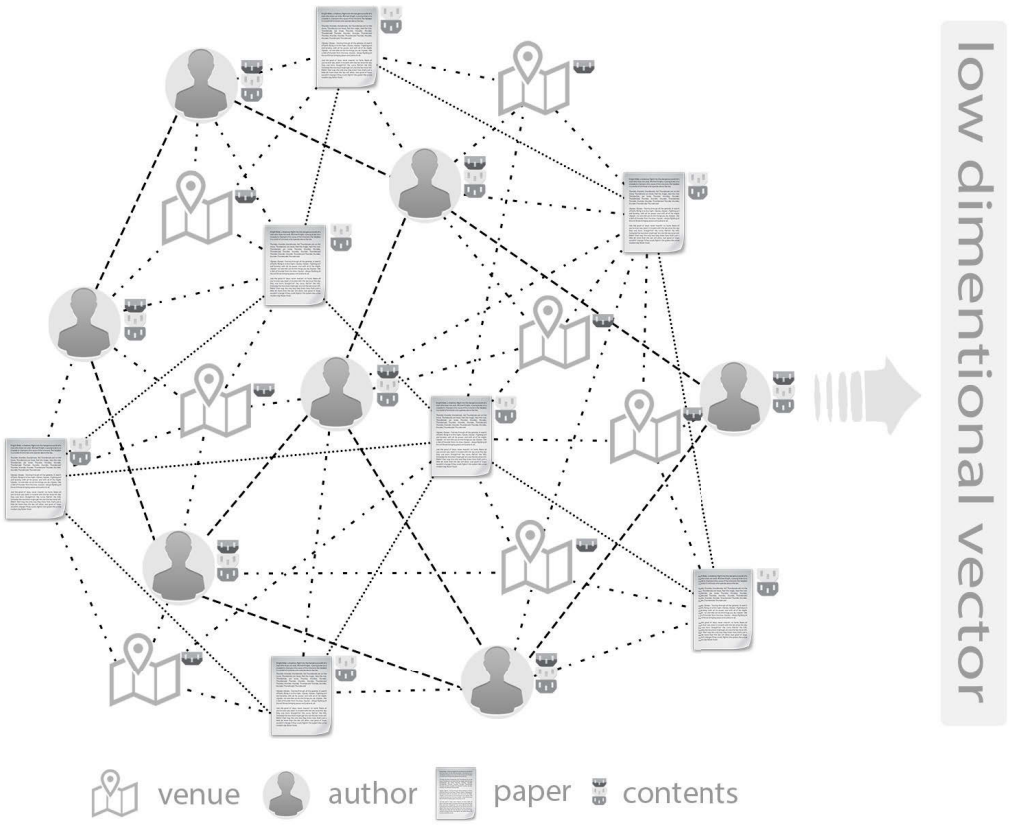

Figure 1.1 : A demonstration of graph embedding/graph representation.

Figure 1.1 gives an example of a heterogeneous information graph, which contains four types of interrelated nodes: the venue, author, paper, and contents of each paper. Our goal is to embed the graph into an universal graph representation for all types of nodes in the graph. The details will be elaborated in Chapter 5.

However, most existing embedding algorithms have the following five major limitations:

1. Existing algorithms separate the graph representation learning and node classification in the graphs into two steps, which may result in sub-optimal results because the node representation may not fit the classification model well and vice versa;

Many approaches to graph classification in recent years are representation-based

methods [13, 14]. They follow a two-step strategy: (1) learn a continuous, compact, and informative vector representation for each node in the network, and (2) train a simple classifier (such as a linear support vector machine) from the transformed vectors to predict the unlabeled nodes in the network. The key challenge of this approach lies in the first step, and the approach has motivated numerous studies, which will be explained in Chapter Literature Survey.

2. Most existing algorithms are superficial and can only capture linear and simple relationships in the data [15, 13];

Although some algorithms, such as NetPLSA [16] and TADW [14], are able to leverage both topological information and content information for graph representation, they assume that the linked nodes (documents) naturally share similar topic distributions. NetPLSA learns topic models and Text-Associated DeepWalk (TADW) performs matrix factorization for network representation. However, their assumption may not be true in real networks like enterprise social networks [6] or even in ordinary social networks. This assumption generally leads to sub-optimal experimental result. Furthermore, these approaches are only superficial as they are not able to learn deep and nonlinear representation for networks.

3. Probabilistic models like DeepWalk [13], node2vec [16], and LINE [17] attempt to learn graph embedding by extracting different patterns from graphs or by calculating matrix factorization-based algorithms such as GraRep [18] and HOPE [19]. M-NMF [20], however, pre-processes the graph structure into an adjacency matrix and Achieves embedding by decomposing the adjacency matrix or by deep learning, such as SDNE [21] and DNGR [22], which have been widely studied in the context of graph embedding. All of the above are unregularized embedding approaches and in practice they often learn a degenerate *identity* mapping where by the latent code space is free of any structure [17] and can often result poor representation in dealing

with real-world sparse and noisy graph data.

4. Existing algorithms have predominantly been designed for homogeneous information networks with all nodes of a network of the same type. However, the majority of real-world information networks are heterogeneous and multi-relational. The sources of nodes in these information networks vary, but are interrelated.

Examples of such graphs include DNA-protein interaction graphs [18] in medicine or complete publication graphs [19, 20, 21] in academic engines. The sources of nodes in these information graphs vary but are interrelated. For instance, one example of a DNA-protein interaction graph consists of two sources of nodes, DNA and proteins, and each source of homogeneous nodes is profiled by an independent graph structure based on their inner reactions; for example, a DNA graph structure and a protein graph structure. Furthermore, the interactions between the heterogeneous nodes (DNA and proteins) provide a more comprehensive, but complex, multi-relational heterogeneous graph, with different sources of nodes, graph structures and node contents. Publication graphs have the same characteristics: two different sources of nodes (papers and authors) with two independent graph expressions (paper citation graphs and author collaboration graphs) respectively according to their inner-reactions, while papers and authors are naturally connected by a heterogeneous graph because of their inherent publication correlations.

The obvious variety and complexity of multi-relational heterogeneous information graphs (MHIGs) limit existing graph representation methods in two ways: (1) They can only leverage one perspective on the data either the graph structure or the node content which simply abandons the integrity of the graph (2) the final learned representation can only represent one of the sources of information - either the DNA **or** the protein in DNA-protein interaction graphs; or the papers **or** the authors in publication graphs.

5. Existing topic discovery or community discovery methods cannot effectively discover functional groups and understand the roles of these groups.

Topology exploration algorithms, such as [22, 23, 24], only consider graph structures and do not consider any of the content information affiliated with each node. Early content-based algorithms employ approaches such as a topic model and bag-of-words, aiming to encode each content document into a vector without considering contextual information (i.e., the order of documents or the order of words), or sub-optimizing the representations. Multi-source captured algorithms like TriDNR [25] smoothly extract and process the information from the textual attributes of each node, topological information from the graph, and the label information to train a robust deep architecture for graph representation.

However, these approaches assume that the links and the content of nodes are highly consistent, which may be not true for some real-world graph data, because a user may be interested in a product, but the content of the user (i.e. node information) has no relevance to that connection. Moreover, two competitors may have very similar node content, but they do not share linkages, due to their rival business roles. Furthermore, these algorithms only detect communities or groups and are not capable of providing the common interests of the detected communities, which makes these algorithms infeasible in real-world applications.

1.2 Research Objectives

The main research objective of the thesis is to study that how to more effectively embed the nodes of a graph into a compact space for the tasks which are most related to the real-world applications.

The main research objective has been studied from four coherently linked perspectives: (1) How to unify the traditional two-step embedding work-flow into one

smooth embedding procedure to avoid the inconsistency between the embedding architecture and classifier; (2) How to learn a universal embedding for all sources of nodes in a graph, so one single embedding can be used to represent the entire heterogeneous information graph; (3) How to smoothly regularize the embedding with a certain distribution during the learning procedure for a more robust embedding; (4) How to automatically generate a human-understandable explaination of each cluster of nodes in the graph and applied the algorithm in the real business world.

Specifically, to achieve the research objective and validate the results, I conducted the research studies with following topics:

i. conduct studies of the unification of representation learning and graph classification into one single framework by exploiting both labeled and unlabeled nodes in a graph.

   Facing challenges 1 and 2 described in the background, I propose a novel approach, graph ladder networks (GLNs), which (1) unify both representation and classifier model learning into one framework; and (2) integrate both the structure and content information of a network by a convolution network architecture. To integrate both representation and classifier learning into one framework, I utilize a ladder network [26] that trains the model to simultaneously minimize the sum of supervised and unsupervised cost functions by backpropagation, without the need for layer-wise pre-training. To integrate both structure and content information for graph classification, I apply a convolution network architecture for networked data, which is motivated by the most recently developed graph convolutional network (GCN) [9] with a layer-wise propagation. Compared to existing representation-based node classification algorithms in networked data, GLN architecture offers a number of advantages: (1) joint representation and classification for networked data; (2)

deep and non-linear representation for networked data; and (3) effective and significantly improved results for graph node classification.

ii. conducting studies on a novel adversarial graph embedding framework for graph data that encodes the topological structure and node content in a graph to a compact representation, on which a decoder is trained to reconstruct the graph structure. Furthermore, the latent representation is enforced to match a prior distribution via an adversarial training scheme.

Facing challenge 3 described above, I developed a novel adversarial framework with two variants, namely *adversarially regularized graph autoencoder* (ARGA) and *adversarially regularized variational graph autoencoder* (ARVGA), for graph embedding. The theme of our framework is to not only minimize the reconstruction errors of the graph structure but also to enforce the latent codes to match a prior distribution. By exploiting both graph structure and node content with a graph convolutional network, our algorithms encodes the graph data in the latent space. With a decoder aiming to reconstruct the topological graph information, I further incorporate an adversarial training scheme to regularize the latent codes to learn a robust graph representation. The adversarial training module aims to discriminate if the latent codes are from a real prior distribution or from the graph encoder. The graph encoder learning and adversarial regularization learning are jointly optimized in a unified framework so that each can be beneficial to the other and finally lead to a better graph embedding. The experimental results on benchmark datasets demonstrate the superb performance of our algorithms on three unsupervised graph analytical tasks, namely link prediction, node clustering, and graph visualization.

iii. conducting studies on a universal graph representation approach (UGRA) that represents different types of nodes in heterogeneous information graphs in a

continuous and common vector space.

Facing challenge 4, I designed UGRA, which uses mutually enhanced neural network architectures to learn representations for all sources of nodes (DNA **and** proteins in DNA-protein interaction graphs/networks; papers **and** authors in publication graphs) for input heterogeneous graphs. From the perspective of graph structures, the UGRA jointly learns the relationships between homogeneous nodes and the connections between heterogeneous nodes by maximizing the probability of discovering neighboring nodes given a node in random walks. From the perspective of node content, the UGRA captures the correlations between nodes and content by exploiting the co-occurrence of word sequences given a node. By doing so, different sources of nodes in one heterogeneous graph are represented by a single universal representation. Given one node from any source, the universal representation can return all sources of nodes which are most related to the given one.

iv. conducting studies on an algorithm which carries out co-clustering on enterprise information graphs for functional group discovery and understanding.

Business information graphs involve diversified user clusters and rich content, and, have emerged as important platforms for enabling enterprise business intelligence and business decision-making in recent years. A key step in many business intelligence processes is to find social audience clusters (SACs) and assess the roles they play in a business graph. A SAC is defined as a group of users who share similar business interests or play similar roles in an organization or industry.

Facing challenge 5, I propose a novel data mining approach, CBIG, that co-clusters business information graphs to discover and understand SACs. The CBIG framework is based on co-factorization, which combines graph structures

and rich contextual information, including node-interaction and node-content correlations to discover SACs. Since SACs often overlap, and the data-driven nature of the processing makes the number of clusters difficult, CBIG uses an overlapping clustering paradigm and a hold-out strategy to discover the optimal number of groups given the underlying data.

In this case, all users from the enterprise information graph are processed in an automatically determined optimal number of groups (social audience clusters) based on their shared interests or roles played in the graph. Furthermore, CBIG will provide readable and understandable functions of each detected social audience cluster.

1.3 Content and Organization

This thesis is organized as follows:

- *Chapter 2*: This chapter presents a survey of classic and state-of-the-art graph representation or embedding-related works.
- *Chapter 3*: The graph ladder network for graph classification is explained in this chapter.
- *Chapter 4*: This chapter presents the adversarially regularized graph autoencoder for graph embedding.
- *Chapter 5*: This chapter discusses universal network representation for heterogeneous information graphs.
- *Chapter 6*: This chapter explains how I tailor a framework that not only discovers functional groups of enterprise social networks, but also understands the roles of these detected groups.

- *Chapter 7*: A brief summary of the thesis contents and its contributions to the field as well as recommendations for future research in this area are given in this final chapter.

Chapter 2

Literature Review

Graph embedding converts graph data into a low dimensional, compact, and continuous feature space. The key idea is to preserve the topological structure, vertex content, and other side information [27]. This new learning paradigm has shifted the tasks of seeking complex models for classification, clustering, visualization and link prediction to learning a robust representation of the graph data, so that any graph analytic task can be easily performed by employing simple traditional models (e.g., a linear SVM for the classification task), which has motivated a number of studies in this area [28, 29].

There are two main ways to category graph embedding algorithms. One is based on the core approach implemented in the algorithms and, then, graph embedding algorithms can be classified into three categories: probabilistic models, matrix factorization-based algorithms, and deep learning-based algorithms. Another way is according to information sources of graph data the algorithms focus on: network structure exploration algorithms, text analysis based algorithms and the algorithms leverage both topological information and textual information.

2.1 Core Approach based Categories

2.1.1 Probabilistic Models

Probabilistic models like DeepWalk [13], node2vec [30] and LINE [31] attempt to learn graph embedding by extracting different patterns from the graph. The captured patterns or walks include global structural equivalence, local neighborhood

connectivities, and first, second or higher order proximity. Probabilistic latent semantic analysis (PLSA) [32] focus on the content of nodes in a network which train topic model from represented content matrix (e.g. TFIDF matrix) by treating each content vector as a document. Then the node representation can be obtained from the topic distribution of nodes (documents) through statistic methods like expectation maximization algorithm (EM) or other approximate text analysis based models. Obviously, the main limitation of this approach is only use textual features without considering any contextual information like the order of words and topology of the network which could carry extra information as well as significantly enhance the performance of classification. Compared with classical methods such as Spectral Clustering [33], these graph embedding algorithms perform more effectively and are scalable to large graphs.

2.1.2 Matrix Factorization based Models

Matrix factorization-based algorithms, such as GraRep [34], HOPE [35], M-NMF [36] pre-process the graph structure into an adjacency matrix and get the embedding by decomposing the adjacency matrix. GNMF [37] is a classic nonnegative matrix factorization method that encodes geometrical information (a k-nearest neighbor graph) as a regularization term in the objective function. Recently it has been shown that many probabilistic algorithms are equivalent to matrix factorization approaches [38]. TADW [30] proves that DeepWalk [2] can be processed by factorizing an approximate probability matrix where one node randomly walks to another in certain steps, and incorporates with feature vectors by factorizing a word-association matrix. However, these algorithms are limited to process large scale data because matrix factorization is computationally expensive.

2.1.3 Deep Learning based Models

Deep learning approaches [39, 40], especially autoencoder-based methods, are also widely studied for graph embedding. SDNE [41] and DNGR [42] employ deep autoencoders to preserve the graph proximities and model positive pointwise mutual information (PPMI). The MGAE algorithm utilizes a marginalized single layer autoencoder to learn representation for clustering [11]. Variants of convolutional neural networks are also applied in the graph domain. Graph convolutional networks (GCN) [9] is a semi-supervised framework based on a variant of CNNs, which attempt to directly operate the graphs. Specifically, the GCN represents the graph structure and the interrelationship between node and feature with an adjacent matrix $\mathbf{A}$ and node-feature matrix $\mathbf{X}$. Hence, GCN can directly embed the graph structure with a spectral convolutional function $f(\mathbf{X}, \mathbf{A})$ for each layer and train the model on a supervised target for all labelled nodes. Because of spectral function $f(\bullet)$ on the adjacent matrix $\mathbf{A}$ of the graph, the model is able to distribute the gradient from the supervised cost and learn the embedding of both the labelled and unlabelled nodes. Although GCN is powerful with graph-structured data sets on semi-supervised tasks like node classification, variational graph autoencoder VGAE [43] extends it into unsupervised implements with outstanding performance on link prediction. VGAE implements the GCN into the variational autoencoder framework [44] by framing the encoder with graph convolutional layers and remolding the decoder with a simple inner production calculation layer. Taking the advantage of GCN layers, VGAE can naturally leverage the information of node features, which expressively muscle the predictive performance.

2.2 Graph Information Source based Categories

2.2.1 Topology Exploration Algorithms

Most previous efforts have concentrated on persevering either the graph structure or the node content. Graph structure analysis-based methods are more prevalent. Two very popular neural graph language models were proposed in [22, 23], where deep learning techniques revealed advantages in natural language processing applications. Illuminated by that, a DeepWalk algorithm was proposed in [24] that learns latent representations of vertices from a corpus of generated random walks in graph data. These algorithms only input graph structures, without considering any content information affiliated with each node.

2.2.2 Text Analysis based Algorithms

Early content-based algorithms employ approaches like, topic model and bag-of-words, to encode each content document into a vector without considering contextual information (i.e., the order of documents or the order of words), or sub-optimizing the representations. To acquire the contextual information, features are modeled using context-preserving algorithms [22, 45, 46, 47, 48] with a certain amount of consecutive words to represent a document. Obviously, an increasing number of content features for these algorithms will exponentially increase the training time and weaken performance.

Within neural graph architectures, alternative algorithms like skip-gram [22] input a certain window of consecutive words in the sentences of a document to learn the representation of words. Recurrent neural network-based models, say, long short term memory(LSTM)[49] and gated recurrent unit(GRU) [50] are able to capture long term dependencies using internal memory to process arbitrary sequences of text input.

2.2.3 Multi-Source Captured Algorithms

Whether graph structure exploration approaches or text analysis based ones with three obvious drawbacks identified in the mentioned algorithms: (1) only one source of the entire graph information has been leveraged, which lowers representation accuracy; (2) learned representations from these algorithms can only represent one type of graph data (i.e., representations from a citation graph only represent papers, but ignore authors and venues); and (3) the algorithms are unsupervised. No labeled data are even available to use, which misses the opportunity to enhance performance in tasks like classification.

Recently, some methods have attempted to simultaneously consider graph structures and node content information. TADW [14] proves that DeepWalk [24] can be processed by factorizing an approximate probability matrix where one node randomly walks to another in certain steps, and incorporates with feature vectors by factorizing a word-association matrix. However, this algorithm is not capable of processing large-scale data because matrix factorization is computationally expensive. It also ignores the contextual information in nodes. TriDNR [25] smoothly solves this problem by simultaneously learning the graph structure and node contents in a neural network architecture. However, TriDNR only leverages one source of the graph structure and the output only represents one type of graph data.

2.3 Summary

To summarize, aforementioned algorithms do not answer the undermentioned four questions to more effectively embed the nodes of a graph into a compact space for the tasks which are most related to the real-world applications: (1) How to unify the traditional two-step embedding work-flow into one smooth embedding procedure to avoid the inconsistency between the embedding architecture and classifier; (2) How to learn a universal embedding for all sources of nodes in a graph, so one single

embedding can be used to represent the entire heterogeneous information graph; (3) How to smoothly regularize the embedding with a certain distribution during the learning procedure for a more robust embedding; (4) How to automatically generate a human-understandable explanation of each cluster of nodes in the graph and applied the algorithm in the real business world.

To provide and validate the solutions to these questions step by step, I proposed four novel algorithms which are elaborated in Chapter 3, Chapter 4 and Chapter 5.

Chapter 3

Graph Ladder Networks for Node Classification

3.1 Motivations

Node classification is one of the most significant tasks to implement real-world graph applications, such as identifying protein residues, social media marketing, paper recommendation etc. Given a single network G, where a small set of nodes is labeled, node classification aims to classify the unlabeled nodes in G with content information into a set of predefined categories .

Many approaches for node classification in recent years are representation-based methods [13, 14]. They follow a two-step strategy: (1) learning a continuous, compact, and informative vector representation for each node in the network, (2) training a simple classifier (such as a linear support vector machine) from the transformed vectors to predict the unlabeled nodes in the network. The first step is the key challenge of representation-based approach, and it has motivated numerous studies in recent years.

Existing approaches like probabilistic latent semantic analysis (PLSA) [32] focus on the content of nodes in a network which train topic model from represented content matrix (e.g. TFIDF matrix) by treating each content vector as a document. Then the node representation can be obtained from the topic distribution of nodes (documents) through statistic methods like expectation maximization algorithm (EM) or other unobserved latent variables based models. Obviously, the main limitation of this approach is only use textual features without considering any contextual information like the order of words and topology of the network which could

carry extra information as well as significantly enhance the performance of classification. Taking enterprise social network (ESN) as an example, the topological structure of some ESNs could be almost self-classified and intriguing information like the personal influence and social complexity of one node can be extracted through the density and trajectory of the surrounding linkages [6].

The LINE [15] and DeepWalk [13] approaches are two classic algorithms for this problem. They explore the topological information of networks to learn the network representation. However, these approaches have limited the performance since they ignore the content information associated with each node. The content information can indicate the strong similarity and connection to others. Social media networks like Facebook, LinkedIn and Twitter, for instance, record the users' profile data such as educational background, location and professional trajectory which is obviously very useful information in terms of classification in a network.

In terms of using both topological information and content information for network representation, NetPLSA [16] and TADW [14] assume that the linked nodes (documents) naturally share similar topic distributions. NetPLSA learns topic models and TADW performs matrix factorization for network representation. However, their assumption may not be true in real networks like enterprise social networks [6] or even in ordinary social networks. This assumption always leads these algorithms to a sub-optimal experimental result. Furthermore, these approaches are only shallow ones that fail to learn deep and nonlinear representation for networks.

TADW incorporates DeepWalk with textual information vectors of node by factorizing an association matrix, the expensive computation of matrix factorization is very hard for these algorithms to handle large-scale data. To learn nonlinear representation, deep learning offers many effective solutions in numerous applications, including image classification and speech recognition. The ladder network [26] in

particular provides an elegant way to conduct semi-supervised classification. In addition to the supervised objective corresponding to the misclassification loss, the ladder network also adds an unsupervised objective corresponding to the reconstruction costs of a stack of denoising autoencoders, which has shown impressive results on image classification. However, the ladder network is only designed for general data such as images. How to apply ladder networks to networked or structure data has not been addressed in the literature.

However, it remains largely under-investigated for networked data analytics, which is mainly because it is very hard to define and learn the convolutions on networks. Until recently, CNN is extended to networked data [9, 51, 52]. Nevertheless, for network classification where both labeled and unlabeled nodes are available, they are still not effective. This is because they only capture the misclassification loss on the labeled nodes and ignore the unlabeled data completely [9].

To summarize, when dealing with network data, existing representation approaches suffer the following flaws: (1) they separate the network classification into two stages. The learned representation may not fit the classifier very well, and vice versa, which leads to sub-optimal classification results on unlabelled data; (2) most approaches are shallow methods that can only learn linear and simple representation for nodes, while deep approaches which are more suitable for representation learning (such as Convolution networks) are not used.

Recently, Tri-Party Deep Network Representation (TriDNR) [25] processes the label information in its neural network while simultaneously extracts topological structure and associated textual information. However, just like majority of network classification algorithms, it separately conduct the optimization of learning representation and training the independent classifiers (normally, the linear SVM [53]). The learnt representations will be fed into the trained classifiers. The seper-

ated and extra steps may cause sub-optimal performance.

To overcome the existing limitations of representation-based algorithms, this paper proposes a novel approach, graph ladder networks (GLN), which (1) unifies both representation and classifier model learning into one framework; and (2) integrates both the structure and content information of a network by a convolution network architecture. To integrate both representation and classifier learning into one framework, I utilize a ladder network [26] which trains the model to simultaneously minimize the sum of supervised and unsupervised cost functions by backpropagation, without the need for layer-wise pre-training. To integrate both structure and content information for network classification, I apply a convolution network architecture for networked data, which is motivated by the most recently developed graph convolutional network (GCN) [9] with a layer-wise propagation. Compared to existing representation-based node classification algorithms in networked data, our novel architecture enjoys a number of merits: (1) joint representation and classification for networked data; (2) deep and non-linear representation for networked data; and (3) effective and impressive classification results for network classification.

Our experimental results validate our design the demonstrate the effectiveness of our algorithm.

The main contributions can be summarized as follows:

i. I propose an effective node classification algorithm for networked data, which unifies the network representation and classification in a single framework.

ii. I advance the ladder network to a graph or structure domain, which integrates both structure and content information for semi-supervised learning.

iii. The experiment results show the superb performance of our algorithm over state-of-the-art algorithms.

The rest of the paper is organised as follows. Section II provides a brief background on network classification. I then define the network classification problem in convolutional network in Section III. In section IV, I mathematically propose our graph ladder Convolution Network algorithm and Section V presents the experimental study and results I obtained. Finally, Section VI concludes our work.

3.2 Most Related Works to Ladder Nets and Graph Node Classification

3.2.1 Traditional node classification

There are two types of traditional network classifications: one is taking multi networks as input and aims to precisely put the networks sharing the similar characteristics into the same group; another one concentrates on analysing the vertexes in one network and associated contextual information and classifies these vertexes with certain numbers of groups.

Most previous efforts have concentrated on persevering either the topology structure or the node content. Net structure analysis-based methods are more prevalent. Early content-based algorithms employ approaches like, topic model and bag-of-words, to encode each content document into a vector without considering contextual information (i.e., the order of documents or the order of words), or suboptimizing the representations. To acquire the contextual information, features are modelled using context-preserving algorithms [47, 45] with a certain amount of consecutive words to represent a document.

3.2.2 Network Representation

Recently, some methods have attempted to simultaneously consider network structures and node content information. TADW [14] proves that DeepWalk [13] can be processed by factorizing an approximate probability matrix where one node

randomly walks to another in certain steps, and incorporates with feature vectors by factorizing a word-association matrix.

3.2.3 Deep Learning for network data

Ladder Network

The Ladder Network with unsupervised learning tasks was proposed in 2015 By Valpola [54], which is an autoencoder that skips the connections between the encoder and decoder and denoises the representations at every level of the network. Semi-supervised learning with Ladder network was published by Rasmus [26] based on the Valpola's work, which extends the previous model for both supervision and unsupervision implements. Semi-supervised Ladder Network is structured with a fully connected Multi Layered Perceptron(MLP) with employing rectifier for every unit as the encoder part. Batch normalization to each preactivation for improving convergence and prevent the denoising cost from encouraging the trivial solution.

Graph Convolution Network

In Thomas's work [9], the network structure is directly encoded using a neural network model and train on a supervised target for all nodes with labels, thereby avoiding explicit network-based regularization in the loss function. Conditioning $f(\bullet)$ on the adjacency matrix of the network will allow the model to distribute gradient information from the supervised loss and will enable it to learn representations of nodes both with and without labels.

Either network structure exploration approaches or text analysis based ones have three obvious drawbacks identified in the mentioned algorithms: (1) only one source of the entire networked information has been leveraged, which lowers representation accuracy; (2) separately conduct the optimization of learning representation and training the independent classifiers and the extra steps may cause sub-optimal

performance.; and (3) the algorithms are unsupervised. No labelled data are even available to use, which misses the opportunity to enhance performance in tasks like classification.

3.3 Problem Definition

A network is defined as $G = \{V, E, X, S\}$, where $V = \{v_i\}_{i=1,...,N}$ consists of a set of nodes and $e_{i,j} =< v_i, v_j >\in E$ indicates an edge encoding the edge relationship between the nodes. $x_i \in X_i$ is the textual feature associated with each node v_i and $S = \mathcal{L} \cup \mathcal{U}$ is the label information in the network, where $\mathcal{L} = \{v_i\}_{i=1,\cdots,L}$ denotes labeled nodes and $\mathcal{U} = \{v_i\}_{i=L+1,\cdots,N}$ denotes unlabeled ones. Our **aim** is to learn a neural network model from G to predict the class labels of the unlabeled nodes of $\mathcal{U}$ in G.

3.4 Graph Ladder Networks

In this paper, I proposed a graph ladder networks (GLN) model for networked data, which unifies the representation learning and classification into a single framework. The proposed graph ladder networks model takes the advantage of ladder networks to simultaneously minimize the sum of the supervised and unsupervised cost function. To handle networked data with both structure and content information, our GLN further employs the convolutional operation for networked data.

3.4.1 Overall Framework

Fig 3.1 shows the architecture of our graph ladder networks.The graph convolutional network is employed as the building block of each layer in our archeticture. Two feedforward paths, $(x \rightarrow z^{(1)} \rightarrow z^{(2)} \rightarrow y)$ and $(x \rightarrow \widetilde{z}^{(1)} \rightarrow \widetilde{z}^{(2)} \rightarrow \widetilde{y})$ share the mappings function $f(\bullet)$. The decoder $(\widetilde{z}^{(l)} \rightarrow \hat{z}^{(l)} \rightarrow \hat{x})$ aims to minimize the gap between $\hat{z}^{(l)}$ and $z^{(l)}$ through the denoising functions $g^{(l)}$ and cost functions $C_d^{(l)}$

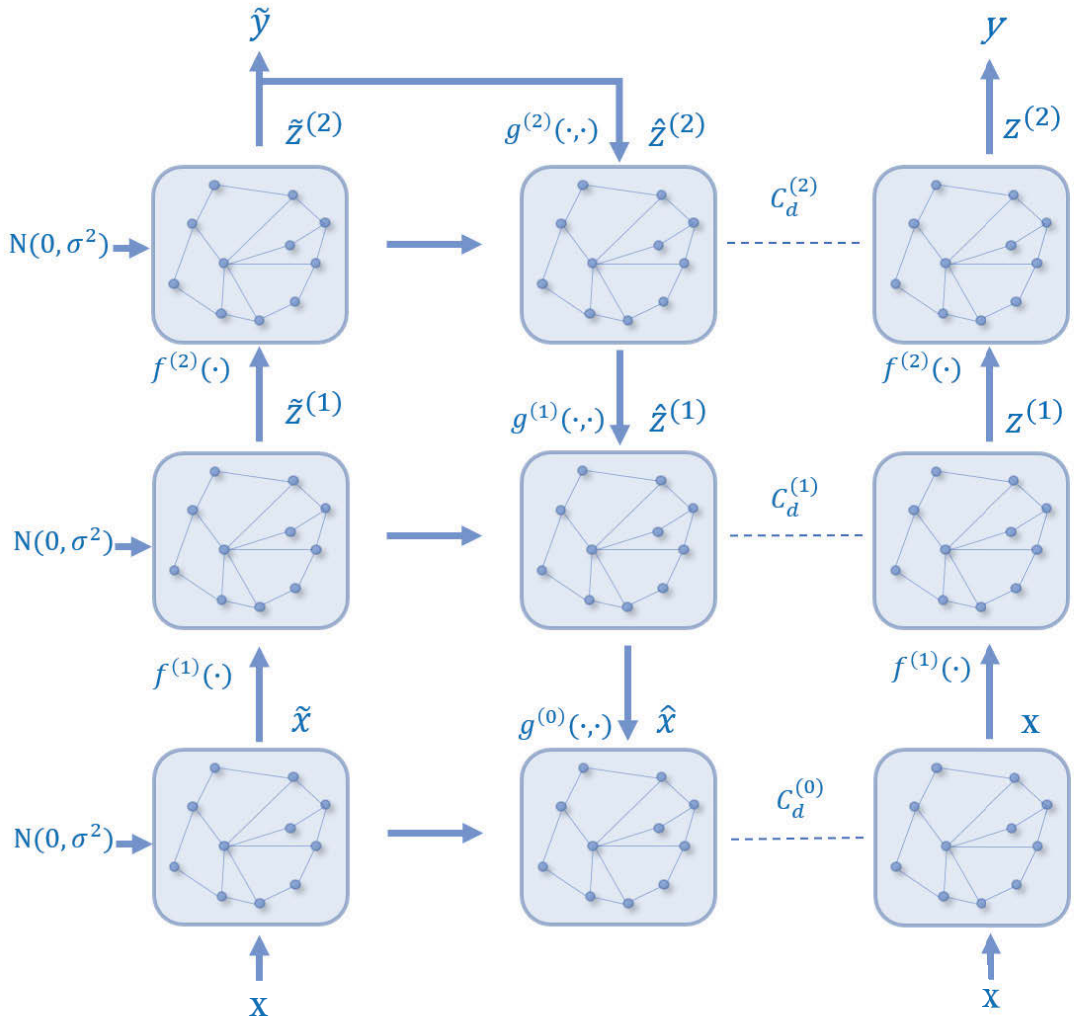

Figure 3.1 : A demonstration for a two-hidden-layer graph ladder network.

on each layer.The output $\widetilde{y}$ of the encoder can also be trained to match available labels $t(n)$. Given a network $G = \{V, E, X, S\}$ with L labeled nodes and $N - L$ unlabeled nodes, the objective is to learn a function that models $P(y|x)$ by using both the labeled nodes and a large quantity of unlabeled nodes. In our graph ladder networks, this function is a deep denoising autoencoder in which noise is injected into all hidden layers and the objective function is a weighted sum of the supervised Cross Entropy cost on the top of the encoder and the unsupervised Square Error costs at each layer of the decoder. Since all layers are corrupted by noise, another encoder path with shared parameters is responsible for providing the clean reconstruction targets.

In traditional ladder networks [26], the building block of each layer is a fully connected neural network. This cannot be applied directly in networked data, because networked data has both link and content information. To handle this problem, I employ the most recently developed graph convolution network (GCN) [9], which provides an natural way to integrates both structure and content information in the spectral domain.

3.4.2 Graph Convolutional Network

The GCN extends the operation of *convolution* to networked data in the spectral domain, and it is our building block for each layer in the graph ladder networks (GLN). Given a network $G = \{V, E, X, S\}$, the edge information can be represented as an adjacency matrix A, where $A_{i,j} = 1$ if $e_{i,j} \in E$. Then the GCN takes the content information X and adjacency matrix A as input, and learns a layer-wise transformation by a spectral convolution function $f(Z^{(l)}, A)$:

$$Z^{(l+1)} = f(Z^{(l)}, A). \tag{3.1}$$

Here, $Z^l \in R^{n \times m}$ (n nodes and m features) is the input for convolution, and $Z^{(l+1)}$ is the output after convolution. I will have $Z^0 = X$ for the problem. If $f(Z^{(l)}, A)$ is

well defined, I can build arbitrary deep convolutional neural networks efficiently.

In GCN, the function $f(Z^{(l)}, A)$ is defined as:

$$f(Z^{(l)}, A) = \sigma(\widetilde{D}^{-\frac{1}{2}}\widetilde{A}\widetilde{D}^{-\frac{1}{2}}Z^{(l)}W^{(l)}), \tag{3.2}$$

where $\widetilde{A} = A + I_N$ and $\widetilde{D}_{ii} = \sum_j \widetilde{A}_{ij}$. σ is an activation function such as $\text{Relu}(t) = \max(0, t)$ or $\text{sigmoid}(t) = \frac{1}{1+e^t}$. W^l is a matrix of filter parameters we need to learn in the neural network.

It is worth noting that GCN is proposed in [9], however, this method only considers the supervised cost hence it does not effectively exploit the unlabeled data. In our algorithm, I extend it into a more elegant framework, graph ladder networks, to exploit both labeled and unlabeled data. The whole network will be directly fed into a neural network model with extracting the matrix of node feature and the adjacent matrix of self-connected vertexes and the loss function is defined by the sum of the supervised loss with network Laplacian regularization loss [55, 56]:

$$\mathbb{L} = \mathbb{L}_o + \lambda\mathbb{L}_{reg} \tag{3.3}$$

where

$$\mathbb{L}_{reg} = \sum_{i,j} A_{i,j} \parallel f(X_i) - f(X_j) \parallel^2 = f(X)^\top \triangle f(X). \tag{3.4}$$

here X is the matrix of node feature vectors X_i and λ is a weighting parameter. $\triangle = D - A$ indicates the unregulated network Laplacian of an indirect network $\mathbb{G} = (\mathbb{V}, \mathbb{E})$ where node $v_i \in \mathbb{V}$, edges $(v_i, v_j) \in \mathbb{E}$. $A \in \mathbb{R}^{N\times N}$ is an adjacency matrix and $D_{ij} = \sum_j A_{ij}$. The propagation rule for a multi-layer Convolutional network can be defined with below equation:

$$H^{l+1} = \sigma(\check{D}^{-\frac{1}{2}}\check{A}\check{D}^{-\frac{1}{2}}H^lW^l) \tag{3.5}$$

where $\check{A} = A + I_N$ denotes the self-connected undirected network $\mathbb{G}$ and I_N is the identity matrix. W^l is a trainable layer weight matrix and $\check{D}_{ij} = \sum_j \check{A}_{ij}$.

The activation function, $\sigma(\bullet)$, can be an analytic function like rectified linear unit (ReLU). $H^{(l)} \in \mathbb{R}^{N \times D}$ is the activation of l^{th} hidden layer.

3.4.3 Graph Ladder Networks

As demonstrated in Figure 1, the general framework of GLN consists of one clean encoder, one corrupted encoder and one decoder:

$$x, z^{(1)}, z^{(2)}, ..., z^{(T)}, y = \text{Encoder}_{clean}(x), \tag{3.6}$$

$$\widetilde{x}, \widetilde{z}^{(1)}, \widetilde{z}^{(2)}, ..., \widetilde{z}^{(T)}, \widetilde{y} = \text{Encoder}_{corrupted}(x), \tag{3.7}$$

$$\hat{x}, \hat{z}^{(1)}, \hat{z}^{(2)}, ..., \hat{z}^{(T)} = \text{Decoder}(\widetilde{z}^{(1)}, ..., \widetilde{z}^{(T)}). \tag{3.8}$$

where x, y, and $\widetilde{y}$ are the input, clean output and corrupted output respectively, while $z^{(l)}$, $\widetilde{z}^{(l)}$ and $\hat{z}^{(l)}$ ($l \in \{1, \cdots, T\}$) indicate the clean hidden representation, corrupted hidden representation and the representation from decoder at layer l. The objective function combines both supervised cost and unsupervised cost on the networked data.

$$\begin{aligned} \mathcal{C} = & -\frac{1}{L} \sum_{n=1}^{L} logP(\widetilde{y} = t(n) \mid x(n)) \\ & + \sum_{l=0}^{T} \frac{\lambda_l}{N m_l} \sum_{n=1}^{N} \parallel z^{(l)}(n) - \hat{z}_{BN}^{(l)}(n) \parallel^2 . \end{aligned} \tag{3.9}$$

The first term is the supervised cost which is calculated with the average negative log probability of the noise output $\widetilde{y}$ targeting available labels $t(n)$ with the inputs $x(n)$, while the second, unsupervised denoising cost function, is the sum of the reconstruction error of T layers. λ_l is the hyper-parameter which determines the significance of the denoising cost, m_l is the size of the layer and N is the number of training instances.

I integrate the GCN into our graph ladder networks. According to Eq. (3.2), we have

$$\widetilde{Z}_{pre}^{(l)} = \widetilde{D}^{-\frac{1}{2}} \widetilde{A} \widetilde{D}^{-\frac{1}{2}} \widetilde{H}^{(l-1)} W^{(l)}. \tag{3.10}$$

where $W^{(l)}$ is a weight matrix between layer $l-1$ and l. $\widetilde{H}^{(l-1)}$ is the activation at the layer $l-1$. Then I perform the batch normalization on $\widetilde{Z}_{pre}^{(l)}$. Supposing $\widetilde{z}_{pre}^{(l)}$ is one row in $\widetilde{Z}_{pre}^{(l)}$ (i.e., the hidden representation for one node), I will normalize $\widetilde{z}_{pre}^{(l)}$ in the mini-batch:

$$\mu^{(l)} = \text{mean}(\widetilde{z}_{pre}^{(l)}), \tag{3.11}$$

$$\sigma^{(l)} = \text{stdv}(\widetilde{z}_{pre}^{(l)}), \tag{3.12}$$

$$\widetilde{z}^{(l)} = \frac{\widetilde{z}_{pre}^{(l)} - \mu^{(l)}}{\sigma^{(l)}} + \mathcal{N}(0, \sigma^2), \tag{3.13}$$

$$\widetilde{h}^{(l)} = \phi(\lambda^{(l)}(\widetilde{z}^{(l)} + \beta^{(l)})) \tag{3.14}$$

I compute pre-activation $\widetilde{z}^{(l)}$ by adding Gaussian noise. $\lambda^{(l)}$ and $\beta^{(l)}$ are adjusting parameters for the activation function $\phi(\cdot)$. It is easy to get a clean encoder by removing the Gaussian noise $\mathcal{N}(0, \sigma^2)$. For the decoder, I combine the layer $\hat{z}^{(l+1)}$ and the corrupted $\widetilde{z}^{(l)}$ to calculate the reconstruction $\hat{z}^{(l)}$ with the following equations:

$$U_{pre}^{(l+1)} = \widetilde{D}^{-\frac{1}{2}}\widetilde{A}\widetilde{D}^{-\frac{1}{2}}\hat{Z}^{(l+1)}Q^{(l)}.$$

Then for each row $u_{pre}^{(l+1)}$ in $U_{pre}^{(l+1)}$, I perform batch normalization as follows:

$$\mu^{(l+1)} = \text{mean}(u_{pre}^{(l+1)}), \tag{3.15}$$

$$\sigma^{(l+1)} = \text{stdv}(u_{pre}^{(l+1)}), \tag{3.16}$$

$$u^{(l+1)} = \frac{u_{pre}^{(l+1)} - \mu^{(l+1)}}{\sigma^{(l+1)}}, \tag{3.17}$$

$$\hat{z}^{(l)} = g(\widetilde{z}^{(l)}, u^{(l+1)}). \tag{3.18}$$

where $Q^{(l)}$ is a weight matrix between layer $l+1$ and l. The combination function is computed with the following equation:

$$g(\widetilde{z}^{(l)}, u^{(l+1)}) = (\widetilde{z}^{(l)} - \omega(u^{(l+1)}))\psi(u^{(l+1)}) + \omega(u^{(l+1)}) \tag{3.19}$$

where ω and ψ are different weighting parameters. While taking a scalar view, the batch normalized latent variable of decoder $\check{z}_{BN}$ can be computed with:

$$\check{z}_{BN} = \frac{\check{z} - \mu}{\sigma} \tag{3.20}$$

where μ and σ are the batch mean and batch standard of the projection Z_{pre}.

Two encoders are generated: one clean forward pass and one corrupted forward pass, which generated clean $z^{(l)}$ and $h^{(l)}$ and corrupted $\hat{z^{(l)}}$ and $\hat{h^{(l)}}$ respectively. Gaussian noise n is added to inputs and batch normalization in corruption part:

$$\hat{x} = \hat{h}_{(0)} = x + n_{(0)} \tag{3.21}$$

$$\hat{z}^{(l)}_{pre} = AW^{(l)}\hat{H}^{l-1}, \tag{3.22}$$

$$\hat{z}^{(l)} = N_B(\hat{z}^{(l)}_{pre} + n^l), \tag{3.23}$$

$$\hat{h}^{(l)} = \phi(\tau^{(l)}(\hat{z}^{(l)} + \beta^{(l)})). \tag{3.24}$$

The value $\hat{z}^{(l)}_{pre}$ is utilized in the decoder cost function C_u. $H^{(l)} \in \mathbb{R}^{N\times D}$ is the activation of l^{th} hidden layer. Backpropagation approach is applied to train the parameters $\beta^{(l)}$, $W^{(l)}$ and $\tau^{(l)}$ for optimized the final cost equation 3.9.

3.4.4 Convolutional Ladder

The nural network is a fully connected MLP which breaks the connections between the two encoders and decoder while denoises the representation at each level. The convergence is iteratively improved through batch normalization. Specifically, batch normalization in Convolutional Ladder is implemented by following formulations:

$$z^{(l)} = N_B(AW^{(l)}h^{(l-1)}) \tag{3.25}$$

$$h^{(l)} = \phi(\tau^{(l)}(z^{(l)} + \beta^{(l)})), \tag{3.26}$$

where l is the number of layers and there are layers l of latent variables $z^{(l)}$. $h^{(0)} =$ input x and $N_B(x_i) = (x_i - \hat{\mu_{x_i}})/\hat{\sigma_{x_i}}$ indicates a batch normalization, where $\hat{\mu_{x_i}}$ and $\hat{\sigma_{x_i}}$ represents the estimates calculated from the minibatch. $\phi(\cdot)$ is the rectified linear unit (ReLU) as the activation function, while $\tau^{(l)}$ and $\beta^{(l)}$ are its parameters varied during the training process.

Encoders

Two encoders are generated: one clean forward pass and one corrupted forward pass, which generate clean $z^{(l)}$ and $h^{(l)}$ and corrupted $\check{z}^{(l)}$ and $\check{h}^{(l)}$ respectively. Gaussian noise n is added to inputs and batch normalization in corruption part:

$$\check{x} = \check{h}_{(0)} = x + n_{(0)} \tag{3.27}$$

$$\check{z}^{(l)}_{pre} = AW^{(l)}\check{H}^{l-1}, \tag{3.28}$$

$$\check{z}^{(l)} = N_B(\check{z}^{(l)}_{pre} + n^l), \tag{3.29}$$

$$\check{h}^{(l)} = \phi(\tau^{(l)}(\check{z}^{(l)} + \beta^{(l)})). \tag{3.30}$$

The value $\check{z}^{(l)}_{pre}$ will be utilized in the decoder cost function and the supervised cost C_c can be calculated with the average negative log probability, given athe inputs $x(n)$:

$$C_s = -\frac{1}{N}\sum_{n=1}^{N} logP(\check{y} = t(n) \mid x(n)), \tag{3.31}$$

where $\check{y}$ is the noise output which matchs the target $t(n)$.

Decoder

In decoder, based on the assumption of the latent variables are conditionaly independent on the latent variables of the layer above. The distribution of latent variables z^l can be represented with conditionally independent Gaussian distributions:

3.5 Experimental Study

3.5.1 Dataset and Experiment Setting

Citeseer [14] is a popular and public publication dataset, presented by the citation links between papers. Each paper is presented by a binary vector indicating

Table 3.1 : Classification Results on Citeseer Dataset

Training Ratio	Deep Walk	PLSA	Text Features	Naive Combination	NetPLSA	TADW	Ladder Network	GLN
10%	0.524	54.1	0.583	0.61	0.587	0.706	0. 664	**0.727**
20%	0.547	58.3	0.664	0.667	0.616	0.719	0.707	**0.742**
30%	0.56	60.9	0.692	0.691	0.633	0.733	0.726	**0.759**
40%	0.565	62.1	0.712	0.708	0.640	0.737	0.739	**0.760**
50%	0.573	62.6	0.722	0.720	0.647	0.742	0.724	**0.763**

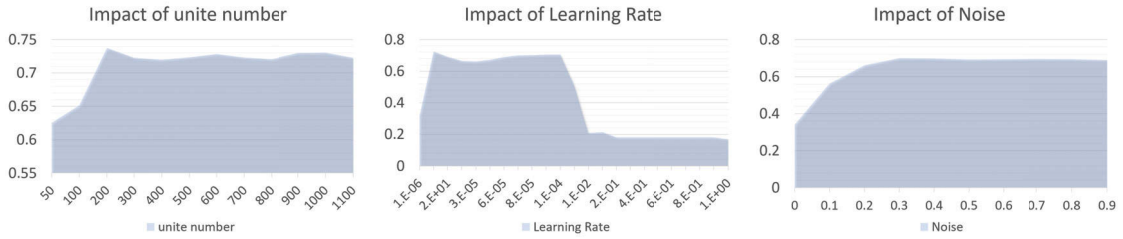

Figure 3.2 : Parameter study. From left to right, the impact on classification accuracy when varying the number of units, learning rate and noise.

the presence of stop-words-removed corresponding words. The dataset consists of 3,312 papers with 4,732 links, with the papers belonging to six fields.

GLN Settings The neural network is established with 500 units in the hidden layer while the decay starts from the 50th iteration. I also added 0.3 noise in the training process. Through experimentation, I found that the learning rate and the number of iterations are very effective for the final results. I aim to use the small iteration number for the best results. In practice, I iterated 300 times with 0.0001 learning rate for all experiments.

3.5.2 Baselines

GLN was compared to the following seven algorithms:

DeepWalk [13] learns representations of vertices in a network.

TADW [14] applies a matrix factorization based framework to merge text vectors into network representation processing.

PLSA [32] aims to estimate the topic distribution in documents and the word distribution in topics.

Text Features [14] applies SVM on the text features.

Naive Combination [14] merges the text feature into DeepWalk.

NetPLSA [16] consider both textual and topological information for representation.

Ladder Networks [26]: It is a network representation algorithm that exploit the text information only.

3.5.3 Classification Results

I conducted paper node classification and employed *Accuracy* as a metric, which is the percentage of correctly prediction on the unlabeled dataset, to evaluate the performance of GLN and all baselines. I varied the percentage of the random nodes with labels from 10% to 50% and the rest were unlabeled. A SVM classifier was applied to perform the classification for all baselines except the Ladder Network algorithm. I varied the training ratio from 10% to 50% for the linear SVM and GLN. The training and test set are randomly selected for each trial. I repeated each trial 15 times and reported the average accuracy score. The detailed results are recorded in Table 3.1. The GLN performed best among all comparing algorithms. Taking 10% training data as an example, we can see that GLN achieved around a 34.9% increase in accuracy compared to DeepWalk, 23.5% compared to PLSA,

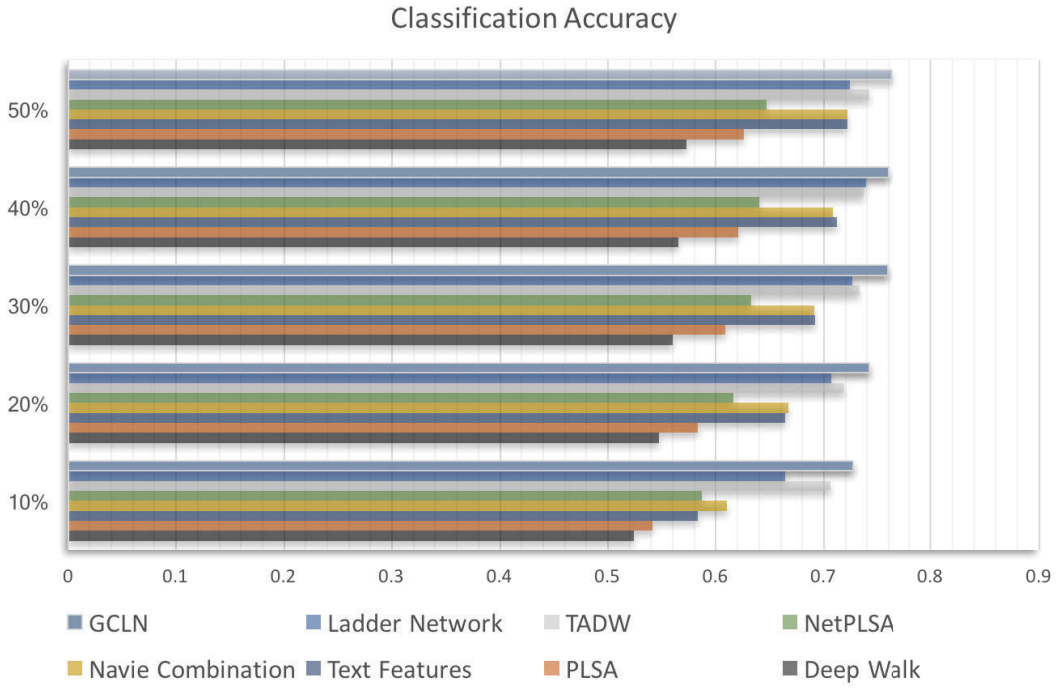

Figure 3.3 : The accuracy results of eight algorithms on dataset Citeseer

21.3% compared to Text Features, 15.9% compared to Naive Combination, 20.4% compared to NetPLSA and 6.5% compared to Ladder Networks.

3.5.4 Parameter Study

I varied the noise rate from 0.0 to 0.9 and fixed the other parameters to observe noise impact on the node classification accuracy. As is shown in Figure 3.2 , the positive impact on accuracy is significant when I slightly added noise from 0 to 0.3. But the difference was extremely small when I continued to add noise.

The value range of the learning rate is large, but it significantly impacts the final results. So, I varied the learning rate with a large value range and tried to find its pattern. Specifically, the learning rate was set from 0.00001 to 0.00009 and from 0.1 to 0.9. I also set it with 0.000001, 0.0001, 0.001 and 0.01 for rough checks. The accuracy increased dramatically when the learning rate is in the range of 0.00001

and 0.0001. Otherwise, the accuracy could drop down. As for the hidden layer, The number of units varied from 50 to 1100. The accuracy sharply increased when I added a number of units from 50 to 200 and the accuracy seems peaked when each layer has 200 unites. The curve behind 200 slightly fluctuated but overall the accuracy at this period is at a similar level.

3.6 Summary

In this paper, I proposed a graph ladder networks (GLN) algorithm for networked data. I argue that most existing representation-based algorithms separate the classification task into two phases which is not the best choice, and they can only learn simple and linear representation for networked data. In this paper, I propose an effective algorithm for jointly representing and classifying nodes in networked data by developing a graph ladder networks architecture. By simultaneously minimizing both supervised cost and unsupervised cost in the objective function, the graph ladder networks provides an effective solution to exploit both labeled and unlabeled data. By employing a graph convolution network, our algorithm can integrate both structure and content into a unified framework for network classification. based on simple neural network structures are unable to effectively leverage the deep information of the networks. Although these are deep learning methods, especially convolution neural network is outstanding in areas like image classification and pattern recognition, the traditional convolution operations cannot effectively exploit the unlabelled network data. Accordingly, I proposed a novel algorithm to explore both labelled and unlabelled data with ladder network structure and employed the graph convolution network architecture for exploiting network data. The experimental results illustrated the outstanding performance of our algorithm. The proposed graph ladder network architecture offer a solution to simultaneously represent and classify networked data and exploit both labelled and unlabelled data, which has

been experimentally proved could enhance the performance of node classification on the learned embedding.

Chapter 4

Adversarially Regularized Graph Autoencoder

4.1 Motivations

Graph embedding converts graph data into a low dimensional, compact, and continuous feature space. The key idea is to preserve the topological structure, vertex content, and other side information [27]. This new learning paradigm has shifted the tasks of seeking complex models for classification, clustering, and link prediction to learning a robust representation of the graph data, so that any graph analytic task can be easily performed by employing simple traditional models (e.g., a linear SVM for the classification task). This merit has motivated a number of studies in this area [28, 29].

Graph embedding algorithms can be classified into three categories: probabilistic models, matrix factorization-based algorithms, and deep learning-based algorithms. Probabilistic models like DeepWalk [13], node2vec [30] and LINE [31] attempt to learn graph embedding by extracting different patterns from the graph. The captured patterns or walks include global structural equivalence, local neighborhood connectivities, and other various order proximities. Compared with classical methods such as Spectral Clustering [33], these graph embedding algorithms perform more effectively and are scalable to large graphs.

Matrix factorization-based algorithms, such as GraRep [34], HOPE [35], M-NMF [36] pre-process the graph structure into an adjacency matrix and get the embedding by decomposing the adjacency matrix. Recently it has been shown that many probabilistic algorithms are equivalent to matrix factorization approaches [38]. Deep

learning approaches, especially autoencoder-based methods, are also widely studied for graph embedding. SDNE [41] and DNGR [42] employ deep autoencoders to preserve the graph proximities and model positive pointwise mutual information (PPMI). The MGAE algorithm utilizes a marginalized single layer autoencoder to learn representation for clustering [11].

The approaches above are typically unregularized approaches which mainly focus on preserving the structure relationship (probabilistic approaches), or minimizing the reconstruction error (matrix factorization or deep learning methods). They have mostly ignored the data distribution of the latent codes. In practice unregularized embedding approaches often learn a degenerate *identity* mapping where the latent code space is free of any structure [17], and can easily result in poor representation in dealing with real-world sparse and noisy graph data. One common way to handle this problem is to introduce some regularization to the latent codes and enforce them to follow some prior data distribution (e.g. Gaussian Distribution and Uniform Distribution) [17]. Recently generative adversarial based frameworks [57, 58, 59, 60] have also been developed for learning robust latent representation. However, none of these frameworks is specifically for graph data, where both topological structure and content information are required to embed to a latent space.

Inspired by the way of processing noised graph data and avoiding the inferior embedding, denoising autoencoder[61] recover the clean samples from the noised ones to learn the robust embedding, which is the denoising criterion. Generative adversarial based frameworks[57, 58, 59, 60] have also been implemented for learning robust and multi-functional graph embedding, which have done well on image data set[60] and text data set[62]. However, none of these frameworks is specifically constructed for graph and able to naturally encode both topological structure and the node content.

In this paper, I propose a novel adversarial framework with two variants, namely *adversarially regularized graph autoencoder* (ARGA) and *adversarially regularized variational graph autoencoder* (ARVGA), for graph embedding. The theme of our framework is to not only minimize the reconstruction errors of the graph structure but also to enforce the latent codes to match a prior distribution. By exploiting both graph structure and node content with a graph convolutional network, our algorithms encodes the graph data in the latent space. With a decoder aiming at reconstructing the topological graph information, I further incorporate an adversarial training scheme to regularize the latent codes to learn a robust graph representation. The adversarial training module aims to discriminate if the latent codes are from a real prior distribution or from the graph encoder. The graph encoder learning and adversarial regularization learning are jointly optimized in a unified framework so that each can be beneficial to the other and finally lead to a better graph embedding. The experimental results on benchmark datasets demonstrate the superb performance of our algorithms on three unsupervised graph analytic tasks, namely link prediction, node clustering, and graph visualization. The contributions can be summarized below:

- I propose a novel adversarially regularized framework for graph embedding, which represent topological structure and node content in a continuous vector space. Our framework learns the embedding to minimize the reconstruction error while enforcing the latent codes to match a prior distribution.

- I develop two variants of adversarial approaches, *adversarially regularized graph autoencoder* (ARGA) and *adversarially regularized variational graph autoencoder* (ARVGA) to learn the graph embedding.

- Experiments on benchmark graph datasets demonstrate that our graph embedding approaches outperforms the others on three unsupervised tasks.

4.2 Most Related Works to Graph Autoencoder

Graph Embedding Models. From the perspective of information exploration, graph embedding algorithms can be also separated into two groups: topological embedding approaches and content enhanced embedding methods.

Topological embedding approaches assume that there is only topological structure information available, and the learning objective is to maximally preserve the topological information. Inspired by the word embedding approach [22], Perozzi et al. propose a DeepWalk model to learn the node embedding from a collection of random walks [13]. Since then, a number of probabilistic models such as node2vec [30] and LINE [31] have been developed. As a graph can be mathematically represented as an adjacency matrix, many matrix factorization approaches such as GraRep [34], HOPE [35], M-NMF [36] are proposed to learn the latent representation for a graph. Recently deep learning models have been widely exploited to learn the graph embedding. These algorithms preserve the first and second order of proximities [41], or reconstruct the positive pointwise mutual information (PPMI) [42] via different variants of autoencoders.

Most previous work focus on persevering one type of information and graph structure analysis-based ones are more prevalent. Doc2Vec[23] and Word2Vec unveil the power of deep learning in natural language processing (NLP) applications. DeepWalk[13] follows the trajectory of Doc2Vec and leans the embedding of each node from a corpus of randomly generated walks of the graph by regarding each walk as a sentence in a document. These algorithms only leverage the graph structure, never considering any features associated with each node.

Bog Of Words(BOW) and topic model support early content based approaches to embed each content document into a low-dimensional vector without viewing contextual features like the order of documents or the order of words. To lever-

age the contextual information, algorithms like skip-gram[22] and skip-thought[50] learn the embedding of words by taking a window-size of consecutive words in the sentences of a document. However, the exponentially increasing number of features always significantly cost the training time and weaken the performance of these algorithms. Recurrent neural network (RNN) allow algorithms like gated recurrent unit(GRU)[63] and long short term memory(LSTM)[49] to capture long-term dependence with internal memory for processing arbitrary textual sequences.

Content enhanced embedding methods assume node content information is available and exploit both topological information and content features simultaneously. TADW [14] proved that DeepWalk can be interpreted as a factorization approach and proposed an extension to DeepWalk to explore node features. TriDNR [25] captures structure, node content, and label information via a tri-party neural network architecture. UPP-SNE employs an approximated kernel mapping scheme to exploit user profile features to enhance the embedding learning of users in social networks [64].

Although these algorithms are well-designed for graph-structured data, they largely ignore the latent distribution of the embedding, which may result in poor representation in practice. In this paper, I explore adversarial training methods to address this issue.

Adversarial Models. Our method is motivated by the generative adversarial network (GAN) [65]. GAN plays an adversarial game with two linked models: the generator $\mathcal{G}$ and the discriminator $\mathcal{D}$. The discriminator can be a multi-layer perceptron which discriminates if an input sample comes from the data distribution or from the generator. Simultaneously, the generator is trained to generate the samples to convince the discriminator that the generated samples come from the prior data distribution. The generator leverage the gradient from discriminator to update itself

and touch up its parameters. Normally, the training process has been split into two moments: (1) Train the discriminator $\mathcal{D}$ for iterations to distinguish the samples from the expected data distribution from the samples generated via the generator. Then (2) train the generator to confuse the discriminator with its generated ones. However, the original GAN does not fit the unsupervised graph embedding, as the absence of precise structure for inference. Due to its effectiveness in many unsupervised tasks, recently a number of algorithms have been proposed including BiGAN [57], EBGAN [58] and ALI [59]. To implement adversarial structure in learning graph embedding, existing works like BiGAN[57], EBGAN[58] and ALI[59] arrive at extending the original adversarial framework with external structures for the inference, which have achieved non-negligible performance in applications, such as document retrieval[62] and image classification[57]. Other solutions like DCGAN[60] and AIDW[66], manage to generating the embedding from the discriminator or generator for semi-supervised and supervised tasks via reconstructed layers.

Recently Makhzani et al. proposed an adversarial autoencoder (AAE) to learn the latent embedding by merging the adversarial mechanism into the autoencoder [17]. However, it is designed for general data rather than graph data. Dai et al. applied the adversarial mechanism to graphs. However, their approach can only exploit the topological information [66]. In contrast, our algorithm is more flexible and can handle both topological and content information for graph data.

Makhzani et al constructed the adversarial autoencoder (AAE) to learn the embedding from generator for unsupervised tasks by merging the adversarial mechanism into the autoencoder [17]. Specifically, AAE treats the latent code vector (embedding neurons) from the encoder of a deep autoencoder as the input of the discriminator. In this case, the generator of the adversarial network is also the encoder of the deep autoencoder. Similarly, the adversarial network and the autoencoder of AAE are jointly trained with two stages: reconstruction stage and

regularization stage. Reconstruction stage updates the autoencoder through minimizing the reconstruction cost between the input of encoder and the outcome of the decoder. Regularization stage updates the discriminator net by differentiating the positive samples from the expected data distribution from the negative samples (embedding vector from the autoencoder). With trained discriminator, the adversarial net update the generator(which is the encoder of the autoencoder) by fooling its discriminative net.

Adervarsial Models like GAN and AAE have achieved impressive success on image classification and generative tasks on the unsupervised learning, however due to the limitation of normal multi-layer perceptron structure, they cannot effectively and directly handle the graph-structured data. In this paper, I exploit the adversarial training scheme to graph data to learn a robust graph embedding.

Graph Convolutional Nets based Models

Graph convolutional networks (GCN) [9] is a semi-supervised framework based on a variant of convolutional neural networks, which attempt to directly operate the graphs. Specifically, the GCN represents the graph structure and the interrelationship between node and feature with an adjacent matrix $\mathbf{A}$ and node-feature matrix $\mathbf{X}$. Hence, GCN can directly embed the graph structure with a spectral convolutional function $f(\mathbf{X}, \mathbf{A})$ for each layer and train the model on a supervised target for all labelled nodes, which shuns the specific the regularization of graph in the cost function. Because of spectral function $f(\bullet)$ on the adjacent matrix $\mathbf{A}$ of the graph, the model is able to distribute the gradient from the supervised cost and learn the embedding of both the labelled and unlabelled nodes. The technical details are described in the chapter 4.2. Although GCN is powerful with graph-structured data sets on semi-supervised tasks like node classification, variational graph autoencoder VGAE [43] extends it into unsupervised implements with outstanding

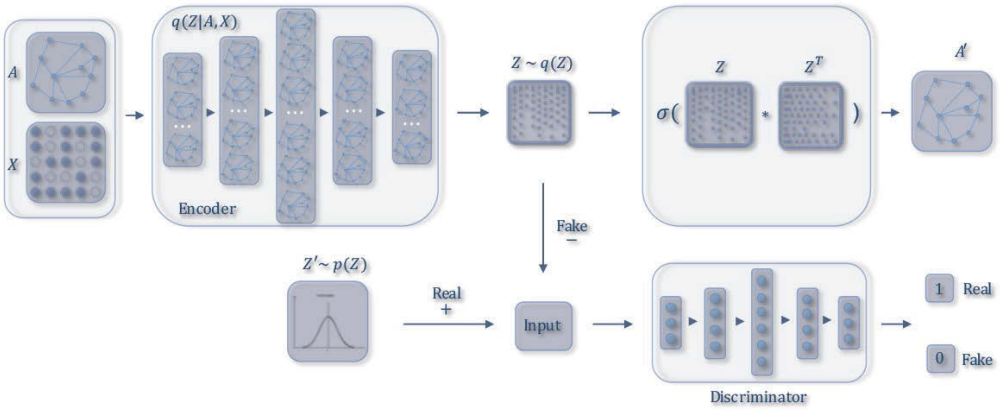

Figure 4.1 : The architecture of the adversarially regularized graph autoencoder (ARGA).

performance on link prediction. VGAE implements the GCN into the variational autoencoder framework [44] by framing the encoder with graph convolutional layers and remolding the decoder with a simple inner production calculation layer. Taking the advantage of GCN layers, VGAE can naturally leverage the information of node features, which significantly improve the predictive performance.

4.3 Problem Definition and Framework

A graph is represented as $\mathbf{G} = \{\mathbf{V}, \mathbf{E}, \mathbf{X}\}$, where $\mathbf{V} = \{\mathbf{v}_i\}_i = 1, \cdots, n$ consists of a set of nodes in a graph and $\mathbf{e}_{i,j} =< \mathbf{v}_i, \mathbf{v}_j > \in \mathbf{E}$ represents a linkage encoding the citation edge between the nodes. The topological structure of graph $\mathbf{G}$ can be represented by an adjacency matrix $\mathbf{A}$, where $\mathbf{A}_{i,j} = 1$ if $\mathbf{e}_{i,j} \in \mathbf{E}$, otherwise $\mathbf{A}_{i,j} = 0$. $\mathbf{x}_i \in \mathbf{X}$ indicates the content features associated with each node $\mathbf{v}_i$.

Given a graph $\mathbf{G}$, our purpose is to map the nodes $\mathbf{v}_i \in \mathbf{V}$ to low-dimensional vectors $\mathbf{z}_i \in \mathbb{R}^d$ with the formal format as follows: $f : (\mathbf{A}, \mathbf{X}) \rightarrowtail \mathbf{Z}$, where $\mathbf{z}_i^\top$ is the i-th row of the matrix $\mathbf{Z} \in \mathbb{R}^{n \times d}$. n is the number of nodes and d is the dimension of embedding. I take $\mathbf{Z}$ as the embedding matrix and the embeddings should well preserve the topological structure $\mathbf{A}$ as well as content information $\mathbf{X}$.

4.3.1 Overall Framework

Our objective is to learn a robust embedding given a graph $\mathbf{G} = \{\mathbf{V}, \mathbf{E}, \mathbf{X}\}$. To this end, I leverage an adversarial architecture with a graph autoencoder to directly process the entire graph and learn a robust embedding. Figure 4.1 demonstrates the workflow of ARGA which consists of two modules: the graph autoencoder and the adversarial network. The upper tier is a graph convolutional autoencoder that reconstructs a graph $\mathbf{A}$ from an embedding $\mathbf{Z}$ which is generated by the encoder which exploits graph structure $\mathbf{A}$ and the node content matrix $\mathbf{X}$. The lower tier is an adversarial network trained to discriminate if a sample is generated from the embedding or from a prior distribution. The adversarially regularized variational graph autoencoder (ARVGA) is similar to ARGA except that it employs a *variational* graph autoencoder in the upper tier (See Algorithm 1 for details).

- **Graph convolutional autoencoder**. The autoencoder takes in the structure of graph $\mathbf{A}$ and the node content $\mathbf{X}$ as inputs to learn a latent representation $\mathbf{Z}$, and then reconstructs the graph structure $\mathbf{A}$ from $\mathbf{Z}$.

- **Adversarial regularization.** The adversarial network forces the latent codes to match a prior distribution by an adversarial training module, which discriminates whether the current latent code $\mathbf{z}_i \in \mathbf{Z}$ comes from the encoder or from the prior distribution.

4.4 Proposed Algorithm

4.4.1 Graph convolutional autoencoder

The graph convolutional autoencoder aims to embed a graph $\mathbf{G} = \{\mathbf{V}, \mathbf{E}, \mathbf{X}\}$ in a low-dimensional space. Two key questions arise (1) how to integrate both graph structure $\mathbf{A}$ and node content $\mathbf{X}$ in an encoder, and (2) what sort of information should be reconstructed via a decoder?

Graph Convolutional Encoder Model $\mathcal{G}(\mathbf{X}, \mathbf{A})$. To represent both graph structure $\mathbf{A}$ and node content $\mathbf{X}$ in a unified framework, I develop a variant of the graph convolutional network (GCN) [9] as a graph encoder. Our graph convolutional network (GCN) extends the operation of *convolution* to graph data in the spectral domain, and learns a layer-wise transformation by a spectral convolution function $f(\mathbf{Z}^{(l)}, \mathbf{A}|\mathbf{W}^{(l)})$:

$$\mathbf{Z}^{(l+1)} = f(\mathbf{Z}^{(l)}, \mathbf{A}|\mathbf{W}^{(l)}) \tag{4.1}$$

Here, $\mathbf{Z}^l$ is the input for convolution, and $\mathbf{Z}^{(l+1)}$ is the output after convolution. We have $\mathbf{Z}^0 = \mathbf{X} \in \mathbb{R}^{n \times m}$ (n nodes and m features) for the problem. $\mathbf{W}^{(l)}$ is a matrix of filter parameters we need to learn in the neural network. If $f(\mathbf{Z}^{(l)}, \mathbf{A}|\mathbf{W}^{(l)})$ is well defined, we can build arbitrary deep convolutional neural networks efficiently.

Each layer of our graph convolutional network can be expressed with the function $f(\mathbf{Z}^{(l)}, \mathbf{A}|\mathbf{W}^{(l)})$ as follows:

$$f(\mathbf{Z}^{(l)}, \mathbf{A}|\mathbf{W}^{(l)}) = \phi(\widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{Z}^{(l)}\mathbf{W}^{(l)}), \tag{4.2}$$

where $\widetilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and $\widetilde{\mathbf{D}}_{ii} = \sum_j \widetilde{\mathbf{A}}_{ij}$. $\mathbf{I}$ is the identity matrix of $\mathbf{A}$ and ϕ is an activation function such as $\text{Relu}(t) = \max(0, t)$ or $\text{sigmoid}(t) = \frac{1}{1+e^t}$. Overall, the graph encoder $\mathcal{G}(\mathbf{X}, \mathbf{A})$ is constructed with a two-layer GCN. In this paper, I develop two variants of encoder, e.g., Graph Encoder and Variational Graph Encoder.

The *Graph Encoder* is constructed as follows:

$$\mathbf{Z}^{(1)} = f_{\text{Relu}}(\mathbf{X}, \mathbf{A}|\mathbf{W}^{(0)}); \tag{4.3}$$

$$\mathbf{Z}^{(2)} = f_{\text{linear}}(\mathbf{Z}^{(1)}, \mathbf{A}|\mathbf{W}^{(1)}). \tag{4.4}$$

$\text{Relu}(\cdot)$ and linear activation functions are used for the first and second layers. Our graph convolutional encoder $\mathcal{G}(\mathbf{Z}, \mathbf{A}) = q(\mathbf{Z}|\mathbf{X}, \mathbf{A})$ encodes both graph structure and node content into a representation $\mathbf{Z} = q(\mathbf{Z}|\mathbf{X}, \mathbf{A}) = \mathbf{Z}^{(2)}$.

A *Variational Graph Encoder* is defined by an inference model:

$$q(\mathbf{Z}|\mathbf{X},\mathbf{A}) = \prod_{i=1}^{n} q(\mathbf{z_i}|\mathbf{X},\mathbf{A}), \tag{4.5}$$

$$q(\mathbf{z_i}|\mathbf{X},\mathbf{A}) = \mathcal{N}(\mathbf{z}_i|\mu_i, \mathrm{diag}(\sigma^2)) \tag{4.6}$$

Here, $\mu = \mathbf{Z}^{(2)}$ is the matrix of mean vectors z_i ; similarly $\log\sigma = f_{\mathrm{linear}}(\mathbf{Z}^{(1)}, \mathbf{A}|\mathbf{W}'^{(1)})$ which share the weights $\mathbf{W}^{(0)}$ with μ in the first layer in Eq. (4.3).

Decoder Model. Our decoder model is used to reconstruct the graph data. I can reconstruct either the graph structure $\mathbf{A}$, content information $\mathbf{X}$, or both. In this paper, I propose to reconstruct graph structure $\mathbf{A}$, which provides more flexibility in the sense that our algorithm will still function properly even if there is no content information $\mathbf{X}$ available (e.g., $\mathbf{X} = \mathbf{I}$). The decoder $p(\hat{\mathbf{A}}|\mathbf{Z})$ predicts whether there is a link between two nodes. More specifically, I train a link prediction layer based on the graph embedding:

$$p(\hat{\mathbf{A}}|\mathbf{Z}) = \prod_{i=1}^{n}\prod_{j=1}^{n} p(\hat{\mathbf{A}}_{ij}|\mathbf{z}_i,\mathbf{z}_j); \tag{4.7}$$

$$p(\hat{\mathbf{A}}_{ij} = 1|\mathbf{z}_i,\mathbf{z}_j) = \mathrm{sigmoid}(\mathbf{z}_i^\top,\mathbf{z}_j), \tag{4.8}$$

Graph Autoencoder Model. The embedding $\mathbf{Z}$ and the reconstructed graph $\hat{\mathbf{A}}$ can be presented as follows:

$$\hat{\mathbf{A}} = \mathrm{sigmoid}(\mathbf{Z}\mathbf{Z}^\top), \text{ here } \mathbf{Z} = q(\mathbf{Z}|\mathbf{X},\mathbf{A}) \tag{4.9}$$

Optimization. For the graph encoder, I minimize the reconstruction error of the graph data by:

$$\mathcal{L}_0 = \mathbb{E}_{q(\mathbf{Z}|(\mathbf{X},\mathbf{A}))}[\log\, p(\mathbf{A}|\mathbf{Z})] \tag{4.10}$$

For the variational graph encoder, I optimize the variational lower bound as follows:

$$\mathcal{L}_1 = \mathbb{E}_{q(\mathbf{Z}|(\mathbf{X},\mathbf{A}))}[\log\, p(\mathbf{A}|\mathbf{Z})] - \mathbf{KL}[q(\mathbf{Z}|\mathbf{X},\mathbf{A}) \parallel p(\mathbf{Z})] \tag{4.11}$$

where $\mathbf{KL}[q(\bullet)||p(\bullet)]$ is the Kullback-Leibler divergence between $q(\bullet)$ and $p(\bullet)$. I also take a Gaussian prior $p(\mathbf{Z}) = \prod_i p(\mathbf{z}_i) = \prod_i \mathcal{N}(\mathbf{z}_i|0,\mathbf{I})$.

4.4.2 Adversarial Model $\mathcal{D}(\mathbf{Z})$

The key idea of our model is to enforce latent representation $\mathbf{Z}$ to match a prior distribution, which is achieved by an adversarial training model. The adversarial model is built on a standard multi-layer perceptron (MLP) where the output layer only has one dimension with a sigmoid function. The adversarial model acts as a discriminator to distinguish whether a latent code is from the prior p_z (positive) or from graph encoder $\mathcal{G}(\mathbf{X}, \mathbf{A})$ (negative). By minimizing the cross-entropy cost for training the binary classifier, the embedding will finally be regularized and improved during the training process. The cost can be computed as follows:

$$-\frac{1}{2}\mathbb{E}_{\mathbf{z}\sim p_z}\log\mathcal{D}(\mathbf{Z}) - \frac{1}{2}\mathbb{E}_{\mathbf{X}}\log(1-\mathcal{D}(\mathcal{G}(\mathbf{X}, \mathbf{A}))), \tag{4.12}$$

In this paper, I use simple Gaussian distribution as p_z.

Adversarial Graph Autoencoder Model. The equation for training the encoder model with Discriminator $\mathcal{D}(\mathbf{Z})$ can be written as follows:

$$\min_{\mathcal{G}}\max_{\mathcal{D}}\mathbb{E}_{\mathbf{z}\sim p_z}[\log\mathcal{D}(\mathbf{Z})] + \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}[\log(1-\mathcal{D}(\mathcal{G}(\mathbf{X}, \mathbf{A})))] \tag{4.13}$$

where $\mathcal{G}(\mathbf{X}, \mathbf{A})$ and $\mathcal{D}(\mathbf{Z})$ indicate the generator and discriminator explained above.

4.4.3 Algorithm Explanation

Algorithm 1 is our proposed framework. Given a graph $\mathbf{G}$, the step 2 gets the latent variables matrix $\mathbf{Z}$ from the graph convolutional encoder. Then I take the same number of samples from the generated $\mathbf{Z}$ and the real data distribution p_z in step 4 and 5 respectively, to update the discriminator with the cross-entropy cost computed in step 6. After K runs of training the discriminator, the graph encoder will try to confuse the trained discriminator and update itself with generated gradient in step 7. I can update Eq. (4.10) to train the **adversarially regularized graph autoencoder (ARGA),** or Eq. (4.11) to train the **adversarially regu-**

Algorithm 1 Adversarially Regularized Graph Embedding

Require:

$\mathbf{G} = \{\mathbf{V}, \mathbf{E}, \mathbf{X}\}$: a Graph with links and features;

T: the number of iterations;

K: the number of steps for iterating discriminator;

d: the dimension of the latent variable

Ensure: $\mathbf{Z} \in \mathbb{R}^{n \times d}$

1: **for** iterator = 1,2,3, $\cdots\cdots$, T **do**

2: Generate latent variables matrix $\mathbf{Z}$ through Eq.(4.4);

3: **for** k = 1,2, $\cdots\cdots$, K **do**

4: Sample m entities $\{\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(m)}\}$ from latent matrix $\mathbf{Z}$

5: Sample m entities $\{\mathbf{a}^{(1)}, \ldots, \mathbf{a}^{(m)}\}$ from the prior distribution p_z

6: Update the discriminator with its stochastic gradient:

$$\bigtriangledown \frac{1}{m} \sum_{i=1}^{m} [\log \mathcal{D}(\mathbf{a}^i) + \log (1 - \mathcal{D}(\mathbf{z}^{(i)}))]$$

7: **end forend for**

8: Update the graph autoencoder with its stochastic gradient by Eq. (4.10) for ARGA or Eq. (4.11) for ARVGA;

9: **end forend for**

10: **return** $\mathbf{Z} \in \mathbb{R}^{n \times d}$

larized variational graph autoencoder (ARVGA), respectively. Finally, the graph embedding $\mathbf{Z} \in \mathbb{R}^{n \times d}$ will be returned in step 8.

4.5 Experiments

I report experiment results on three unsupervised graph analytic tasks: link prediction, node clustering, and graph visualization. The benchmark graph datasets used in the paper are summarized in Table 1. Each data set consists of scientific publications as nodes and citation relationships as edges. The features are unique words in each document.

Table 4.1 : Real-world Graph Datasets Used in the Paper

Data Set	# Nodes	# Links	# Content Words	# Features
Cora	2,708	5,429	3,880,564	1,433
Citeseer	3,327	4,732	12,274,336	3,703
PubMed	19,717	44,338	9,858,500	500

4.5.1 Link Prediction

Baselines. I compared the proposed algorithms against state-of-the-art algorithms for the link prediction task:

- **DeepWalk** [13]: is a network representation approach which encodes social relations into a continuous vector space.
- **Spectral Clustering** [33]: is an effective approach for learning social embedding.
- **GAE***: is an autoencoder based unsupervised framework for graph data. *

version does not consider the node features, only leveraging the topological information source.

- **VGAE***: is a Variational autoencoder based unsupervised framework for graph data. * version does not consider the node features, only leveraging the topological information source.

- **GAE** [43]: is the most recent autoencoder-based unsupervised framework for graph data, which naturally leverages both topological and content information.

- **VGAE** [43]: is a variational graph autoencoder approach for graph embedding with both topological and content information.

- **ARGA**: Our proposed adversarially regularized autoencoder algorithm which uses graph autoencoder to learn the embedding.

- **ARVGA**: Our proposed algorithm, which uses a *variational* graph autoencoder to learn the embedding.

Metric. I report the results in terms of AUC score (the area under the ROC curve) and average precision (AP) [43] score. I conduct each experiment 10 times and report the mean values with the standard errors as the final scores. Each dataset is separated into a training, testing set and validation set. The validation set contains 5% citation edges for hyperparameter optimization, the test set holds 10% citation edges to verify the performance, and the rest are used for training.

Parameter Settings. For DeepWalk, I stick with the described setting of each epoch in the paper: 128 embedding dimension, 10 random walks with 80 length every node. Spectral Clustering has the same 128 embedding dimension as DeepWalk. For Graph Autoencoder(GAE) related algorithms, I initialize their weights at each layer

with a commonly used heuristic:

$$\mathcal{W} \sim \mathcal{U}[-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}] \tag{4.14}$$

where $\mathcal{W}$ is the weights of each layer and $\mathcal{U}[-a, a]$ indicates the uniform distribution with the window size (-a,a). n is the number of neurones in the previous layer. For the Cora and Citeseer data sets, I train all autoencoder-related models for 200 iterations and optimize them with the Adam algorithm. Both learning rate and discriminator learning rate are set as 0.001. As the PubMed data set is relatively large (around 20,000 nodes), I iterate 2,000 times for an adequate training with a 0.008 discriminator learning rate and 0.001 learning rate. I construct encoders with a 32-neuron hidden layer and a 16-neuron embedding layer for all the experiments and all the discriminators are built with two hidden layers(16-neuron, 64-neuron respectively). For the rest of the baselines, I retain to the settings described in the corresponding papers.

Metrics

For link prediction, I report the AUC score shorted for area under a receiver operating characteristic (ROC) curve which can be computed as follow:

$$AUC = \frac{\sum_i^1 \sum_j^1 pred(x_i) > pred(y_j)}{N * M}$$

where $pred(\bullet)$ is the outputs from the predictor and N and M are the number of positive samples $X \ni x_i$ and the number of negative samples $Y \ni y_j$ respectively. I also report the Average Precision(AP) which indicates the area under the precision-recall curve. The calculation of AP as follow:

$$Precision = \frac{true\_positive}{true\_positive + false\_positive}$$

$$AveragePrecision(AP) = \frac{\sum_k Precision(k)}{\#\{positive\_sample\}}$$

where k is an index if positive sample.

Practically I make use of the functions of roc_auc_score() and average_precision_score() from sklearn.metrics to achieve the AUC and AP.

Experimental Results. The details of the experimental results on the link prediction are shown in Table 2. The results show that by incorporating an effective adversarial training module into our graph convolutional autoencoder, ARGA and ARVGA achieve outstanding performance: all AP and AUC scores are as higher as 92% on all three data sets. Compared with all the baselines, ARGE increased the AP score from around 2.5% compared with VGAE incorporating with node features, 11% compared with VGAE without node features; 15.5% and 10.6% compared with DeepWalk and Spectral Clustering respectively on the large PubMed data set .

Parameter Study. I vary the dimension of embedding from 8 neurons to 1024 and report the results in Fig 4.2.

The results from both Fig 4.2 (A) and (B) reveal similar trends: when adding the dimension of embedding from 8-neuron to 16-neuron, the performance of embedding on link prediction steadily rises; but when I further increase the number of the neurons at the embedding layer to 32-neuron, the performance fluctuates however the results for both the AP score and the AUC score remain good.

It is worth mentioning that if I continue to set more neurons, for examples, 64-neuron, 128-neuron and 1024-neuron, the performance rises markedly.

4.5.2 Node Clustering

For the node clustering task, I first learn the graph embedding, and then perform K-means clustering algorithm based on the embedding.

Baselines I compare both embedding based approaches as well as approaches directly for graph clustering. Except for the baselines I compared for link prediction,

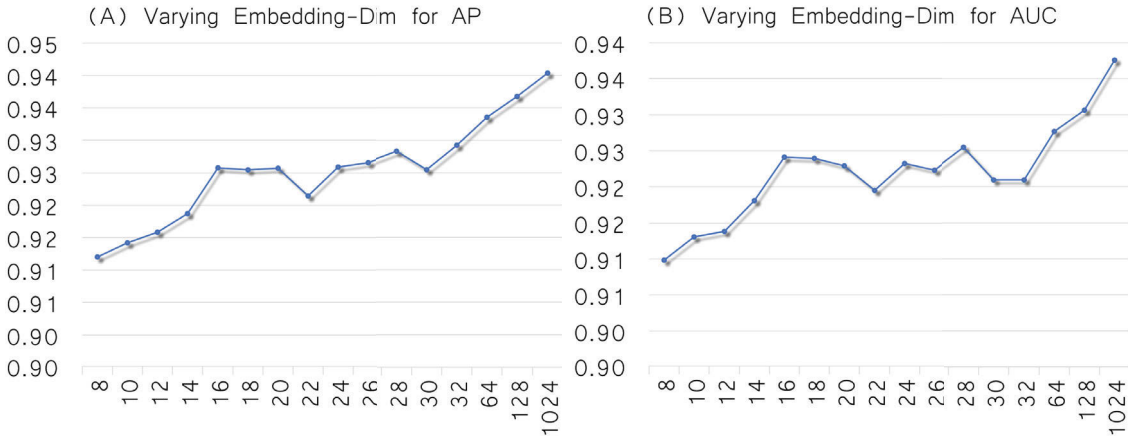

Figure 4.2 : Average performance on different dimensions of the embedding. (A) Average Precision score; (B) AUC score.

Table 4.2 : Results for Link Prediction. GAE* and VGAE* are variants of GAE and VGAE, which only explore topological structure, i.e., $\mathbf{X} = \mathbf{I}$.

Approaches	Cora		Citeseer		PubMed	
	AUC	AP	AUC	AP	AUC	AP
SC	84.6 ± 0.01	88.5 ± 0.00	80.5 ± 0.01	85.0 ± 0.01	84.2 ± 0.02	87.8 ± 0.01
DW	83.1 ± 0.01	85.0 ± 0.00	80.5 ± 0.02	83.6 ± 0.01	84.4 ± 0.00	84.1 ± 0.00
GAE*	84.3 ± 0.02	88.1 ± 0.01	78.7 ± 0.02	84.1 ± 0.02	82.2 ± 0.01	87.4 ± 0.00
VGAE*	84.0 ± 0.02	87.7 ± 0.01	78.9 ± 0.03	84.1 ± 0.02	82.7 ± 0.01	87.5 ± 0.01
GAE	91.0 ± 0.02	92.0 ± 0.03	89.5 ± 0.04	89.9 ± 0.05	96.4 ± 0.00	96.5 ± 0.00
VGAE	91.4 ± 0.01	92.6 ± 0.01	90.8 ± 0.02	92.0 ± 0.02	94.4 ± 0.02	94.7 ± 0.02
ARGE	92.4 ± 0.003	**93.2 ± 0.003**	91.9 ± 0.003	93.0± 0.003	**96.8 ± 0.001**	**97.1 ± 0.001**
ARVGE	**92.4 ± 0.004**	92.6 ± 0.004	**92.4 ± 0.003**	**93.0 ± 0.003**	96.5± 0.001	96.8± 0.001
$ARGE_{gd}$	74.5 ± 0.003	74.3 ± 0.003	69.1 ± 0.003	68.4± 0.003	93.3 ± 0.001	92.0 ± 0.001
$ARVGE_{gd}$	73.1 ± 0.004	70.1 ± 0.004	69.3 ± 0.003	62.6 ± 0.003	92.3± 0.001	91.0 ± 0.001

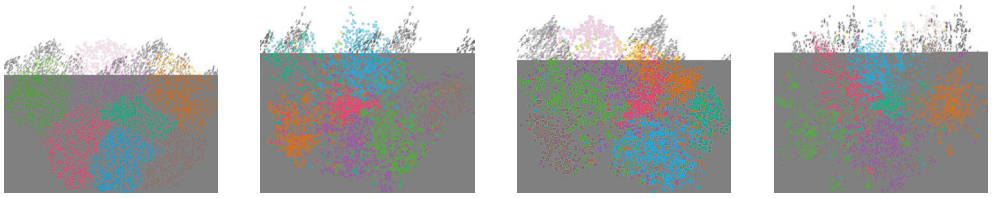

Figure 4.3 : The Cora data visualization comparison. From left to right: graphs from our ARGA, VGAE, GAE, and DeepWalk. The different colors represent different groups.

and also include baselines which are designed for clustering: Twelve approaches in total are compared in the experiments. For a comprehensive validation, I take the algorithms which only consider one perspective of information source, say, network structure or node content, while these consider both.

Node Content or Graph Structure Only:

i. **K-means** is a classical method and also the foundation of many clustering algorithms.

ii. **Big-Clam** [33] is a non-negative matrix factorization algorithm for community detection.

iii. **Graph Encoder** [67] learns graph embedding for spectral graph clustering.

iv. **DNGR** [42] trains a stacked denoising autoencoder for graph embedding.

Both Content and Structure

v. **Circles** [68] is an overlapping graph clustering algorithm which treats each node as an ego and builds the ego graph with the linkages between the ego's friends.

vi. **RTM** [69] learns the topic distributions of each document from both text and citation.

vii. **RMSC** [70] is a multi-source clustering algorithm which recovers the shared low-rank transition probability matrix from each source to conduct the clustering. In this paper, node content and topological structure are treated as two sources of information.

viii. **TADW** [14] applies matrix factorization for network representation learning.

Here the first three algorithms only exploit the graph structures, while the last three algorithms use both graph structure and node content for the graph clustering task.

Metrics: Following [70], I employ five metrics to validate the clustering results: accuracy (Acc), normalized mutual information (NMI), precision, F-score (F1) and average rand index (ARI).

Experimental Results. The clustering results on the Cora and Citeseer data sets are given in Table 3 and Table 4. The results show that ARGA and ARVGA have achieved a dramatic improvement on all five metrics compared with all the other baselines. For instance, on Citeseer, ARGA has increased the accuracy from 6.1% compared with K-means to 154.7% compared with GraphEncoder; increased the F1 score from 31.9% compared with TADW to 102.2% compared with DeepWalk; and increased NMI from 14.8% compared with K-means to 124.4% compared with VGAE. The wide margin in the results between ARGE and GAE (and the others) has further proved the superiority of our adversarially regularized graph autoencoder.

4.5.3 Graph Visualization

I visualize the Cora data in a two-dimensional space by applying the t-SNE algorithm [71] on the learned embedding. The results in Fig 3 validate that by applying adversarial training to the graph data, I can obtained a more meaningful layout of the graph data.

The results of classfication on Citeseer data set are visualized with Figure 3. Each color of nodes indicates a document class and the fine lines are the citation links between nodes. The nodes from different groups mixed significantly by DeepWalk, while the color in different classes are relatively clearer for GAE and VAGE, but still not as clear as the result from ARGA and ARVGA.

Table 4.3 : Clustering Results on Cora

Cora	Acc	NMI	F1	Precision	ARI
K-means	0.492	0.321	0.368	0.369	0.230
Spectral	0.367	0.127	0.318	0.193	0.031
BigClam (•)	0.272	0.007	0.281	0.180	0.001
GraphEncoder	0.325	0.109	0.298	0.182	0.006
DeepWalk	0.484	0.327	0.392	0.361	0.243
DNGR	0.419	0.318	0.340	0.266	0.142
Circles	0.607	0.404	0.469	0.501	0.362
RTM	0.440	0.230	0.307	0.332	0.169
RMSC	0.407	0.255	0.331	0.227	0.090
TADW	0.560	0.441	0.481	0.396	0.332
GAE* (•)	0.439	0.291	0.417	0.453	0.209
VGAE* (•)	0.443	0.239	0.425	0.430	0.175
GAE	0.596	0.429	0.595	0.596	0.347
VGAE	0.609	0.436	0.609	0.609	0.346
ARGE	0.640	0.449	0.619	0.646	0.352
ARVGE	0.638	0.450	0.627	0.624	0.374
$ARGE_{gd}$	0.624	0.464	0.602	0.626	0.395
$ARVGE_{gd}$	**0.711**	**0.506**	**0.690**	**0.692**	**0.487**

Table 4.4 : Clustering Results on Citeseer

Citeseer	Acc	NMI	F1	Precision	ARI
K-means	0.540	0.305	0.409	0.405	0.279
Spectral	0.239	0.056	0.299	0.179	0.010
BigClam (•)	0.250	0.036	0.288	0.182	0.007
GraphEncoder	0.225	0.033	0.301	0.179	0.010
DeepWalk	0.337	0.088	0.270	0.248	0.092
DNGR	0.326	0.180	0.300	0.200	0.044
Circles	0.572	0.301	0.424	0.409	0.293
RTM	0.451	0.239	0.342	0.349	0.203
RMSC	0.295	0.139	0.320	0.204	0.049
TADW	0.455	0.291	0.414	0.312	0.228
GAE* (•)	0.281	0.066	0.277	0.315	0.038
VGAE* (•)	0.304	0.086	0.292	0.331	0.053
GAE	0.408	0.176	0.372	0.418	0.124
VGAE	0.344	0.156	0.308	0.349	0.093
ARGE	0.573	0.350	0.546	0.573	0.341
ARVGE	0.544	0.261	0.529	0.549	0.245
$ARGE_{gd}$	**0.624**	**0.448**	**0.607**	**0.630**	**0.386**
$ARVGE_{gd}$	0.572	0.315	0.548	0.564	0.297

Table 4.5 : Clustering Results on Pubmed

Citeseer	Acc	NMI	F1	Precision	ARI
K-means	0.398	0.001	0.195	0.579	0.002
Spectral	0.403	0.042	0.271	0.498	0.002
BigClam (•)	0.250	0.036	0.288	0.182	0.007
DeepWalk	0.684	0.279	0.670	0.686	0.299
DNGR	0.458	0.155	0.467	0.629	0.054
Circles	0.572	0.301	0.424	0.409	0.293
RTM	0.574	0.194	0.444	0.455	0.148
TADW	0.354	0.001	0.335	0.336	0.001
GAE* (•)	0.581	0.196	0.569	0.636	0.162
VGAE* (•)	0.504	0.162	0.504	0.631	0.088
GAE	0.672	0.277	0.660	0.684	0.279
VGAE	0.630	0.229	0.634	0.630	0.213
ARGE	0.668	0.305	0.656	0.699	0.295
ARVGE	0.690	0.290	0.678	0.694	0.306
$ARGE_{gd}$	0.666	0.268	0.666	0.680	0.269
$ARVGE_{gd}$	**0.709**	**0.317**	**0.698**	**0.710**	**0.342**

Above results have proved that incorporating adversarial mechanism into graph autoencoder could essentially enhance the performance on the tasks like link prediction. Explain more the potential reasons why it works so well..

4.6 Summary

In this paper, I proposed a novel adversarial graph embedding framework for graph data. I argue that most existing graph embedding algorithms are unregularized methods that ignore the data distributions of the latent representation and suffer from inferior embedding in real-world graph data. I proposed an adversarial training scheme to *regularize* the latent codes and enforce the latent codes to match a prior distribution. The adversarial module is jointly learned with a graph convolutional autoencoder to produce a robust representation. Experiment results demonstrated that by introducing the adversarial mechanism into graph convolutional autoencoder can significantly improve the quality of embedding of the graph and enhance its performance on unsupervised tasks.

Chapter 5

Universal Representation for Heterogeneous Information Graphs

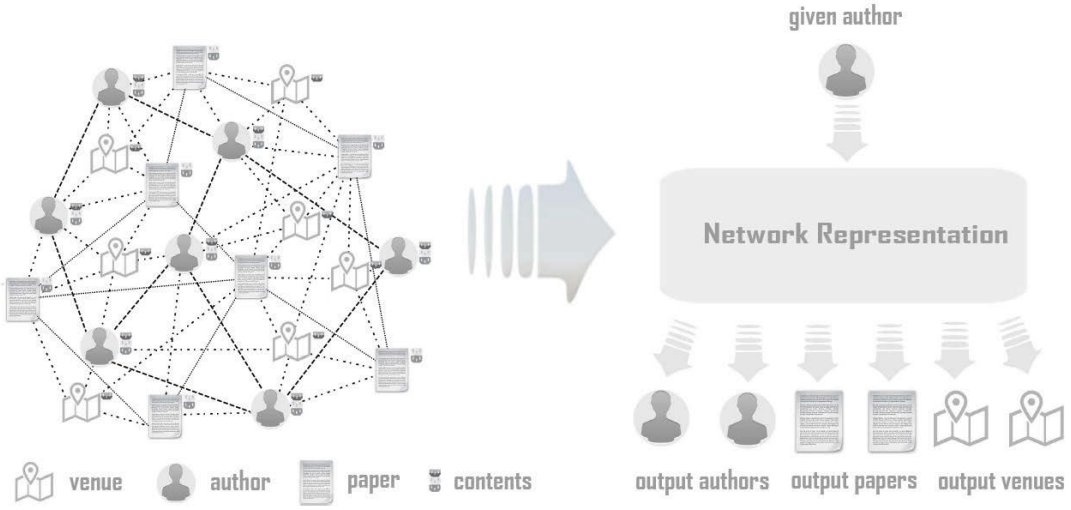

Figure 5.1 : An illustrated example of a heterogeneous publication graph.

5.1 Motivations

However, the majority of real world information graphs are heterogeneous and multi-relational, with many examples in same areas mentioned above, such as DNA-protein interaction graphs[18] in medicine or complete publication graphs[19, 20, 21] in academic engines. The sources of nodes in these information graphs vary but are interrelated. For instance, one example of a DNA-protein interaction graph consists of two sources of nodes, DNA and proteins, and each source of homogeneous nodes is profiled by an independent graph structure based on their inner reactions, say, a DNA graph structure and a protein graph structure. Furthermore, the interactions

between the heterogeneous nodes (DNA and protein) provide a more comprehensive, but complex, multi-relational heterogeneous graph, with different sources of nodes, graph structures and node contents. Publication graphs have the same characteristics: two different sources of nodes (papers and authors) with two independent graph expressions (paper citation graphs and author collaboration graphs) respectively according to their inner-reactions, while papers and authors are naturally connected by a heterogeneous graph because of their inherent publication correlations.

The obvious variety and complexity of multi-relational heterogeneous information graphs (MHIG) limit existing graph representation methods in two ways: (1) They can only leverage one perspective of data - either the graph structure or the node content - which simply ignores the integrity of the graph. (2) The final learned representation can only represent one of the sources of information - either the DNA **or** the protein in DNA-protein interaction graphs; or the papers **or** the authors in publication graphs. Most previous efforts have concentrated on persevering either the graph structure or the node content. Graph structure analysis-based methods are the more prevalent. Two very popular neural graph language models were proposed in [22, 23], where deep learning techniques revealed advantages in natural language processing applications. Inspired by that, a DeepWalk algorithm was proposed in [24] that learns latent representations of vertices from a corpus of generated random walks in graph data. These algorithms only input graph structures, without considering any content information affiliated with each node.

Early content-based algorithms employ approaches like, topic model and bag-of-words, to encode each content document into a vector without considering contextual information (i.e., the order of documents or the order of words), or sub-optimizing the representations. To acquire the contextual information, features are modeled using context-preserving algorithms [22, 45, 46, 47, 48] with a certain amount of consecutive words to represent a document. Obviously, an exponentially increas-

ing number of content features for these algorithms will dramatically increase the training time and weaken performance.

Within neural graph architectures, alternative algorithms like skip-gram [22] input a certain window of consecutive words in the sentences of a document to learn the representation of words. Recurrent neural network-based models, say, long short term memory(LSTM)[49] and gated recurrent unit(GRU) [50] are able to capture long term dependencies using internal memory to process arbitrary sequences of text input.

Either graph structure exploration approaches or text analysis based ones with three obvious drawbacks identified in the mentioned algorithms: (1) only one source of the entire graph information has been leveraged, which lowers representation accuracy; (2) learned representations from these algorithms can only represent one type of graph data (i.e., representations from a citation graph only represent papers, but ignore authors and venues); and (3) the algorithms are unsupervised. No labeled data are even available to use, which misses the opportunity to enhance performance in tasks like classification.

Recently, some methods have attempted to simultaneously consider graph structures and node content information. TADW [14] proves that DeepWalk [24] can be processed by factorizing an approximate probability matrix where one node randomly walks to another in certain steps, and incorporates with feature vectors by factorizing a word-association matrix. However, this algorithm is unsuit for processing large-scale data because matrix factorization is computationally expensive. It also ignores the contextual information in nodes. TriDNR [25] smoothly solves this problem simultaneously by learning the graph structure and node contents in a neural network architecture. However, TriDNR only leverages one source of the graph structure and the output only represents one type of graph data. The chal-

lenges facing complete graph data input into universal representation learning are listed below:

i. Comprehensively integrate and feed the complete graph data with heterogeneous but multi-relational node graph structures, node contents and labels;

ii. Generate a universal representation for all source of information in heterogeneous graph data.

In this paper, I propose a UGRA, a universal representation model in heterogeneous graphs, which uses mutually enhanced neural network architectures to learn representations for all sources of nodes (DNA **and** proteins in DNA-protein interaction graphs/networks; papers **and** authors in publication graphs) for input heterogeneous graphs. From the perspective of graph structures, the UGRA jointly learns the relationship among homogeneous nodes and the connections among heterogeneous nodes by maximizing the probability of discovering neighboring nodes given a node in random walks. From the perspective of node content, the UGRA captures the correlations between nodes and content by exploiting the co-occurrence of word sequences given a node.

To summarize, our contributions are as follows:

i. I propose a novel graph representation model that simultaneously leverages different sources of graph structures, node contents and labels in one heterogeneous graph; I propose a novel graph representation algorithm that simultaneously generates representations for different sources of nodes in a heterogeneous graph;

ii. I conduct a suite of experiments from different perspectives on real world data sets. The results of accuracy tests prove the effectiveness of our proposed

UGRA, and a case study practically shows the implementation of the UGRA in a real world setting.

An example demonstrating how the UGRA works in a multi-relational information graph is shown in Fig. 5.1. The graph consists of three different sources of nodes (authors, papers and venues) with two independent graph expressions(a paper citation graph and an author collaboration graph) according to their inner-reactions. Papers, authors and venues are naturally connected within a heterogeneous graph because of their inherent publication correlations. I also leverage the content information of the nodes (the title and abstracts of the papers).The graph representation is learned from the complete heterogeneous publication graph and it simultaneously represents all the sources of the nodes. I then input an author into the graph representation, which returns several authors, papers and venues that are most related to the given author. Any source of node can be compared to the representation for returning the most related nodes from other sources.

5.2 Problem Definition

A multi-relational information graph is defined as $N = \{V_k, E_k, D_k, S\}$,where $V_k = \{v_{ki}\}_{i=1,\cdots,n}$ consists of a set of nodes in the k_{th} source of a graph (i.e., V_1 denotes the nodes in a citation graph, which represents papers while V_2 denotes the nodes in an authors graph, which represents authors) and $e_{k(i,j)} =< v_{ki}, v_{kj} >\in E_k$ indicates an edge encoding the edge relationship between the nodes. $d_{ki} \in D_{ki}$ is a text document associated with each node v_{ki} and $S = L \cup U$ is the label information in the graph data, where L denotes labeled nodes and U denotes unlabeled ones. Different sources of the graph structures and the node contents are associated by an unique index according to their correlations. For example, all cited papers $paper_A$, all authors of $paper_A$, its title and abstract information, have the same index.

Given a multi-relational information graph defined as $N = \{V_k, E_k, D_k, S\}$, the purpose of graph representation is to learn a low dimensional vector $v_{v_{ki}} \in \mathbb{R}^r$ (r is a smaller number) for each node v_{ki} in k_{th} source of graphs. In this way, the nodes sharing connections, or with similar contents, stay closer to each other in the representation space. I assume that the graph data are partially labeled, but the UGRA is still valid if the set of label $S = \emptyset$. An example of the type of heterogeneous information graph used in this paper is illustrated in Fig. 5.1. Specifically, each source of a node graph structure, with its nodes and edges($\{V_k, E_k\}$), is independently extracted from the given heterogeneous information graph to generate random walks $\{v_{k1}, v_{k2}, v_{k5}, \cdots, v_{kn}\}$. All sources of the text content associated with one node are integrated into one line to generate a text document $\{d_{ki} \in D_{ki}\}$. Different sources of information are trained independently and multi-updated in the UGRA framework. The details will be explained in the following section.

5.3 The UGRA Algorithm

In this section, I demonstrate the details of the algorithm for generating universal representation for different types of nodes in multi-relational information graphs by leveraging different sources of graph structures, text node contents and label information. The essence of the UGRA algorithm is two-fold:

i. **The random walk path generator** inputs each source of graph structure, and generates a succession of steps over the nodes. Each walk starts at a node v_{ki} and then randomly passes by other nodes each time. The node relationship is captured and stored in a random walk corpus.

ii. **The neural network model training** inputs the complete multi-relational information graph, and embeds the different types of nodes into one continuous vector space. The information graph consists of (1) all random walks of the

graph structures(the node relationships); (2) the text node content corpus (the node-content correlation; (3)the label information to enhance performance in tasks like classification.

5.3.1 Framework Architecture

Since the number of node sources varies from graph to graph, I use a publication graph with two sources of node (papers and authors) as an example to explain how the UGRA works. Specifically, the UGRA has five steps in this publication graph. The source of the paper nodes are denoted as k = 1, and the source of the author nodes are denoted as k = 2:

i. **Papers(Nodes) Relationship Modeling**

Inspired by the idea of DeepWalk[24], I construct a random walk corpus S on all sources of the graph structures. To treat each random walk path ($v_{11} \rightarrow v_{12} \rightarrow v_{15} \rightarrow \cdots \rightarrow v_{1n}$) as a sentence and each node v_{1n} as a word, I use DeepWalk to train skip-gram models with a generated random walk corpus, obtaining a distributed vector representation for each node. The objective function maximizes the likelihood of the neighboring nodes, given a node v_{1i} for the random walk corpus $s \in S$:

$$\begin{aligned} \mathcal{L}_1 &= \sum_{i=1}^{N} \sum_{s \in S} \log \mathcal{P}(v_{1(i-b)} : v_{1(i+b)} \mid v_{1i}) \\ &= \sum_{i=1}^{N} \sum_{s \in S} \sum_{-b \leq j \leq b, j \neq 0} \log \mathcal{P}(v_{1(i+j)} \mid v_{1i}). \end{aligned} \tag{5.1}$$

where N is the total number of nodes in 1_{th} graph. With a soft-max function, the probability of capturing the surrounding papers $v_{1i-b} : v_{1i+b}$ given a paper v_{1i} is calculated by

$$\mathcal{P}_1(v_{1(i+j)} \mid v_i) = \frac{\exp(v_{v_{1i}}^{\top} \hat{v}_{v_{1(i+j)}})}{\sum_{i=1}^{N} \exp(v_{v_{1i}}^{\top} \hat{v}_{1_v})} \tag{5.2}$$

where $\hat{v}_{v_1}$ is the output representation of the node v_1 in the first source (the paper nodes) of the graph.

ii. **Authors(Nodes) Relationship Modeling**

Following a very similar operation, the probability of finding the surrounding authors $v_{2i-b} : v_{2i+b}$ given a author v_{2i} is calculated by:

$$\mathcal{P}_2(v_{2(i+j)} \mid v_i) = \frac{\exp(v_{v_{2i}}^\top \hat{v}_{v_{2(i+j)}})}{\sum_{i=1}^{N} \exp(v_{v_{2i}}^\top \hat{v}_{2_v})} \tag{5.3}$$

where $\hat{v}_{2_v}$ represents the output representation of the author node v_2 in the second source (the author nodes) of the graph.

iii. **Paper-Contents Correlation Modeling**

I assemble all the text content associated with one node in one line. For example, in the citation data sets, the title and abstract information of the same paper are assembled into one line in the original order of words. Based on Mikolov's work[23], I aim to exploit the neighboring words(context information) within a certain window size, given a particular word. The objective function is achieved by maximizing the below log-likelihood:

$$\mathcal{L}_2 = \sum_{t=1}^{T} log\mathcal{P}(w_{t-b} : w_{t+b} \mid w_t) \tag{5.4}$$

where b is the window size and $w_{t-b} : w_{t+b}$ is a sequence of words which w_t is in the middle of. The conditional probability of sequence $w_{t-b} : w_{t+b}$ given w_t is calculated by:

$$\mathcal{P}(w_{t-b} : w_{t+b} \mid w_t) = \prod_{-b \le j \le b, j \neq 0} \mathcal{P}(w_{t+j} \mid w_t) \tag{5.5}$$

which assumes contextual sequence is independent given word w_t. The conditional probability of w_{t+j} given w_t is calculated:

$$\mathcal{P}(w_{t+j} \mid w_t) = \frac{\exp(v_{wt}^\top \hat{v}_{w_{t+j}})}{\sum_{w=1}^{W} \exp(v_{wt}^\top \hat{v}_w)} \tag{5.6}$$

where v_w and $\hat{v}_w$ are the input and output word vectors of the given word w. Further, given a node v_{ki}, the conditional probability of capturing consecutive

words $w_{t-b} : w_{t+b}$ is:

$$\mathcal{P}(w_j \mid v_t) = \frac{\exp(v_{v_{ki}}^{\top} \hat{v}_{w_j})}{\sum_{w=1}^{W} \exp(v_{v_{ki}}^{\top} \hat{v}_w)} \tag{5.7}$$

where W is the total number of words in the entire graph and $\hat{V}_{w_j}$ is the representation of the word w_j.

iv. **Label Information Correspondence Modeling**

I experimentally prove that label information, which is not leveraged in the majority of current relevant algorithms, could enhance the performance of graph representation in tasks like classification. These results are shown in the next section. I input the label of each document with its corresponding text content to simultaneously learn the label and word vectors. The objective function to discover the document and label information can be calculated by:

$$\begin{aligned} \mathcal{L}_3 = &\sum_{t=1}^{L} log\mathcal{P}(w_{i-b} : w_{i+b} \mid s_i) \\ &+ \sum_{i=1}^{N} log\mathcal{P}(w_{i-b} : w_{i+b} \mid v_i) \end{aligned} \tag{5.8}$$

where L is the set of label information and s_i is the label of node v_i. Following a similar manipulation, the probability of capturing a word sequence given a label s_i is:

$$\mathcal{P}(w_j \mid s_i) = \frac{\exp(v_{s_i}^{\top} \hat{v}_{w_j})}{\sum_{w=1}^{W} \exp(v_{v_{si}}^{\top} \hat{v}_w)} \tag{5.9}$$

v. **Model Assembly**

The k node relationship models for k sources of each graph structure, one node-content correlation model and one label-words correspondence model are assembled and mutual-influenced according to their unique correlation index. The final model inputs a complete multi-relational information graph with the different sources of the graph structures, node contents and label information, defined as $N = \{V_k, E_k, D_k, S\}$. The objective of the UGRA model is to

maximize the log likelihood function:

$$\begin{aligned}\mathcal{L} &= (1-\alpha)\sum_{i=1}^{N} log\mathcal{P}(w_{i-b}:w_{i+b} \mid v_i) \\ &+ (1-\alpha)\sum_{t=1}^{L} log\mathcal{P}(w_{i-b}:w_{i+b} \mid s_i) \\ &+ \alpha\sum_{k=1}^{K}\sum_{i=1}^{N}\sum_{s\in S}\sum_{-b\leq j\leq b, j\neq 0}\log\mathcal{P}(v_{k(i+j)} \mid v_{ki}).\end{aligned} \tag{5.10}$$

where α is the weight for balancing graph structures, text contents and label information, and b is the window size of word sequence. The first term is used to calculate the conditional probability of capturing a sequence of consecutive words $\{w_{i-b} : w_{i+b}\}$ given a word w_j to model the correlation between a paper and its content, while the second is to model the label information correspondence by computing the probability of capturing a sequence of words given a label s_i. Finally different sources of the node relationships are modeled by calculating the sum of the probabilities of the captured neighboring nodes $v_{ki-b} : v_{ki+b}$ in the k_{th} source of graph given a node v_k.

5.3.2 Algorithm Explanation

Given a heterogeneous information graph, the first three steps are corpus-generation work for different sources of information in the graph. Step 1 generates corpus of random walks for each source of the node graphs. Steps 2 and 3 generate Huffman binary trees for content and label corpus to significantly save computation costs when computing soft-max equations in Steps 9 to 11(the details are described in the Optimization section). Steps 3 to 6 initialize the vectors for k sources of the nodes, contents and labels respectively. After this preparation work is done, Steps 9 to 11 iteratively optimize the objective functions explained in Eq.(5.10) to model the relationship between the different sources of the information. Step 9, in particular, discovers the inter-relationship between homogeneous nodes by fixing the word representation $\hat{v}_{w_j}$ and the label representation v_{c_i} while solving Eq.(5.10) to update the node representation. Through a similar manipulation, Steps 10 and 11

Figure 5.2 : The CiteSeerX-Avs data set result comparison. From left to right: visualized graphs from Deepwalk, the Doc2Vec, Deepwalk + Doc2Vec, Tridnr and UGRA. Each color represents a group. The purer the color of a group, the better the performance.

find the node-content relationships and the label correlations by separately solving the hierarchical soft-max function using a stochastic gradient method.

5.3.3 Optimization

Stochastic gradient descent (SGD)[72] is used to model the relationships between different sources of information (homogeneous and heterogeneous nodes, and the correlations between nodes and contents)Eq.(5.10), the UGRA performs many expensive computations for conditional probabilities (Eq.(5.2),Eq.(5.3),Eq.(5.7),Eq.(5.9)). To solve this problem, I took the advantage of hierarchical soft-max [73, 74] which builds Huffman trees instead of nodes vectors (authors, papers and words).The path to the leaf node $v_{v_{ki}}$ can then be represented with a sequence of vertices passing through $(s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \cdots \rightarrow s_{end})$, where s_0 is the root of the tree and s_{end} is the target node $v_{v_{ki}}$. This can be computed by the following:

$$\mathcal{P}(v = v_{v_{ki}}) = \prod_{t=1}^{end} \mathcal{P}(s_t \mid v_{v_{ki}}) \tag{5.11}$$

I can further model $\mathcal{P}(s_t \mid v_{v_{ki}})$ with a binary classifier as a sigmoid function, and the time complexity reduces from $\mathcal{O}(V)$ to $\mathcal{O}(n \log n)$.

Table 5.1 : Enterprise Social Network Datasets Used in the Paper

Data Set	Paper Nodes	Author Nodes	Paper Edges	Author Edges	Content words	Labels
DBLP	56,503	58,279	106,752	142,581	3,262,885	4
CiteSeerX-Avs	18,720	40,139	54,601	41,458	2,649,720	5
CiteSeerX-M10	10,310	21,289	77,218	21,966	1,516,893	10

5.4 Experimental Study

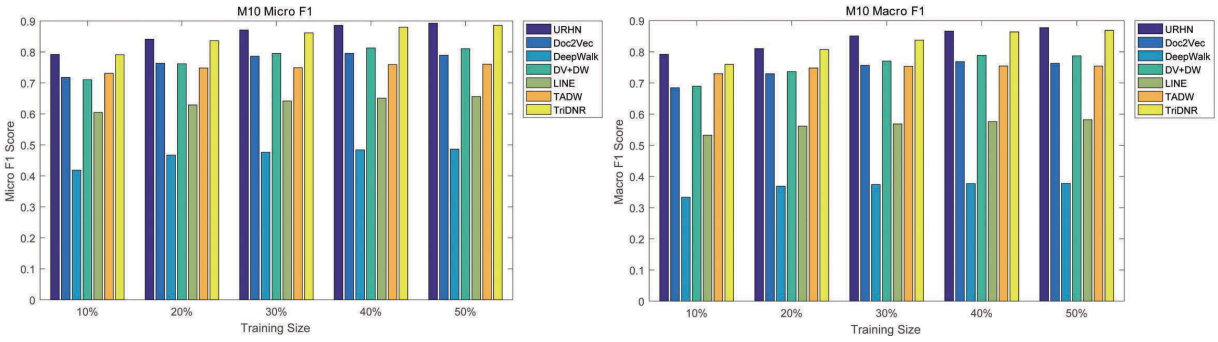

Figure 5.3 : Average performance on different percentage of training data of UGRA and six baselines.

The experiments are based on three real world publication data sets. The reported results show (1) the UGRA outperforms the state-of-the-art graph representation algorithms based on different techniques in machine learning tasks like node classification; (2) graph representation by the UGRA can represent different sources of nodes in heterogeneous graphs and this unique characteristic can be very effective

Algorithm 2 UGRA: Universal Graph Representation for Multi-relational Information Graphs

Require:

$N = \{V_k, E_k, D_k, S\}$: A heterogeneous information graph;

n: Expected Number of Dimension;

b: Size of Windows;

T: Number of Iterations;

K: Number of sources of graph structure;

len: Walk Length of Random Walk

Generate Random Walk Corpus S_k from k_{th} source of graph

Generate a Vocabulary Binary Tree T_w

Generate Node Binary Trees T_{v_k}

Initial input vector $v_{v_{ki}} \in \mathbb{R}^r$ and output vector $\hat{v}_{v_{ki}} \in \mathbb{R}^r$ for each node $v_{ki} \in V_k$

Initial output vector $\hat{v}_{w_j} \in \mathbb{R}^r$ for each word $w_j \in W$

Initial input vector $v_{c_i} \in \mathbb{R}^r$ for each label class c_i

for iterator = 1,2,3, $\cdots\cdots$, T **do**

  for k = 1,2, $\cdots\cdots$, K **do**

    Fix $\hat{v}_{w_j}$ and v_{c_i}, solve Eq.(5.10) to update $v_{v_{ki}}$ and $\hat{v}_{v_{ki}}$; //inter-node relationship

    Fix $v_{v_{ki}}$ and v_{c_i}, solve Eq.(5.10) to update $v_{v_{ki}}$ and $\hat{v}_{w_j}$; // node content correlation

    Fix all k $v_{v_{ki}}$ and $\hat{v}_{v_{ki}}$, solve Eq.(5.10) to update v_{c_i} and $\hat{v}_{w_j}$; // label and content correlation

  end for

end for

return $v_{v_i} \in \mathbb{R}^r$,$v_k \in V$,$\forall v_i \in V$

and useful in real world applications. Table5.1 provides summarized details of data sets.

5.4.1 Experimental Settings

Datesets: I built multi-relational heterogeneous information graphs from three real world publication data sets [75, 7] which each consist of two sources of nodes (authors and papers), two sources of node contents (titles and abstracts), each containing natural label information within the data sets. Different sources of graph structure and node content were associated by a unique index according to their correlations. For example, all cited papers $paper_A$, all authors of $paper_A$ and its title and abstract information have the same unique index. Further, I combined two sources of the node content (title and abstract) into one line associated with the same index, for simplicity, as both sources of contents were text in this case. However, the different sources of content can be separately in the UGRA model.

Baselines

The UGRA was compared to the following graph representation algorithms for paper node classification:

(1)Graph exploration methods:

LINE[15]: The state-of-the-art graph representation method, designed for embedding very large information graphs into low-dimensional vector spaces.

DeepWalk[24]: A novel approach for learning latent representations of vertices in a graph. These latent representations encode social relations in a continuous vector space

(2)Text modeling methods:

Doc2Vec[23]: An unsupervised algorithm that learns fixed length feature representations from variable-length pieces of texts, such as sentences, paragraphs, and documents.

(3)Methods that consider both graph structure and content:

TriDNR[25]: A deep graph representation model that simultaneously considers graph structure and node content within a neural network architecture

DeepWalk + Doc2Vec: An approach that simply links different representations learned from DeepWalk and Doc2Vec

TADW[14]: An approach that incorporates the text features of vertices into graph representation and learns in a matrix factorization framework.

Evaluation

I conducted paper node classification and employed Macro_F1 and Micro_F1 [76] to evaluate the performance of the UGRA with all baselines. I varied the percentages(from 10% to 50%) of the random nodes with labels and the rest will be unlabeled. The complete multi-relational information graph will be fed to learn graph representation while getting node vectors from baseline algorithms. A linear SVM classifier was applied to perform the classification on learned vectors. I repeated each experiment 20 times under the same parameter setting and used the mean value as the final score for each algorithm.

Parameter Settings

Training sizes impacted the performance significantly. I varied the training size from 10% to 50% for the UGRA and the baselines to learn the accuracy trends. The updated weights for the models for different sources of information were practically set to 0.8.

5.4.2 Experimental Results

These experiments compare graph representation algorithms that independently consider graph structure[15, 24], node content[23] and both structure and content[25,

Table 5.2 : Macro F1 DBLP

Training size	UNRA	TriDNR	Doc2Vec(DV)	DeepWalk(DW)	DV+DW	LINE	TADW
10%	**0.732**	0.715	0.638	0.385	0.659	0.431	0.670
20%	**0.732**	0.727	0.644	0.320	0.669	0.439	0.698
30%	**0.736**	0.730	0.643	0.317	0.668	0.445	0.709
40%	**0.739**	0.736	0.643	0.308	0.668	0.446	0.711
50%	**0.742**	0.738	0.650	0.320	0.675	0.446	0.712

Table 5.3 : Micro F1 DBLP

Training size	UNRA	TriDNR	Doc2Vec(DV)	DeepWalk(DW)	DV+DW	LINE	TADW
10%	**0.791**	0.777	0.717	0.455	0.728	0.488	0.662
20%	**0.792**	0.787	0.722	0.478	0.737	0.494	0.670
30%	**0.795**	0.788	0.722	0.478	0.736	0.498	0.711
40%	**0.797**	0.793	0.722	0.479	0.739	0.499	0.705
50%	**0.798**	0.798	0.724	0.482	0.740	0.511	0.716

14], by varying the training size from 10% to 50% with three different complete publication information graph in a paper nodes classification task. UGRA achieved outstanding performance, increasing the accuracy from 3% compared to TriDNR, 14.6% compared to Doc2Vec, 69.8% compared to LINE to 132% compared to DeepWalk on the DBLP data set. The Macro_F1 and Micro_F1 scores are illustrated in Tables 5.2 and 5.3, and the average performance on the CiteSeerX-M10 data set is demonstrated via the bar charts in Fig.6.6. The visualization for the paper nodes classification on CiteSeerX-Avs data set (see Fig. 5.2) intuitively indicates similar results. Each color represents a group and it is obvious that nodes in different groups mix significantly in term of DeepWalk, while the colors in each group are relatively clearer for tridnr, Doc2Vec, but not as clear as the result from UGRA.

These results clearly indicate that methods that only consider one view of heterogeneous information graphs, or a simple combination of the methods (Doc2Vec + DeepWalk) are sub-optimal for multi-relational information graph. The strategy UGRA employs by training different sources of information and using theor inner relations to update each other, dramatically improves performance in node classification.

5.4.3 Case Study

The generated UGRA representation has been tested for its ability to detect the most related heterogeneous nodes in multi-relational graphs, whatever the source of given a node. This is achieved by computing the cosine similarity between a simple mean of the projection weight vectors of the given node and the vectors for each node in the model. Given the similarity scores, I can directly capture the heterogeneous nodes that are most related to given node regardless of the source. As the experiments were based on publication data sets, I derived three tests for the proposed UGRA representation: (1) given a paper, return the most related

authors and other papers; (2) given an author, return the most related papers and other authors; and (3) given more than one nodes, returns the most other related heterogeneous nodes.

First, I input a paper into the representation, *Boolean functions and artificial neural networks* published by Martin Anthony in CDAM 2003. The most related papers are other versions of this paper which has been slightly renamed, *Connections between Neural Networks and Boolean Functions*, also first-authored by M. Anthony. The similarity score was around 0.961%. The most related author is, with no surprise, M. Anthony with a 0.939% similarity. Despite the paper's other versions and co-authors of the paper, the output included a paper by the same author: *Probabilistic Analysis of Learning in Artificial Neural Networks: The PAC Model and its Variants* and other authors with same research direction like Dr. Simone Fiori who published *Topics in Blind Signal Processing by Neural Networks*.

I also input the author Dr. Steve Lawrence and the most related author nodes were three of his co-authors Sandiway Fong(0.960% similarity), A. C. Tsoi(0.885% similarity) and C. Lee Giles (0.895% similarity). Together they have published papers like *On the applicability of neural network and machine learning methodologies to natural language processing* (0.879% similarity) and *Natural language grammatical inference with recurrent neural networks* (0.813% similarity)), which were also included in the outputs.

Despite the citation and co-author relationship, papers and authors with very similar research direction but no direct citation or co-work relationships were also output. For example, a paper, *Unsupervised Learning in Recurrent Neural Networks*, was input to return the author Felix A. Gers who published the paper *Kalman filters improve LSTM network performance in problems unsolvable by traditional recurrent nets*. The output author was not in the author list of the given paper, meanwhile the

papers from the given and output had a very similar research direction - recurrent neural network - with no citation relationship to one another. Through checking these two papers, I found they both specifically focus on long short term memory (LSTM).

The most interesting results came when I simultaneously input more than one node into the UGRA representation. Two papers were input: *Unsupervised Learning in Recurrent Neural Networks* with no mention of natural language processing in its title or the abstract; and *Empirical Learning of Natural Language Processing Tasks* again not related to neural networks. The top returned authors were the authors of given two papers : Walter Daelemans, Magdalena Klapper-Rybicka, Nicol N. Schraudolph, Antal Van Den Bosch, Ton Weijters. In terms of the most related papers, beyond the other versions of the given papers, the results returned were papers that use neural network approaches for natural language processing like *Natural language processing with subsymbolic neural networks.*

I also compared the results from the UGRA with the baseline algorithms. The results are listed in Table 5.4. I input a paper, *Learning in Neural Networks* (published by Dr. J. Stephen Judd), into each representation and selected the top five most related papers. The results from the UGRA were all relevant to learning in neural networks, while the resulting papers from Doc2Vec and DeepWalk were not really related to the input paper.

All tests were conducted on the same UGRA representation.

5.5 Summary

In this paper, I proposed a UGRA, a novel graph representation algorithm leveraging multiple-sources of information in multi-relational heterogeneous graph data. My survey shows current graph representation works normally only consider one

Table 5.4 : outputs from representations. The matched is marked with ⊙

Input: Learning in Neural Networks
URHN:
1. Adjoint-Functions and Temporal Learning Algorithms in Neural Networks ⊙
2. Bit-Serial Neural Networks ⊙
3. An Information-theoretic Learning Algorithm for Neural Network Classification ⊙
4. Polynomial Time Algorithms for Learning Neural Nets ⊙
5. Training of Large-Scale Feed-Forward Neural Networks ⊙
Doc2Vec:
1. Non-Cumulative Learning in METAXA.3
2. Learning Filaments
3. Learning of Kernel Functions in Support Vector Machines
4. Incremental Learning in SwiftFile
5. Learning While Searching in Constraint-Satisfaction-Problems
DeepWalk:
1. Estimating image motion from smear: a sensor system and extensions
2. Inferring 3D Volumetric Shape of Both Moving Objects and Static Background Observed by a Moving Camera
3. Secure face biometric verification in the randomized Radon space
4. An Ensemble Prior of Image Structure for Cross-Modal Inference
5. Closed Non-derivable Itemsets

aspect of the information - graph structure or node content - within a whole graph, while the learned representation only represents one source of the nodes. The graph representation from the UGRA learned from all sources of information in a graph, and represents all sources of nodes in the graph. Experimental results practically show that the UGRA outperforms the state-of-the-art peer algorithms, and that the characteristic of representing all sources of heterogeneous nodes can be very effective and useful in real applications. Our key contributions are summarized by three points:(1) I propose a novel graph representation model that simultaneously leverages different sources of graph structures, node contents and labels in one heterogeneous graph; (2) I propose a novel graph representation algorithm that simultaneously generates a representation for all sources of nodes in a heterogeneous graph; and (3) I conduct a thorough suite of experiments from different perspectives on real world data sets. The results of our accuracy tests prove the effectiveness of our proposed UGRA and a case study practically shown the implementation of our UGRA in a real world setting.

Chapter 6

Co-Clustering Enterprise Social Graphs

6.1 Motivations

Due to advancements in web techniques, and the rapid growth of social graphs, many enterprises are seizing the opportunity to offer timely feedback and provide tailored services to their social audiences. In this article, I define a business information graph (BIG) as a social graph where people share similar interests in business products or activities. There are two main kinds of BIGs [77]. Private BIGs are built for the employees or members of one specific company or industry. Public BIGs, such as Twitter, Facebook, and LinkedIn, are external platforms that allow all the *social audiences* of an enterprise (i.e., business partners and followers) to post their views (textual content) or interact (links). In this paper, I concentrate on public BIGs, since they are more customer-oriented, and are generally considered to be good tools for improving business intelligence. According to Barnetta [78], the majority of Fortune 500 companies, such as Toyota, IBM, DE, and Sears, have significantly optimized their business work flow by using of BIGs. For example, companies are now using LinkedIn to gain strategic advantage [79], and there is a significant growth in online recruitment and marketing strategies since 2007 [80]. Moreover, according to [81], 32% of social enterprises increased their volume of business in 2012. Public BIGs have also been successfully used for other objectives, such as advertising [82], online recruitment [79], marketing [80], decision making [83], branding, and customer relationship management [84, 85, 86].

Figure. 6.1 provides an example of Sony Pictures' business intelligence work-flow

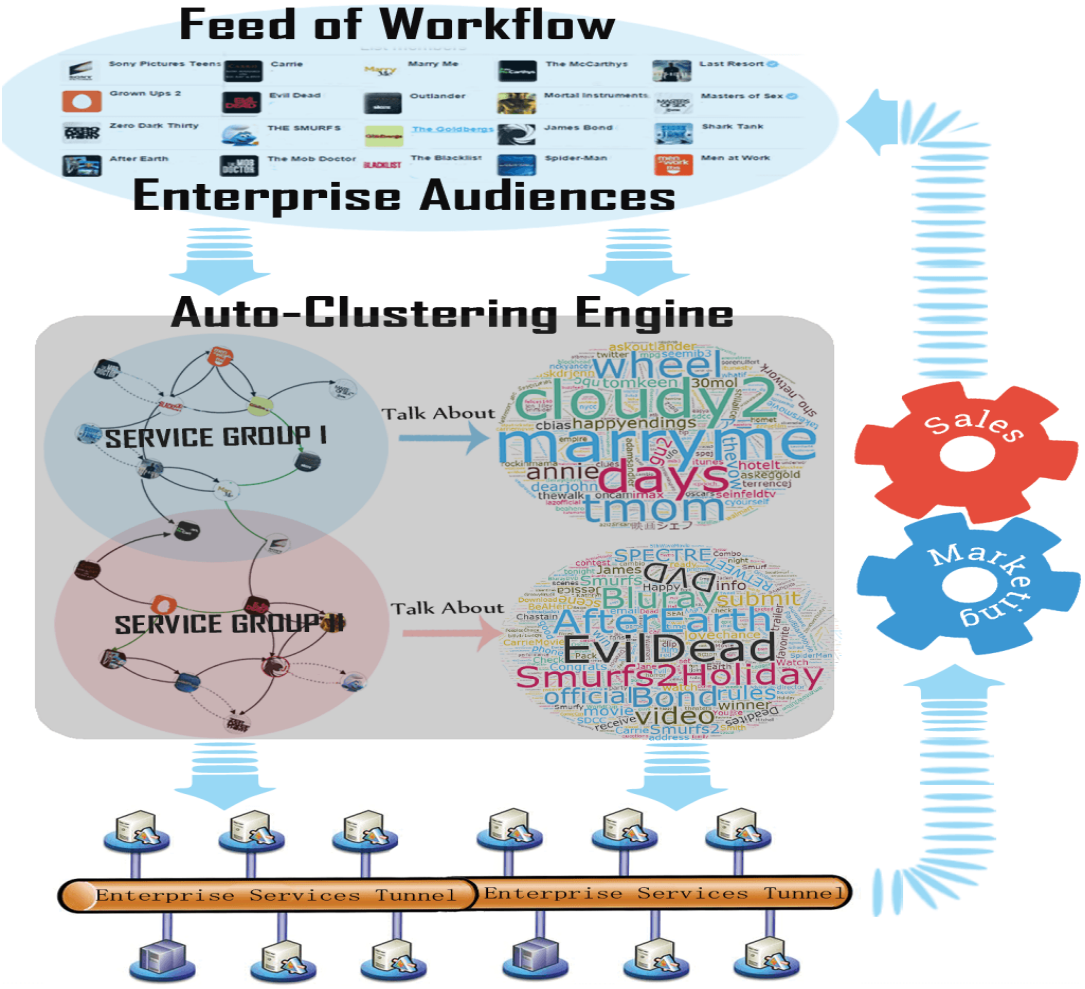

Figure 6.1 : An example of the business intelligence workflow for Sony Pictures Entertainment. The loop contains three main sections: a. Automatic Audiences (followers) and their interests clustering engine; b. Enterprise Service Tunnel; c. Sales and Marketing Analysis.

on Twitter. In this particular example the social audience is their Twitter followers, and all their information, including their friendships, profiles, and posts, is fed into an automatic clustering engine that segments the audiences into SACs, with each SAC sharing similar interests or preferences for Sony Pictures' products. Each SAC is referred to an appropriate enterprise service tunnel, based on their shared common preferences, and feedback is delivered to sales and marketing to further enhance products and services for each group of users. Obviously, the automatic clustering engine is the key to the entire business intelligence process. Identifying the SAC, especially their shared interests, can assist enterprises in offering tailored services to targeted users, and dramatically reduce costs.

In this article, I focus on a data mining approach to automatically discover these SACs defined in the abstract. For example, in Fig. 6.1, users that are interested in discussing movies might form a SAC, SERVICE GROUP I, whiles users interested in television might form another, SERVICE GROUP II.

Intuitively, finding a SAC seems easy to solve using simple clustering algorithm based on graph topology, such as AgmFit [87] and BigClam [88] (both are available on SNAP *). However, due to the limitations of graph topology information, these algorithms do not provide an additional understanding of each SAC. Further, these approaches assume that the users (here after referred to as nodes) in the information graphs are densely connected, which is not the case in BIGs.

To provide further insights into each group's function, relational topic discovery based algorithms could be employed, such as the methods in [89, 69, 90] or the nonnegative matrix factorization methods in [37, 91, 92]. However, these approaches assume that the nodes and graphs are highly consistent, and that the graph follows manifold assumptions [37, 91, 92]. In other words, data is distributed smoothly

*http://snap.stanford.edu/

over a low-dimensional space so the manifold regularization delivers good clustering performance. Unfortunately, the consistency between nodes and graphs is relatively low in BIGs. For example, many users in the BiGs may follow the official social media account of an enterprise because of a specific product, otherwise these users barely share other common interests in the social media networks. In this case, the existing algorithms which assume linked nodes always share similar attributes may have a sub-optimal performance.

Beyond the above disadvantages, even if existing tools could help find SACs, they would normally find a single hard cluster membership for each node. Consequently, nodes could not have two or more roles in the graph and this does not fit many business models. In addition, these tools would certainly rely on a user-specified threshold to find the appropriate number of SACs. In reality, nodes often have many roles, the number of SACs is hard to define as it is data-driven, and both have strong seasonal or event-driven characteristics. Hence, clustering needs to be overlapped, and the optimal number of SACs needs to be determined automatically.

In this article, I propose CBIG, a novel clustering method to discover SACs in BIGs. Through a non-negative matrix factorization (NMF) method, I factorize graph information with node content and integrate the clustering results via a consensus principle. I further propose a heuristic approach to set the threshold for overlapping clustering. The optimal number of SACs is automatically determined by minimizing the reconstructed error with a hold-out set method.

Our main contributions are summarized below:

i. I propose a novel algorithm to discover SACs in BIGs, to advancing existing works that focus on generic graph information such as personal or academic graph analysis.

ii. I propose an effective overlapping co-clustering algorithm that simultaneously

clusters social audiences and features into an automatically-determined optimal number of SACs, to provide a deeper understanding of the functional roles customers play in a company's business model.

iii. I conducted experiments on 13 real-world enterprise data sets, and the results indicate that CBIG achieves outstanding performance compared to several state-of-the-art approaches.

6.2 Most Related Works to Co-clustering Enterprise Graph

Graph or network analytics has attracted much attention in recent years [93, 94, 95, 96, 97, 3]. Business information graphs have a wide range of applications in social media support systems, marketing [98, 99], advertising [100], decision making [83], and customer relationship management [85]. Understanding the social audiences within BIGs plays an important role in business intelligence and decision making. In [101], Lo *et.al* proposed a method to rank high-value social audiences on Twitter. In this article, I propose clustering the social audiences within BIGs to understand SACs. From a technical perspective, SAC discovery in BIGs is closely related to clustering problems in a graph setting.

Many existing clustering algorithms only leverage textual information. For example, Dam [102] applies an MCA K-means approach to cluster Facebook users with collected profile information. Alternatively, community detection approaches, such as AgmFit [87], BigClam [88], GDPSO[103] and [99, 93], are also available for clustering graph data by purely using graph structures. However, these approaches only consider one aspect of the information, which may lead to sub-optimal results.

Other approaches do consider both textual information and graph topologies [37, 91, 11, 104]. Co-clustering approaches, such as [37, 91, 92], factorize the user-feature matrix using NMF and use the graph structure to regularize the objective

function. Relational topic model approaches, such as [89, 69, 90], attempt to simultaneously use the structure and the textual information for clustering. However, these approaches assume the graph structures and the nodes are closely consistent, which may not be true for BIGs. NMF-based methods [37, 91, 92] also assume that the data resides in a manifold, whereas graph data is known to follow power law distributions.

Recently, researchers have proposed new algorithms for overlapping graph data clustering [68, 105]. Leskovec and Mcauley [68] consider the graph structure of friendships as an ego graph and modeled node memberships to multiple circles. Circle-specific user metrics are learned to enable overlapping clustering. In [105], the authors proposed the Censa algorithm to model the interaction between the graph structure and a node's attributes. Although [68, 105] fulfill the overlapping clustering task and specifies which attributes are useful for forming a community, they do not group features into clusters to give better interpretations of the topics that users are interested in.

Compared to exiting studies, our research advances this work from generic information graph analysis into BIG mining. Our framework has the following attractive properties: (1) it is specifically designed for BIGs, (2) it enables increased freedom to address the inconsistencies in the graph structure and node content, and (3) it provides interpretable meanings for SACs.

6.3 Problem Definition

A BIG is defined as $\mathsf{G} = \{\mathsf{V}, \mathsf{E}\}$. $\mathsf{V} = \bigcup\{\mathsf{v_i}\}_{\mathsf{i}=1,\cdots,\mathsf{n}}$ denotes the set of audiences in a BIG, and $\mathsf{e_{i,j}} =< \mathsf{v_i}, \mathsf{v_j} > \in \mathsf{E}$ indicates an edge encoding the link between two audiences. A matrix $\mathbf{W_s} \in \mathbb{R}_+^{\mathsf{n}\times\mathsf{n}}$ is applied to further the simplify graph information. If a link between audiences $\mathsf{v_i}$ and $\mathsf{v_j}$ exists, $[\mathbf{W_s}]_{\mathsf{ij}} = 1$; otherwise, $[\mathbf{W_s}]_{\mathsf{ij}} = 0$. The user-feature vector $\mathbf{X}_{\cdot\mathsf{i}} \in \mathbb{R}_+^{\mathsf{m}}$ is associated with each audience $\mathsf{v_i}$. Thus, the user-

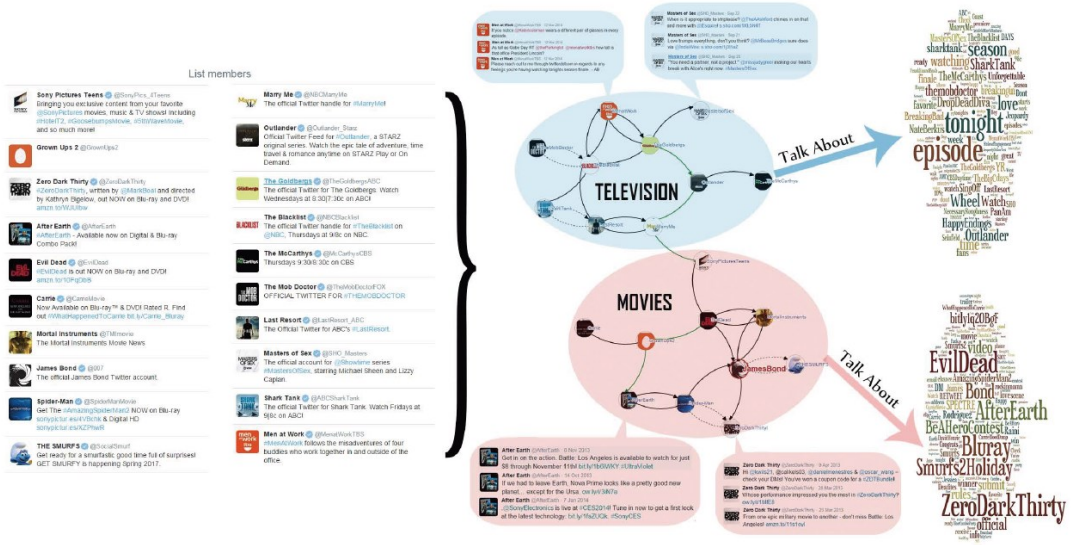

Figure 6.2 : An illustrated example of Sony Pictures company.

feature correlation of all audiences is embedded within a matrix $\mathbf{X} \in \mathbb{R}_{+}^{m \times n}$, where $\mathbf{X} = [\mathbf{X}_{\cdot 1}, \cdots, \mathbf{X}_{\cdot n}]$ and each column in $\mathbf{X}$ indicates a node instance.

In this article, the **aim** of SAC discovering and understanding SACs is to automatically and simultaneously segment the audiences and features into an optimal number of clusters. Specifically, the audiences $V = \bigcup\{v_i\}_{i=1,\cdots,n}$ in a BIG are clustered into k SACs $\mathcal{C} = [C_1, \cdots, C_k]$. To provide insights into the function of SACs in a BIG, the features are also clustered into q clusters $\mathcal{Q} = [Q_1, \cdots, Q_q]$. Every feature in a cluster represents a particular interest of audience in the BIG. To this end, a possible clustering result for the audience would be represented as $\mathbf{G} = [\mathbf{G}_{1\cdot}; \cdots; \mathbf{G}_{n\cdot}] \in \mathbb{R}_{+}^{n \times k}$ where $\mathbf{G}_{ij}$ corresponds to the degree of membership of the i-th audience for cluster C_j. Similarly, a possible clustering result for a feature would be indicated by the matrix $\mathbf{F} \in \mathbb{R}^{m \times q}$.

It is worth noting that a SAC is a cluster of audiences with interacting links and common content. Compared to existing community detection algorithms [87], the proposed SAC discovery method has two unique features. (1) I allow nodes to

overlap, *i.e.*, $C_i \cap C_j \neq \emptyset$., which reflects situations where a node plays different roles in different SACs; and (2) my method automatically determines the optimal amount of SACs given the data provided. The Fig. 6.2 provides an example about the workflow to detect SACs: Followers and business partners are segmented into two groups, say *Television* and *Movies*. The followers in each group interact with each other and show similar interest. For instance, people in *Television* group are highly interested in "SharkTank", "Outlander" and "MarryMe", which are popular TV shows of Sony Pictures. For the users in *Movies* group, they are mainly interested in the movies Sony Pictures made like "Evil Dead", "After Earth", "Zero Dark Thirty".

6.4 CBIG Algorithm

This section outlines the technical details of CBIG for SAC discovery, and then extends CBIG into an overlapping clustering setting. Fig. 6.3 illustrate the overall framework of the CBIG algorithm. Given a business information graphs, CBIG factorizes three channels of information via a consensus principle, i.e., the node-feature content matrix $\mathbf{X}$, the graph topology structure $\mathbf{W_s}$, and the feature-feature correlations $\mathbf{W_f}$. After the factorization, CBIG cluters the nodes into ($\mathbf{G}$) and the features into ($\mathbf{F}$), and further extends $\mathbf{G}$ for overlapping clustering.

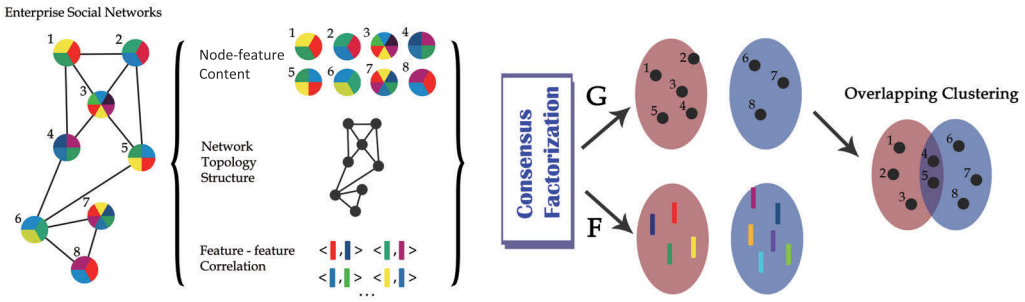

Figure 6.3 : The overall framework of the CBIG algorithm.

6.4.1 CBIG for Social Audience Cluster Discovery

In BIGs, data is obtained from two main sources: graph topological structures and graph node-content. All audiences are represented by a user-feature matrix X to derive content values for the user-features of the BIG. To determine the links between audiences, pairwise audience connections inside the BIG are represented as a $\mathsf{n} \times \mathsf{n}$ matrix $\mathbf{W_s}$. More significantly, as the audiences in a SAC share similar interests, I explicitly explore the correlations between the audience's features ($\mathbf{W}_f$), and use such correlations in the clustering process.

To discover the SACs, I factorize three channels of information: the feature correlations matrix ($\mathbf{W}_f$), the user-feature ($\mathbf{X}$), and the graph structure ($\mathbf{W}_s$). $\mathbf{X}$, $\mathbf{W}_s$, and $\mathbf{W}_f$ are each factorized separately; then consensus is enforced on the results. The overall framework for CBIG is shown in Fig.6.3. The essential difference between our framework and existing NMF-based algorithms, such as [37, 91, 92], is that I comprehensively leverage the graph information from different perspectives, while previous models only factorize the user-feature matrix.

User-feature Matrix Factorization:

The user-feature matrix $\mathbf{X}$ provides a tabular mapping between users and features. Using NMF [106], I factorize $\mathbf{X}$ into two non-negative matrices $\mathbf{G}$ and $\mathbf{F}$ by minimizing the error function using the Frobenius norm:

$$\underset{\mathbf{F},\mathbf{G}}{\operatorname{argmin}} J_1 = ||\mathbf{X} - \mathbf{F}\mathbf{G}^\top||_F^2, \quad s.t.\ \mathbf{F} \geq 0, \mathbf{G} \geq 0, \tag{6.1}$$

where $\mathbf{F}\mathbf{G}^\top$ is the approximation of $\mathbf{X}$. The clustering results of users and features are naturally exposed in $\mathbf{G}$ and $\mathbf{F}$ [107, 37, 91, 92]. For example, a node v_i can be assigned to the cluster $C_{j^\star}$, where $j^\star$ is determined by Eq.(6.2):

$$j^\star = \underset{j=1,\cdots,k}{\operatorname{argmax}} \mathbf{G}_{i,j} \tag{6.2}$$

In reality, because the two-factor NMF in Eq.(6.1) is restrictive, i.e., the number of clusters q and k have to be equal, I have introduced an additional factor $\mathbf{S} \in \mathbb{R}_{+}^{q \times k}$ to accommodate the different scales of $\mathbf{X}$, $\mathbf{F}$ and $\mathbf{G}$. This leads to an extension of NMF, called NMTF [108, 109]:

$$\underset{\mathbf{F},\mathbf{G}}{\operatorname{argmin}} J_2 = ||\mathbf{X} - \mathbf{F}\mathbf{S}\mathbf{G}^{\top}||_F^2, \quad s.t.\ \mathbf{F} \geq 0, \mathbf{G} \geq 0, \tag{6.3}$$

$\mathbf{S}$ provides increased degrees of freedom such that the low-rank matrix representation remains accurate, while the value of q and k can be different. The mapping information between node clusters is recorded in $\mathbf{S}$.

Graph Topology Structure Matrix Factorization:

The adjacency matrix $\mathbf{W}_s$ holds pairwise user connections, which provides topological information to discover the similarity between the users for co-clustering. In practice, I factorize the matrix $\mathbf{W}_s$ into an $n \times k$ matrix $\mathbf{G}_s$, and its transposition $\mathbf{G}_s^{\top}$, where $\mathbf{G}_s \in \mathbb{R}_{+}^{n \times k}$ is an indicator matrix that shows the potential clustering results by only leveraging topological information:

$$\underset{\mathbf{G}_s}{\operatorname{argmin}} J_3 = ||\mathbf{W}_s - \mathbf{G}_s\mathbf{G}_s^{\top}||_F^2, \quad s.t.\ \mathbf{G}_s \geq 0, \tag{6.4}$$

It is worth mentioning that $\mathbf{G} \in \mathbb{R}^{n \times k}$ in J_2 and $\mathbf{G}_s \in \mathbb{R}^{n \times k}$ in J_3 each contains separate factorization results for all the graph nodes but from different perspectives. In this way, I factorize $\mathbf{W}_s$ and $\mathbf{X}$, while retaining the maximum freedom to find the optimal results. The consensus function will later enforce consensus in these two sets of results to produce the optimal outcome.

Feature Correlation Matrix Factorization:

To enhance the feature clustering performance, I also capture pairwise feature correlations using the matrix $\mathbf{W}_f \in \mathbb{R}_{+}^{m \times m}$. I assume that the features $\mathbf{X}_{j\cdot}$ and $\mathbf{X}_{i\cdot}$ will be assigned to the same feature cluster, when they are highly associated, for

example, commonly co-occurring words. Thus, correlation measurement approaches, such as a neighbor-based method [91] or heat kernels [110], are the reasonable ways to construct $\mathbf{W}_f$. In our experiments, I simply apply a linear kernel $[\mathbf{W}_f]_{ij} =< \mathbf{X}_{i\cdot}, \mathbf{X}_{j\cdot} >$, where $\mathbf{X}_{i\cdot}$ indicates the embedding of the i_{th} column of features across all users, while $< \mathbf{X}_{i\cdot}, \mathbf{X}_{j\cdot} >$ measures the similarity between $\mathbf{X}_{i\cdot}$ and $\mathbf{X}_{j\cdot}$.

The factorization of the feature matrix $\mathbf{W}_f$ is similar to Eq.(6.4):

$$\underset{\mathbf{F}_f}{\operatorname{argmin}} J_4 = ||\mathbf{W}_f - \mathbf{F}_f\mathbf{F}_f^\top||_F^2, \quad s.t.\ \mathbf{F}_f \geq 0, \tag{6.5}$$

Consensus Factorization:

Through the above factorizations, J_2, J_3 and J_4 each provide the clustering results for the entire graph from different sources. To unify the results, I jointly formulate J_2, J_3 and J_4 into a single objective with the consensus function:

$$\begin{aligned} J_5 = &||\mathbf{X} - \mathbf{F}\mathbf{S}\mathbf{G}^\top||_F^2 + \alpha||\mathbf{W}_s - \mathbf{G}_s\mathbf{G}_s^\top||_F^2 + \beta||\mathbf{W}_f - \mathbf{F}_f\mathbf{F}_f^\top||_F^2 \\ &+ \rho(||\mathbf{G} - \mathbf{G}_s||_F^2 + ||\mathbf{F} - \mathbf{F}_f||_F^2), \\ &s.t.\ \mathbf{F} \geq 0, \mathbf{G} \geq 0, \mathbf{F}_f \geq 0, \mathbf{G}_s \geq 0, \end{aligned} \tag{6.6}$$

The aim of Eq. (6.6) is to factorize $\mathbf{W}_f$, $\mathbf{X}$, and $\mathbf{W}_s$ separately, and enforce consensus between each. For example, $||\mathbf{G} - \mathbf{G}_s||_F^2$ minimizes the difference between $\mathbf{G}$ and $\mathbf{G}_s$ which represents the potential clustering results from the user-features and the topological structure, respectively. Similarly, $||\mathbf{F} - \mathbf{F}_f||_F^2$ enforces that $\mathbf{F}$ should be maximally consistent with $\mathbf{F}_f$. α and β are responsible for balancing each factorization. ρ is a consistency trade-off. Intuitively, a large ρ would lead to $\mathbf{G}$ that is close to $\mathbf{G}_s$ and an $\mathbf{F}$ that is close to $\mathbf{F}_f$. A small ρ would lead to an independent $\mathbf{G}$ and $\mathbf{G}_s$. In summary, our approach provides an increased degree of freedom to exploit the inconsistencies between the contents and the topological structure, a typical characteristic of BIGs as discussed in the Introduction.

Algorithm Optimization

The objective function Eq.(6.6) is processed with regard to $\mathbf{F}_f$, $\mathbf{F}$, $\mathbf{G}_s$, $\mathbf{G}$, and $\mathbf{S}$, which is not convex when simultaneously considering all variables. In this case, I optimize the Eq.(6.6) in terms of one variable while fixing the other variables, repeating the procedure until the function converges.

If I optimize $\mathbf{G}$ first and keep the others fixed, the Lagrange function for Eq. (6.6) is

$$L = ||\mathbf{X} - \mathbf{F}\mathbf{S}\mathbf{G}^\top||_F^2 + \rho||\mathbf{G} - \mathbf{G}_s||_F^2 + \lambda_G\mathbf{G} \tag{6.7}$$

By making the partial derivatives with respect to $\mathbf{G}$ zero, I have

$$\frac{\partial L}{\partial \mathbf{G}} = -2\mathbf{X}^\top\mathbf{F}\mathbf{S} + 2\mathbf{G}\mathbf{S}^\top\mathbf{F}^\top\mathbf{F}\mathbf{S} + 2\rho\mathbf{G} - 2\rho\mathbf{G}_s + \lambda_G = 0 \tag{6.8}$$

which leads to an update of $\mathbf{G}$ with the following rule:

$$\mathbf{G} \leftarrow \mathbf{G} \odot \frac{\mathbf{X}^\top\mathbf{F}\mathbf{S} + \rho\mathbf{G}_s}{\mathbf{G}\mathbf{S}^\top\mathbf{F}^\top\mathbf{F}\mathbf{S} + \rho\mathbf{G}}; \tag{6.9}$$

Here "$\odot$" means the entry-wise product (performed with ".*" in MatLab), *i.e.* $(\mathbf{A} \odot \mathbf{B})_{ij} = (\mathbf{A})_{ij} \cdot (\mathbf{B})_{ij}$. Similarly, I can update $\mathbf{G}_s$, $\mathbf{F}_f$, $\mathbf{F}$, $\mathbf{S}$ as follows:

$$\mathbf{F} \leftarrow \mathbf{F} \odot \frac{\mathbf{X}\mathbf{G}\mathbf{S}^\top + \rho\mathbf{F}_f}{\mathbf{F}\mathbf{S}\mathbf{G}^\top\mathbf{G}\mathbf{S}^\top + \rho\mathbf{F}}; \tag{6.10}$$

$$\mathbf{G}_s \leftarrow \mathbf{G}_s \odot \frac{\rho\mathbf{G} + 2\alpha\mathbf{W}_s^\top\mathbf{G}_s}{2\alpha\mathbf{G}_s\mathbf{G}_s^\top\mathbf{G}_s + \rho\mathbf{G}_s}; \tag{6.11}$$

$$\mathbf{F}_f \leftarrow \mathbf{F}_f \odot \frac{\rho\mathbf{F} + 2\beta\mathbf{W}_s^\top\mathbf{F}_f}{2\beta\mathbf{F}_f\mathbf{F}_f^\top\mathbf{F}_f + \rho\mathbf{F}_f}; \tag{6.12}$$

$$\mathbf{S} \leftarrow \mathbf{S} \odot \frac{\mathbf{F}^\top\mathbf{X}\mathbf{G}}{\mathbf{F}^\top\mathbf{F}\mathbf{S}\mathbf{G}^\top\mathbf{G}}, \tag{6.13}$$

These parameters are iteratively updated until convergence to obtain the clustering result matrices:

$$\hat{\mathbf{F}} = \mathbf{F} + \mathbf{F}_f; \quad \hat{\mathbf{G}} = \mathbf{G} + \mathbf{G}_s \tag{6.14}$$

A user v_i is assigned to cluster $C_{j^\star}$ where $j^\star$ is defined as:

$$j^\star = \underset{j=1,\cdots,k}{\operatorname{argmax}} \hat{\mathbf{G}}_{i,j} \tag{6.15}$$

6.4.2 Automatic Overlapping Clustering

The formulation in Eq. (6.15) can only deal with a single hard cluster membership problems, i.e., where each node is assigned to a single cluster. However in BIGs, nodes often overlap in multiple SACs with different roles. In this section, I describe how to perform overlapping clustering.

Overlapping Clustering. The strategy in most current clustering algorithms is to assign a node to the cluster with the highest probability of membership. Technically, the initial clustering results could be derived using Eq.(6.6) and Eq.(6.15), which could then be used to fine-tune the overlapping clustering results. However, to make this process automatic, our strategy relies on a threshold for each cluster where each node is assigned to a cluster if its memebership probability exceeds that threshold. Specifically, I determine a threshold γ_j for each cluster C_j, and instead of assigning node v_i to a single cluster $C_{j^\star}$ via Eq. (6.15), I assign v_i to any cluster C_j as long as $\hat{\mathbf{G}}_{i,j} \geq \gamma_j$.

Suppose the initialized groups obtained from Eq.(6.15) are $\mathcal{C} = \{C_1, C_2, \cdots, C_k\}$, where $C_j = \bigcup\{v_i\}$ is the set of nodes in each cluster. For each node $v_i \in C_j$, the membership value of $\hat{\mathbf{G}}$ is $\hat{\mathbf{G}}_{i,j}$. C_j will have the set of membership values $P_j = \bigcup_{v_i \in C_j}\{\hat{\mathbf{G}}_{i,j}\}$. Next, I sort it to find the minimum value of P_j

$$\gamma_j = \min P_j \tag{6.16}$$

γ_j is the minimum threshold for assgining a node to cluster C_j. Thus the final clustering result of the j-th cluster is:

$$f(v_i) = C_j \quad : \quad \text{if} \quad \hat{\mathbf{G}}_{i,j} \geq \gamma_j, j = 1 \cdots k \tag{6.17}$$

CBIG automatically performs the overlapping clustering using Eq.(6.17).

Algorithm 3 describes the algorithm for SAC discovery, which requires the graph structure matrix $\mathbf{W}_s$, the user-feature matrix $\mathbf{X}$, the number of feature clusters q,

and the number of node clusters k as inputs. The algorithm initializes $n \times k$ matrix $\mathbf{G}$ and $m \times q$ matrix $\mathbf{F}$ using a k-means algorithm in Steps 2-3. In Steps 4-10, $\mathbf{G}_s$,$\mathbf{G}$, $\mathbf{F}_s$, $\mathbf{F}$ and $\mathbf{S}$ are iteratively updated until there are no more changes to the matrixes or it reaches the maximum number of iteration S_{max} (In our experiments S_{max} was 80). The clustering results for the nodes from matrix $\hat{\mathbf{G}}$ and the clustering results for the features from matrix $\hat{\mathbf{F}}$ are determined in Step 11. In Steps 14-16, CBIG fine tunes the clustering results for overlapping clustering by first finding the threshold γ_j for each cluster group C_j in Steps 15-16, and then assigns the nodes with membership values not less than γ_j to cluster C_j.

Determining Optimal Amount of Clusters. II propose a hold-out strategy for extending CBIG to automatically determine the optimal number of clusters k instead of manually setting them. Specifically, I hold out a subset of $\mathbf{X}$ and $\mathbf{W}_s$ as the test set and the rest are used for training. Then I conduct factorization on the training subset of $\mathbf{X}$ and $\mathbf{W}_s$ and predict the reconstruction errors on the test entries, which leads to the revised objective function:

$$\begin{aligned} J_6 =& ||\mathbf{Y} \odot \mathbf{X} - \mathbf{F}\mathbf{S}\mathbf{G}^\top||_F^2 + \alpha||\mathbf{Y} \odot \mathbf{W}_s - \mathbf{G}_s\mathbf{G}_s^\top||_F^2 \\ &+ \beta||\mathbf{W}_f - \mathbf{F}_f\mathbf{F}_f^\top||_F^2 \\ &+ \rho(||\mathbf{G} - \mathbf{G}_s||_F^2 + ||\mathbf{F} - \mathbf{F}_f||_F^2), \\ &s.t.\ \mathbf{F} \geq 0, \mathbf{G} \geq 0, \mathbf{F}_f \geq 0, \mathbf{G}_s \geq 0, \end{aligned} \tag{6.18}$$

where $\mathbf{Y} \in \mathbb{R}_+^{n \times n}$ is binary. If the entry $\mathbf{Y}_{i,j} = 1$, it is for training, otherwise it is for testing.

Then the reconstruction error on the test entries ($\mathbf{Y}_{i,j} = 0$) is computed as follows:

$$ReconsErr = ||\neg\mathbf{Y} \odot \mathbf{X} - \mathbf{F}\mathbf{S}\mathbf{G}^\top||_F^2 + ||\neg\mathbf{Y} \odot \mathbf{W}_s - \mathbf{G}_s\mathbf{G}_s^\top||_F^2 \tag{6.19}$$

where $\mathbf{Y}$ is the negation of $\neg\mathbf{Y}$. Specifically, $\mathbf{Y}_{ij} = 0$ if $\neg\mathbf{Y}_{ij}$=1, otherwise it is 1. In the experiments, k is varied from k_{min} to k_{max} until the reconstruction error

ReconsErr does not decrease. By doing so, the number of k clusters is empirically determined.

Time Complexity. Here I briefly analyze the time complexity of Algorithm 3. Initializing the k-means algorithm in Step 1 as linear time complexity, i.e., $O(nmkt)$ [111], where n is the number of m-dimensional vectors, k the number of clusters and t the number of iterations needed until convergence. Steps 5 to 9 employ multiplicative update rules, the time complexity for each step is $O(nmr)$ [112, 113], where $r = \max(k, q)$. Thus the time complexity for the matrix factorization is $\#\text{interation} * O(nmr)$. In Step 21, sorting costs $O(n \log n)$. Thus, the total time complexity is

$$O(nmkt) + \#\text{interation} * O(nmr) + O(n \log n).$$

6.5 Experiments

I assembled real-world business social graphs to evaluate CBIG. The aim was to show that (1) CBIG offers outstanding performance when clustering users(SAC discovery);(2) CBIG clusters features and provides a meaningful understanding of the topics in graphs; and (3) the correlations between the feature clusters and the node clusters provides insights into the role of each SAC in the BIG.

6.5.1 Setup of the Experiments

Business Information Graph Datasets

To evaluate CBIG and conduct the experiments, I assembled 13 real-world datasets from Twitter. These datasets cover various industries, including sports associations, motor enterprises, political parties, and news agencies. Each enterprise has an official Twitter homepage, and their online followers are manually managed with *Twitter lists*. Each of the followers in one list share some common interests

Algorithm 3 CBIG: Clustering Business Information Graphs

Require: user-feature matrix $\mathbf{X} = [\mathbf{x}_{\cdot 1}, \cdots, \mathbf{x}_{\cdots n}] \in \mathbb{R}_+^{d \times n}$,

Graph structure matrix $\mathbf{W}_s \in \mathbb{R}_+^{n \times n}$;

k: number of node clusters;

q: number of feature clusters.

Constructed $\mathbf{W}_f \in \mathbb{R}_+^{m \times m}$, i.e., $[\mathbf{W}_f]_{ij} =< \mathbf{x}_{i\cdot}, \mathbf{x}_{j\cdot} >$

Initialize $\mathbf{G} \in \mathbb{R}_+^{n \times k}$ and $\mathbf{F} \in \mathbb{R}_+^{m \times q}$ using k-means on $\mathbf{X}$ and $\mathbf{X}^\top$, respectively;

Initialize $\mathbf{G}_s = \mathbf{G}$ and $\mathbf{F}_f = \mathbf{F}$;

repeat

$\mathbf{G} \leftarrow \mathbf{G} \odot \frac{\mathbf{X}^\top \mathbf{F}\mathbf{S} + \rho \mathbf{G}_s}{\mathbf{G}\mathbf{S}^\top \mathbf{F}^\top \mathbf{F}\mathbf{S} + \rho \mathbf{G}}$;

$\mathbf{F} \leftarrow \mathbf{F} \odot \frac{\mathbf{X}\mathbf{G}\mathbf{S}^\top + \rho \mathbf{F}_f}{\mathbf{F}\mathbf{S}\mathbf{G}^\top \mathbf{G}\mathbf{S}^\top + \rho \mathbf{F}}$;

$\mathbf{G}_s \leftarrow \mathbf{G}_s \odot \frac{\rho \mathbf{G} + 2\alpha \mathbf{W}_s^\top \mathbf{G}_s}{2\alpha \mathbf{G}_s \mathbf{G}_s^\top \mathbf{G}_s + \rho \mathbf{G}_s}$;

$\mathbf{F}_f \leftarrow \mathbf{F}_f \odot \frac{\rho \mathbf{F} + 2\beta \mathbf{W}_s^\top \mathbf{F}_f}{2\beta \mathbf{F}_f \mathbf{F}_f^\top \mathbf{F}_f + \rho \mathbf{F}_f}$;

$\mathbf{S} \leftarrow \mathbf{S} \odot \frac{\mathbf{F}^\top \mathbf{X}\mathbf{G}}{\mathbf{F}^\top \mathbf{F}\mathbf{S}\mathbf{G}^\top \mathbf{G}}$;

until Converges;

// **Final clustering results for audiences and features**

$\hat{\mathbf{G}} = \mathbf{G} + \mathbf{G}_s; \quad \hat{\mathbf{F}} = \mathbf{F} + \mathbf{F}_f$;

// **Overlapping Clustering**

Get the initial clustering result $\{C_1, C_2, \cdots, C_k\}$, where $C_j = \bigcup\{v_i\}$ according to Eq. (6.15);

Get the threshold γ_j for cluster C_j according to Eq.(6.16);

Overlapping Clustering according to Eq. (6.17).

or have a similar relationship to the enterprise. Each enterprise hires dedicated employees to manually categorize the audiences into the different lists. Figure 6.1 shows an example of an intelligence workflow using a Twitter list.

As Twitter users have limited profile information, I shaped each follower's interests by collecting and processing their most recent 100 tweets (retrived in March, 2015). Specifically, I took the top 1000 of the most-frequent words or phases marked with a hashtag # (to indicate an event) or the sign @(to mention a user) to denote all the interests in one dataset. Table 1 contains the details of the datasets: the screen name of each organization, the number of followers, the number of labeled followers, the number of edges(linkages) between followers, the number of tweets by follower, the number of lists (clusters), and a brief description of each enterprise. To evaluate CBIG's performance, I regarded the *Twitter lists* from each enterprise as the ground truth user labels. If two users belonged to the same list, I considered them as one cluster.

Table 1: Business social graph datasets used in this article

Companies	Total Nodes	Labeled	Edges	Tweets	List	Description
ABCNews	845,196	729	43,032	69,018	17	News division of Australian Broadcasting Corpo.
NBA	17,387,126	137	2,036	12,402	7	National Basketball Association
TwitterAU	210,194	531	13,718	45,324	13	Twitter official account in Australia
SonyPictures	1,454,582	129	186	10,801	5	Sony Pictures Entertainment Corporation
AustralianLabor	111,447	346	11,484	9,646	5	Labour Party in Australia
WhiteHouse	7,377,128	161	3,925	13,503	5	White House
MercedesBenz	1,434,283	142	1,174	11,846	7	Mercedes-Benz Automobile Manufacturer Corpo
Techreview	452,766	155	1,220	14,024	7	MIT Technology Review
Cambridge_Uni	214,640	529	4,125	47,329	12	Cambridge University
The_Nationals	22,253	27	173	1,941	3	The National Party in Australia
Greens	68,607	154	2,384	13,078	9	The Greens Party in Australia
MGM_Studios	835,751	68	621	6,414	6	Disney Metro Goldwyn Mayer Hollywood Studio
BBCNews	5,019,860	624	16,974	61,533	11	British Broadcasting Corporation

Baselines

I compared CBIG to nine relevant algorithms to validate its performance. The comparison algorithms included approaches that only leverage either textual infor-

mation or topological information, as well as approaches that consider both.

The algorithms that only consider either textual or topological information are listed below:

- **BigClam [88]** combines NMF with the block stochastic gradient descent to detect communities from topological information.
- **AgmFit [87]** is a community detection algorithm based on community-affiliation graph model.
- **K-means** is a classical and effective unsupervised approach. I only used k-means on textual information for this experiment.

The baseline algorithms that consider both textual and topological information, include relational topic models [89], NMF based algorithms [37, 91, 92], and the state-of-the-art algorithms in [68, 105]:

- **Censa [105]** is a statistic-based model that forms the interactions between the node content and the graph structure to elicit a more accurate community detection.
- **Circle [68]** is an attributed graph clustering algorithm that represents overlapping hard-membership approaches for graph clustering.
- **Block-LDA [89]** is an LDA-based relational topic model method that considers both textual and topological information for clustering tasks.
- **FNMTF [92]** is an NMF-based co-clustering method that targets large scale datasets.
- **DRCC [91]** extracts nodes and features from the information graph to construct a user graph and a feature graph for clustering.

- **GNMF [37]** adds the encoded k-NN graph as a regularization term to the objective function of the classical NMF method.

To use NMF based methods [37, 91, 92], I needed to replace the manifold graphs (k-NN graphs) with graph structures for regularization. However, in practice, the graph structures may be not the same as the k-NN graphs. It is worth noting that although NMF based methods and relational topic models [89] can cluster nodes and features simultaneously, they are not designed for overlapping clustering. However, Circle [68] and Censa [105] can perform overlapping clustering, but cannot provide any interpretation of the clustering results at the feature level. By comparison, CBIG not only simultaneously clusters nodes and features in a business social graph, but it also enables overlapping clustering for SAC discovery.

Table 2 compares all ten algorithms.

Table 2: Comparison of CBIG with other algorithms

	Ours	K-Means	BigClam	AgmFit	Censa	FNMTF	GNMF	DRCC	Nips	BLOCK-LDA
Content	*	*			*	*	*	*	*	*
Structure	*		*	*	*	*	*	*	*	*
Overlapping	*		*	*	*					
Explain Clusters	*									
Auto-determine Clusters number	*		*	*	*					

Evaluation Metrics

Although CBIG is unsupervised, I evaluated its performance by comparing the predicted clusters $\mathcal{C} = \{C_1, \cdots, C_k\}$ with the manually labeled ground truth $\overline{\mathcal{C}} = \{\overline{C_1}, \cdots, \overline{C_k}\}$. Ideally, the predicted clusters should be consistently aligned with the ground truth.

I applied a balanced error rate (BER)[68, 114] to calculate the error between the

predicted clusters $\mathcal{C}$ and the ground truth $\overline{\mathcal{C}}$. The calculation is defined as:

$$\text{BER}(C, \overline{C}) = \frac{1}{2}(\frac{|C/\overline{C}|}{|C|} + \frac{|C^c/\overline{C^c}|}{|C^c|}) \tag{6.20}$$

BER assigns equal importance to false positives and false negatives, so that trivial or random predictions incur an error rate of 0.5 on average. Such a measure is preferable to, say, a $0/1$ loss, which assigns an extremely low error rate to trivial predictions.

I also used F_1 scores as an evaluation metric (F-measure)[115]. F_1 scores consider both the precision and recall of the clustering result. Only results with a high precision and recall rate produce good F_1 scores.

Parameter Study

For a fair comparison, I conducted experiments on each baseline by varying the parameter settings according to [91, 37, 92], and used the best as the final performance for each baseline. For CBIG, I varied α, β, and σ from 0.2 to 2 in increments of 0.2. For DRCC [91], GNMF [37], and FNMTF [92], I set $\lambda = \mu$ and λ was tuned by searching the grids $\{0.1, 1, 10, 100, 500, 1000\}$ as described in their paper. For AgmFit, I set the parameter e (edge probability between the nodes that do not share any community) by searching the grids$\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$ and chose the best value as the final result. For the Circle method, I set the regularization parameter $\lambda_\epsilon \in \{0, 1, 10, 100\}$ as described in their paper.

In each experiments, CBIG automatically determined the number of clusters k and I compared the resulting performance to the performance of the existing overlapping algorithms: BigClam [88], AgmFit [87], and Censa [105].

I also fixed the value of k with ranges of 3, 5, 7, 9, 11, to compare a broad range of settings on clustering tasks. I determined each setting for each dataset with every algorithm 30 times and used the average value as the reported score.

Unless otherwise specified, I set k_{min}=2, k_{max} = 10 and Z=30 for CBIG to automatically determine the value of k.

6.5.2 Results of the Experiments

This section presents the results of the experiments on a user clustering task. I assumed the number of clusters k was unknown and compared CBIG with the overlapping clustering algorithms including BigClam [88], AgmFit [87], and Censa [105]. I then specified the value of k, and compared the performance of all algorithms.

Overlapping Clustering

This section compares the results for CBIG with the overlapping clustering methods: BigClam [88], AgmFit [87], and Censa [105], all of which are available on the SNAP platform [†]. However, because the Circle [68] source code does not provide an automatic clustering functionality, I have not included its result in this section, and instead only report the results for a given desired k in next subsection.

Looking at Tables 3 and 4, I can see that CBIG dramatically outperformed the other algorithms. For example, CBIG achieved F1 scores of 0.988 and 0.925 on the NBA and WhiteHouse datasets, while, Censa only achieved F1 scores of 0.791 and 0.295.

Overall, CBIG outperformed the algorithms in 12 of the 13 real-world datasets according to BER and surpassed all the baselines in terms of the F1 scores. One reason could be that algorithms like BigClam and AgmFit only leverage topological structures when clustering users, ignoring any contextual information that might provide complementary knowledge. As a consequence, their results were suboptimal. Alternatively, algorithms like Censa, which consider both topological information

[†]http://snap.stanford.edu/

Table 3: BER scores of methods that auto detect k

Auto Ber_loss	Ours	BigClam	AgmFit	Censa
ABCNews	0.19	0.303	0.484	0.413
NBA	0.007	0.125	0.368	0.105
TwitterAU	0.045	0.313	0.467	0.249
SonyPictures	0.357	0.376	0.461	0.414
AustralianLabor	0.132	0.229	0.332	0.23
WhiteHouse	0.037	0.281	0.447	0.315
MercedesBenz	0.046	0.227	0.403	0.295
Techreview	0.317	0.297	0.435	0.342
Cambridge_Uni	0.091	0.264	0.421	0.242
The_Nationals	0.102	0.133	0.212	0.228
Greens	0.222	0.295	0.44	0.327
MGM_Studios	0.126	0.269	0.261	0.263
BBCNews	0.116	0.233	0.138	0.138

Table 4: F1 scores of methods that auto detect k

Auto F1 Score	Ours	BigClam	AgmFit	Censa
ABCNews	0.515	0.317	0.038	0.171
NBA	0.988	0.729	0.27	0.791
TwitterAU	0.829	0.376	0.072	0.44
SonyPictures	0.419	0.257	0.085	0.167
AustralianLabor	0.765	0.487	0.345	0.478
WhiteHouse	0.925	0.244	0.115	0.295
MercedesBenz	0.85	0.528	0.202	0.405
Techreview	0.445	0.371	0.138	0.255
Cambridge_Uni	0.692	0.453	0.159	0.521
The_Nationals	0.869	0.549	0.593	0.403
Greens	0.589	0.45	0.137	0.358
MGM_Studios	0.715	0.456	0.485	0.427
BBCNews	0.752	0.447	0.179	0.712

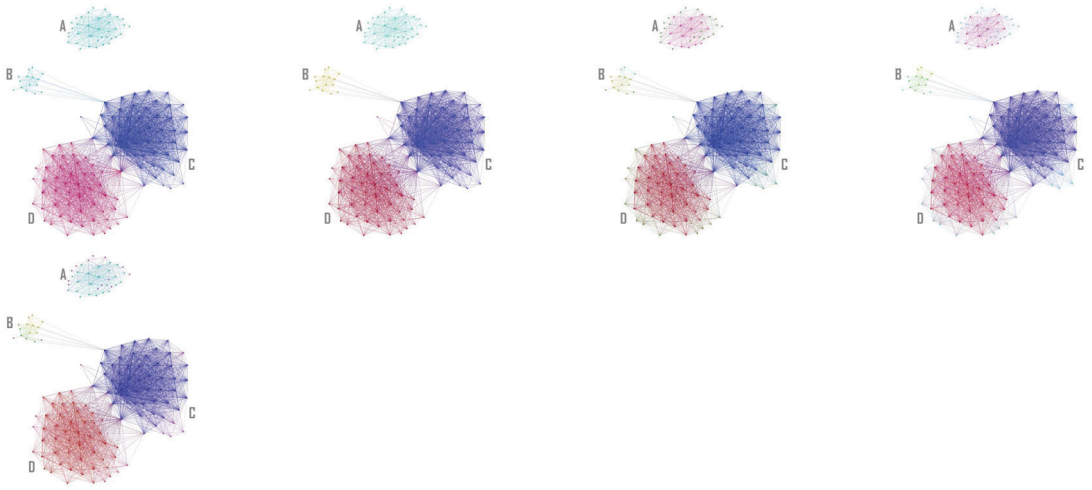

Figure 6.4 : Comparison results of the White House dataset. From left to right, the graph comes from our algorithm, the ground truth, BigClam, Censa and AMG-Fit. Each color represents a cluster. The purer the color of a cluster, the better performance.

and textual information, are based on the assumption that the communities "generate" both the topological structure and its textual features. Specifically, Censa assumes that the graph links have a high consistency or high dependency on the users' attributes, which is not always true in BIGs. As preciously discussed the graph links and user attributes in BIGs are highly inconsistent. As a result, the performance of Censa was only comparable to BigClam.

Case Study

Fig 6.4 illustrates the clustering results, showing that the clusters predicted by CBIG were almost perfectly aligned with the ground truth. However, the results from the other baselines contained noticeable errors. For example, Bigclam misclassified a large group of users in Cluster A.

Table 5: BER score with K = 5

K = 5 Ber_loss	Ours	K-Means	BigClam	AgmFit	Censa	FNMTF	GNMF	DRCC	Nips	BLOCK-LDA
ABCNews	0.23	0.324	0.409	0.468	0.411	0.343	0.431	0.324	0.188	0.343
NBA	0.027	0.192	0.153	0.186	0.138	0.291	0.302	0.175	0.215	0.185
TwitterAU	0.039	0.288	0.402	0.436	0.401	0.213	0.359	0.137	0.185	0.418
SonyPictures	0. 290	0.343	0.344	0.449	0.338	0.413	0.469	0.307	0.415	0.113
AustralianLabor	0.126	0.348	0.27	0.197	0.279	0.409	0.364	0.36	0.214	0.079
WhiteHouse	0.037	0.381	0.36	0.268	0.363	0.392	0.462	0.413	0.21	0.125
MercedesBenz	0.115	0.407	0.245	0.281	0.259	0.398	0.352	0.343	0.234	0.193
Techreview	0.271	0.436	0.116	0.416	0.122	0.429	0.454	0.44	0.335	0.263
Cambridge_Uni	0.092	0.426	0.355	0.403	0.358	0.351	0.472	0.348	0.206	0.317
The_Nationals	0.036	0.419	0.094	0.176	0.119	0.241	0.22	0.322	0.247	0.308
Greens	0.333	0.343	0.261	0.287	0.256	0.365	0.364	0.333	0.183	0.301
MGM_Studios	0.083	0.349	0.239	0.203	0.202	0.394	0.468	0.204	0.187	0.265
BBCNews	0.113	0.394	0.221	0.371	0.223	0.398	0.458	0.386	0.187	0.303
Average	0.156	0.358	0.267	0.318	0.267	0.357	0.398	0.321	0.231	0.247

Table 6: F1 score with K = 5

K = 5 F1 Score	Ours	K-Means	BigClam	AgmFit	Censa	FNMTF	GNMF	DRCC	Nips	BLOCK-LDA
ABCNews	0.432	0.215	0.181	0.075	0.177	0.294	0.216	0.222	0.46	0.686
NBA	0.922	0.641	0.698	0.633	0.728	0.467	0.393	0.61	0.458	0.368
TwitterAU	0.834	0.444	0.203	0.133	0.207	0.44	0.28	0.687	0.484	0.833
SonyPictures	0. 527	0.42	0.33	0.113	0.337	0.338	0.192	0.501	0.348	0.224
AustralianLabor	0.783	0.366	0.457	0.603	0.442	0.302	0.341	0.397	0.632	0.157
WhiteHouse	0.924	0.297	0.285	0.459	0.279	0.311	0.217	0.265	0.504	0.248
MercedesBenz	0.757	0.238	0.516	0.42	0.486	0.291	0.363	0.36	0.482	0.383
Techreview	0.462	0.224	0.757	0.176	0.746	0.248	0.242	0.218	0.367	0.522
Cambridge_Uni	0.713	0.229	0.295	0.197	0.289	0.244	0.155	0.237	0.366	0.633
The_Nationals	0.962	0.356	0.737	0.622	0.716	0.663	0.658	0.528	0.676	0.593
Greens	0.379	0.346	0.51	0.447	0.523	0.318	0.319	0.34	0.517	0.597
MGM_Studios	0.776	0.458	0.515	0.552	0.583	0.297	0.189	0.578	0.462	0.522
BBCNews	0.748	0.291	0.577	0.252	0.559	0.303	0.16	0.285	0.51	0.605
Average	0.681	0.348	0.466	0.396	0.467	0.347	0.286	0.402	0.482	0.49

Table 7: BER score with K = 9

K = 9 BER Loss	Ours	K-Means	BigClam	AgmFit	Censa	FNMTF	GNMF	DRCC	Nips	BLOCK-LDA
ABCNews	0.274	0.374	0.353	0.37	0.344	0.345	0.411	0.359	0.223	0.365
NBA	0.043	0.322	0.129	0.16	0.144	0.342	0.352	0.287	0.231	0.248
TwitterAU	0.142	0.268	0.276	0.328	0.277	0.261	0.371	0.201	0.225	0.271
SonyPictures	0. 339	0.473	0.387	0.442	0.391	0.411	0.36	0.366	0.357	0.259
AustralianLabor	0.118	0.36	0.256	0.291	0.227	0.375	0.32	0.336	0.234	0.3455
WhiteHouse	0.089	0.486	0.266	0.281	0.302	0.397	0.458	0.383	0.227	0.334
MercedesBenz	0.148	0.481	0.234	0.281	0.28	0.418	0.392	0.355	0.274	0.181
Techreview	0.313	0.475	0.3	0.306	0.319	0.421	0.468	0.405	0.304	0.21
Cambridge_Uni	0.054	0.447	0.16	0.371	0.131	0.397	0.481	0.384	0.231	0.235
The_Nationals	0.272	0.377	0.202	0.237	0.227	0.259	0.271	0.279	0.197	0.308
Greens	0.34	0.428	0.295	0.328	0.301	0.402	0.441	0.409	0.249	0.337
MGM_Studios	0.135	0.398	0.239	0.324	0.225	0.394	0.27	0.219	0.237	0.242
BBCNews	0.179	0.418	0.118	0.262	0.13	0.427	0.475	0.402	0.239	0.213
Average	0.19	0.408	0.247	0.306	0.254	0.373	0.39	0.337	0.248	0.273

Table 8: F1 score with K = 9

K = 9 F1 Score	Ours	K-Means	BigClam	AgmFit	Censa	FNMTF	GNMF	DRCC	Nips	BLOCK-LDA
ABCNews	0.366	0.192	0.257	0.226	0.272	0.174	0.231	0.159	0.319	0.729
NBA	0.941	0.437	0.709	0.652	0.689	0.419	0.399	0.501	0.432	0.493
TwitterAU	0.625	0.434	0.455	0.35	0.453	0.415	0.285	0.529	0.329	0.54
SonyPictures	0.451	0.172	0.24	0.116	0.224	0.308	0.441	0.414	0.426	0.513
AustralianLabor	0.798	0.405	0.433	0.378	0.49	0.325	0.462	0.445	0.574	0.689
WhiteHouse	0.825	0.122	0.357	0.356	0.32	0.297	0.207	0.321	0.494	0.665
MercedesBenz	0.714	0.115	0.521	0.41	0.43	0.249	0.263	0.375	0.396	0.359
Techreview	0.384	0.161	0.348	0.321	0.303	0.261	0.193	0.253	0.371	0.418
Cambridge_Uni	0.81	0.187	0.684	0.261	0.741	0.205	0.097	0.274	0.315	0.469
The_Nationals	0.549	0.385	0.586	0.468	0.404	0.624	0.566	0.578	0.734	0.593
Greens	0.375	0.165	0.295	0.379	0.414	0.256	0.177	0.211	0.351	0.669
MGM_Studios	0.723	0.275	0.239	0.258	0.534	0.27	0.473	0.671	0.389	0.478
BBCNews	0.678	0.244	0.76	0.459	0.732	0.237	0.093	0.279	0.406	0.426
Average	0.634	0.254	0.453	0.402	0.462	0.311	0.299	0.385	0.426	0.542

Results of the Experiments on Node Clustering with User-specified k.

To compare CBIG with all baselines, including both the overlapping and non-overlapping methods, I specified the number of clusters (k) in a user clustering task. The results for K=5, 9, and 11 are shown in Tables 5 - 10.

Table 9: BER score with K = 11

K = 11 BER Loss	Ours	K-Means	BigClam	AgmFit	Censa	FNMTF	GNMF	DRCC	Nips	BLOCK-LDA
ABCNews	0.307	0.387	0.323	0.316	0.335	0.386	0.424	0.386	0.238	0.197
NBA	0.036	0.296	0.129	0.142	0.148	0.349	0.329	0.241	0.241	0.27
TwitterAU	0.063	0.229	0.299	0.202	0.324	0.289	0.349	0.26	0.249	0.222
SonyPictures	0.338	0.469	0.387	0.442	0.389	0.401	0.369	0.368	0.362	0.245
AustralianLabor	0.106	0.357	0.238	0.261	0.23	0.398	0.354	0.389	0.272	0.385
WhiteHouse	0.092	0.479	0.276	0.228	0.315	0.393	0.407	0.399	0.212	0.328
MercedesBenz	0.118	0.483	0.231	0.288	0.295	0.428	0.389	0.378	0.276	0.193
Techreview	0.298	0.464	0.342	0.352	0.342	0.416	0.469	0.421	0.333	0.233
Cambridge_Uni	0.159	0.444	0.238	0.343	0.238	0.416	0.483	0.397	0.272	0.165
The_Nationals	0.113	0.356	0.122	0.108	0.228	0.264	0.259	0.28	0.222	0.212
Greens	0.314	0.403	0.314	0.343	0.327	0.418	0.433	0.401	0.258	0.36
MGM_Studios	0.157	0.39	0.218	0.324	0.263	0.345	0.272	0.251	0.24	0.311
BBCNews	0.244	0.432	0.162	0.25	0.138	0.433	0.479	0.402	0.262	0.191
Average	0.18	0.399	0.252	0.277	0.275	0.38	0.386	0.352	0.264	0.255

Table 10: F1 score with K = 11

K = 11 F1 Score	Ours	K-Means	BigClam	AgmFit	Censa	FNMTF	GNMF	DRCC	Nips	BLOCK-LDA
ABCNews	0.338	0.132	0.281	0.294	0.318	0.168	0.198	0.164	0.271	0.393
NBA	0.95	0.461	0.709	0.698	0.633	0.393	0.433	0.565	0.418	0.5368
TwitterAU	0.808	0.539	0.406	0.601	0.359	0.363	0.275	0.44	0.278	0.443
SonyPictures	0.439	0.21	0.24	0.119	0.233	0.332	0.42	0.394	0.43	0.487
AustralianLabor	0.86	0.395	0.458	0.396	0.478	0.306	0.408	0.346	0.494	0.767
WhiteHouse	0.863	0.149	0.31	0.43	0.295	0.291	0.313	0.297	0.502	0.652
MercedesBenz	0.787	0.103	0.52	0.367	0.405	0.233	0.271	0.318	0.383	0.383
Techreview	0.442	0.201	0.267	0.221	0.255	0.247	0.176	0.248	0.342	0.463
Cambridge_Uni	0.652	0.167	0.527	0.316	0.527	0.174	0.087	0.256	0.261	0.329
The_Nationals	0.85	0.403	0.619	0.661	0.403	0.624	0.624	0.558	0.697	0.407
Greens	0.413	0.249	0.389	0.335	0.358	0.222	0.169	0.246	0.341	0.714
MGM_Studios	0.704	0.257	0.515	0.298	0.427	0.318	0.463	0.463	0.413	0.612
BBCNews	0.554	0.205	0.658	0.495	0.712	0.223	0.079	0.28	0.363	0.381
Average	0.666	0.267	0.454	0.402	0.416	0.299	0.301	0.352	0.399	0.505

These results show that CBIG outperformed the other baselines in most cases. Algorithms such as AgmFit [87], BigClam [88], and k-means only consider textual

information about users or the topological information of the graph, ignoring the other sources of information. As a result, these algorithms cannot fully leverage different sources of information available in the BIG and, therefor, produced sub-optimal performance.

By contrast, our algorithm maximizes the degree of freedom to explore the distinct information encoded in the graph data which leads to an increase in performance.

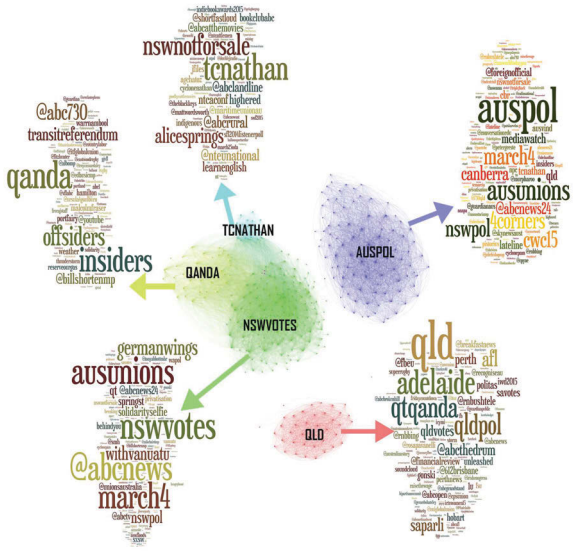

Figure 6.5 : Our algorithm detected five groups from AustralianLabor data set and clustered corresponding features groups simultaneously.

Interestingly, the algorithms that use both topological information and textual information, like Censa [105], Block-LDA [89], Circle [68], FNMTF [92], DRCC [91], and GNMF [37] did not outperform algorithms like BigClam and k-means, which only leverage one source of information. These experiments confirm that there is a risk of confusing the model when integrating multiple sources of information. The GNMF, DRCC, and FNMTF methods are all NMF based algorithms regularized by a manifold graph (k-NN graph); however real-world graph structures usually

follow a power law distribution which is different from a k-NN graph. In addition, in a k-NN graph, a node is supposed to be connected to its k nearest neighbors, which is calculated using feature values, so the links are highly consistent with the node contents. This is, unfortunately, not the case for BIG where topological information is typically highly inconsistent with the textual information. As a result, the algorithms that draw on multiple sources of information did not perform as well. By comparison, our algorithm extracts the best ingredients from both the topological structure and the textual information, through the proposed consensus factorization framework. Hence, the performance is outstanding compared to the other nine algorithms.

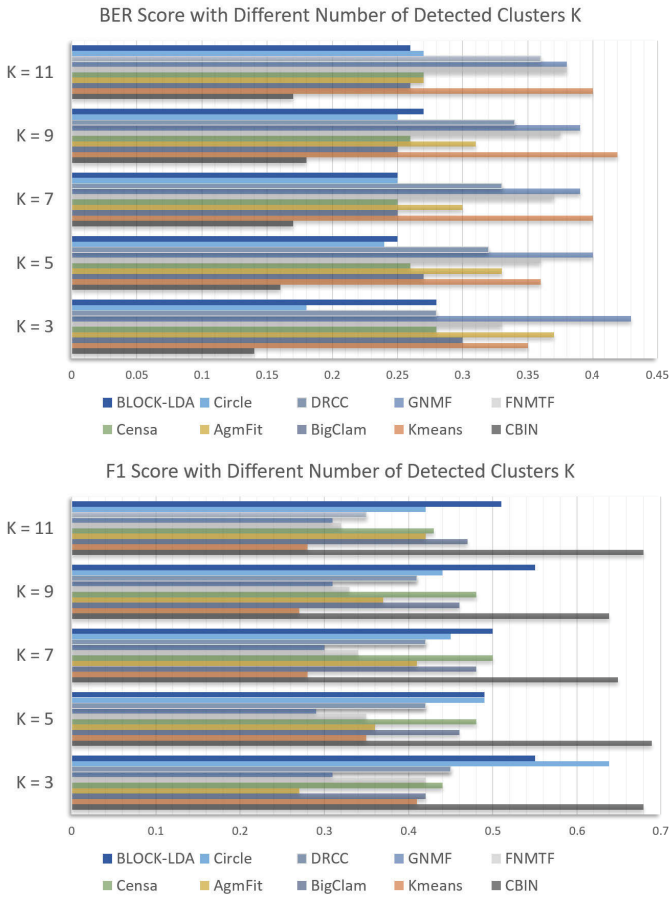

Figure 6.6 : Average performance on different number of detected clusters k. (A) BER score (smaller is better), (B) F_1 score (larger is better).

The average performance of each algorithm is illustrated in Fig 6.6, which shows that our algorithm had low BER scores and higher F_1 scores for different K values.

Figure 6.7 : Feature word clouds on the *AustralianLabor* dataset. Each word cloud represents a word cluster. The larger a word in a cloud, the more frequent it is discussed online.

Table 11: Keywords and explanation in word clouds

TCNathan	A tropical cyclone which lashed North Queensland
qanda	Question and Answer, an ABC politic TV show
Auspol	Australian Policy/Politic
qld	Queensland, the second largest state in Australia
ausunions	Australian Unions

6.5.3 Word Cloud: Understanding the Feature Groups

One noticeable property of CBIG is its capacity to group node features into clusters with a better understanding of the SACs relevance to an organization's business model.

The clustering results shown in In Fig. 6.7 demonstrate that each word cloud is indeed very meaningful in practice ,while Table 11 shows the meaning of the most frequent word appearing in each word cloud from Fig. 6.7.

The results in Table 11 indicate that the feature clusters discovered by CBIG indeed provide an alternative understanding of the users' interests. For instance, the keyword "qld"(meaning "Queensland", the second-largest and third-most-populated state in Australia) is larger than other words. This means that *Queensland* is frequently discussed on Twitter because the members of the group mostly come from Queensland or care about things that happen in Queensland.

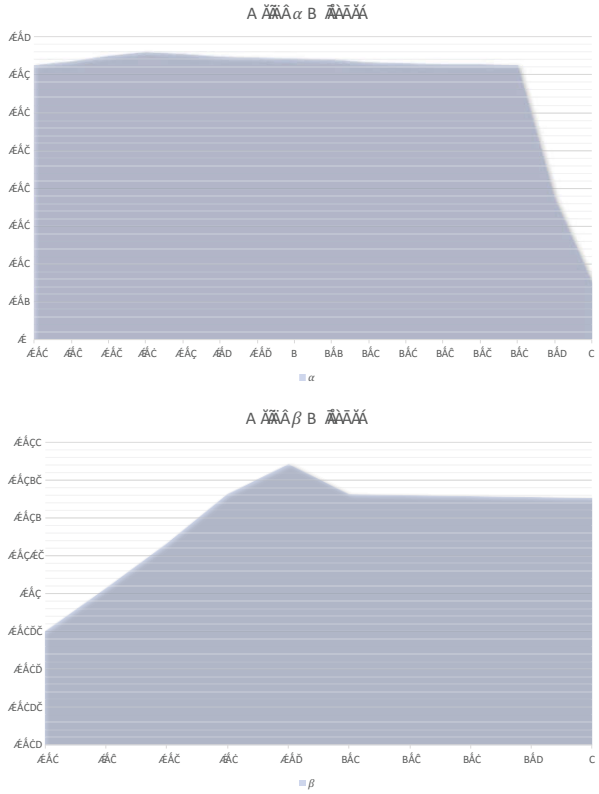

Figure 6.8 : Clustering performance *w.r.t.* different α, β values

6.5.4 Connections between Feature Cluster and Node Cluster

Another attractive property of CBIG algorithm is that it can reveal the connections between the node clusters and feature clusters, though a latent mapping matrix **S**.

$\mathbf{S}_{ij}$ unveils the corresponding weight between feature cluster C_i and node cluster C_j. Take *AustralianLabor* as an example, whose $\mathbf{S}$ matrix is described in Table 6.1. Based on the best matching principle, CBIG clusters nodes of AustralianLabor into functional user groups 1 to groups 5 corresponding with feature clusters "tc-nathan","qanda", "auspol", "qld" and "ausunions" respectively. From Table 6.1, it is easy to find that node cluster 1 is most related to feature cluster 2, "qanda", ($\mathbf{S}_{12}$), and node cluster 2 is closely connected to feature cluster 5, "ausunions", ($\mathbf{S}_{25}$).

Based on $\mathbf{S}$ matrix, we can better understand and interpret the interest of each function user group. The functional group interest is shown in Fig 6.5.

$\mathbf{S}_{ij}$ unveils the corresponding weight between feature cluster C_i and node cluster C_j. Take *AustralianLabor* as an example, whose $\mathbf{S}$ matrix is described in Table 6.1. Based on the best matching principle, CBIG clusters nodes of AustralianLabor into functional user groups 1 to groups 5 corresponding with feature clusters "tc-nathan","qanda", "auspol", "qld" and "ausunions" respectively. From Table 6.1, it is easy to find that node cluster 1 is most related to feature cluster 2, "qanda", ($\mathbf{S}_{12}$), and node cluster 2 is closely connected to feature cluster 5, "ausunions", ($\mathbf{S}_{25}$).

Based on $\mathbf{S}$ matrix, we can better understand and interpret the interest of each function user group. The functional group interest is shown in Fig 6.5.

6.5.5 Parameter Sensitivity

I varied α, β and σ from 0.3 to 2.0, to validate the performance of CBIG in terms of F_1 score, and the results are shown in Fig. 6. I have omitted a diagram for σ due to space limitation. As the value of α increased from 0.3 to 1.5, each dataset generated very similar results in terms of both BER and F_1 scores. However, performance plummeted when α was further increased. The results of the F_1 and BER scores in terms of β were similar to the results observed when changing the values for α.

Table 6.1 : Matching-Matrix (**S**) between node and feature clusters

1.586e-3	**1.181e-2**	1.320e-4	6.137e-5	1.393e-3
3.538e-2	3.818e-1	7.843e-4	3.172e-1	**1.520**
1.379e-1	5.081e-3	2.594e-1	**1.931**	1.296
4.301	2.291e-3	4.968e-4	4.129e-2	3.618e-1
2.377e-3	9.925e-1	**2.808**	1.254e-1	1.583e-2

6.6 Summary

I argue that a BIG, driven by relationships, policies, and business interests, is significant in enabling business intelligence and decision making for companies. To cluster enterprise audiences in BIGs, CBIG discovers SACs and further insights into the relevance of each circle with a factorization-based co-clustering approach. CBIG allows the discovery of overlapping functional SAC members and can also automatically determine the optimal number of SACs depending on the underlying data. This article makes three main contributions. (1) CBIG helps to identify SACs in BIGs, to advance current studies that concentrate on generic social information graphs, such as personal or academic graph analysis. (2) CBIG simultaneously co-factorizes three channels of information from different perspectives: the topological structure (the audiences), the textual features of the audiences (features), and the correlations between those features. The results are, then integrated based on a consensus principle, which effectively addresses inconsistencies in the BIG for the best performance. Further, this approach provides an in-depth functional knowledge about an organization's customers and their relevance to the business; (3) I validate CBIG's effectiveness through solid experiments on 13 real-world enterprise datasets

against nine other classical and state-of-the-art algorithms.

However, as with all studies, our approach has some limitations. In the real world, enterprises may prefer a more simple but effective methods for decision-making. One complication of our algorithm is that its three parameters α, β, and σ, require careful tuning to make appropriate trade-offs between the importance of the graph structure information, the feature correlations, and the consistency of the factorization results. While different companies may require different weights for each stream of information to achieve good performance, unfortunately, these weights are not easy to set in a purely unsupervised setting. In the future, I intend to investigate how to incorporate prior knowledge into the process of social audience clustering and design an approach to learn the weight of each component automatically.

Chapter 7

Conclusion

This thesis aims to study that how to more effectively embed the nodes of a graph into a compact space for the tasks which are most related to the real-world applications. I have studied the research objective from four coherently linked perspectives: (1) How to unify the traditional two-step embedding work-flow into one smooth embedding procedure to avoid the inconsistency between the embedding architecture and classifier; (2) How to learn a universal embedding for all sources of nodes in a graph, so one single embedding can be used to represent the entire heterogeneous information graph; (3) How to smoothly regularize the embedding with a certain distribution during the learning procedure for a more robust embedding;(4) How to automatically generate a human-understandable explanation of each cluster of nodes in the graph and applied the algorithm in the real business world.

Specifically, I proposed four algorithms to effectively solve the mentioned challenges. Specifically, I conduct a solid research work and proposes a novel approach, graph ladder networks (GLN), which (1) unifies both representation and classifier model learning into one framework; and (2) integrates both the structure and content information of a network by a convolution network architecture; I develop a novel adversarial framework to not only minimize the reconstruction errors of the graph structure but also to enforce the latent codes to match a prior distribution; I design a universal representation model for heterogeneous graphs to learn representations for all sources of nodes; I proposed a co-factorization based data mining approach, CBIG, to co-cluster enterprise information graph for functional group discovery and

understanding. Each topic and corresponding algorithm are evaluated with related tasks including: graph classification, graph node clustering, graph visualization and link prediction. All experimental studies with real-world data sets have proved the effectiveness of each algorithm.

Chapter 8

Future Work

My future work will still focus on graph related topics and try to employ our algorithms in real-world applications.

One of potential future works is to find a elegant way for representing the raw graph data from various perspectives with a single matrix. The way how existing algorithms extract information from graph data is to purely construct topological structure (normally in the form of an adjacency matrix) or purely use a binary matrix with fixed dimension to present features of every node, or simply feed these two kinds of matrices into the framework to train the model. From my best knowledge, currently, there is no work discovering a way which simultaneously represents both topological information and contextual information based on their nature inner-relationship. By doing so, I are not only have a more efficient approach to represent the raw graph data and reduce the computation cost for the training models, but also could enhance the performance of tasks like graph clustering and classification since the matrix naturally contain various sources of information from the graph.

Another thought is to introduce attention mechanisms and generative adversarial mechanisms into graph domain and conduct some works to tailor attention approaches and GANs for graph. For example, I may build a architecture to learn the nature and unique distribution of the target graph data and then apply generative adversarial mechanism to force the representation to compromise the learned distribution.

Bibliography

[1] Jeong, Hawoong and Mason, Sean P and Barabási, A-L and Oltvai, Zoltan N, "Lethality and centrality in protein networks," *Nature*, vol. 411, no. 6833, pp. 41–42, 2001.

[2] Wang, Haishuai and Zhang, Peng and Zhu, Xingquan and Tsang, Ivor Wai-Hung and Chen, Ling and Zhang, Chengqi and Wu, Xindong, "Incremental subgraph feature selection for graph classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 1, pp. 128–142, 2017.

[3] Pan, Shirui and Wu, Jia and Zhu, Xingquan and Zhang, Chengqi and Philip, S Yu, "Joint structure feature exploration and regularization for multi-task graph classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 3, pp. 715–728, 2016.

[4] Parthasarathy, Srinivasan and Ruan, Yiye and Satuluri, Venu, "Community discovery in social networks: Applications, methods and emerging trends," in *Social network data analytics*. Springer, 2011, pp. 79–113.

[5] Wang, Haishuai and Wu, Jia and Pan, Shirui and Zhang, Peng and Chen, Ling, "Towards large-scale social networks with online diffusion provenance detection," *Computer Networks*, 2016.

[6] Hu, Ruiqi and Pan, Shirui and Long, Guodong and Zhu, Xingquan and Jiang, Jing and Zhang, Chengqi, "Co-clustering enterprise social networks," in *Neural Networks (IJCNN), 2016 International Joint Conference on*. IEEE, 2016, pp. 107–114.

[7] Li, Huajing and Councill, Isaac and Lee, Wang-Chien and Giles, C Lee, "Citeseerx: an architecture and web service design for an academic document search engine," in *Proceedings of the 15th international conference on World Wide Web*. ACM, 2006, pp. 883–884.

[8] Pan, Shirui and Wu, Jia and Zhu, Xingquan and Zhang, Chengqi, "Graph ensemble boosting for imbalanced noisy graph stream classification," *IEEE transactions on cybernetics*, vol. 45, no. 5, pp. 954–968, 2015.

[9] Kipf, Thomas N and Welling, Max, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[10] Wang, Zhitao and Chen, Chengyao and Li, Wenjie, "Predictive network representation learning for link prediction," in *In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2017, pp. 969–972.

[11] Wang, Chun and Pan, Shirui and Long, Guodong and Zhu, Xingquan and Jiang, Jing, "MGAE: Marginalized graph autoencoder for graph clustering," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 2017, pp. 889–898.

[12] Cui, Peng and Wang, Xiao and Pei, Jian and Zhu, Wenwu, "A survey on network embedding," *arXiv preprint arXiv:1711.08752*, 2017.

[13] Perozzi, Bryan and Al-Rfou, Rami and Skiena, Steven, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 701–710.

[14] Yang, Cheng and Liu, Zhiyuan and Zhao, Deli and Sun, Maosong and Chang, Edward Y, "Network representation learning with rich text information," in

Proceedings of the 24th International Joint Conference on Artificial Intelligence, Buenos Aires, Argentina, 2015, pp. 2111–2117.

[15] Tang, Jian and Qu, Meng and Wang, Mingzhe and Zhang, Ming and Yan, Jun and Mei, Qiaozhu, "Line: Large-scale information network embedding," in *Proceedings of the 24th International Conference on World Wide Web*. ACM, 2015, pp. 1067–1077.

[16] Mei, Qiaozhu and Cai, Deng and Zhang, Duo and Zhai, ChengXiang, "Topic modeling with network regularization," in *Proceedings of the 17th international conference on World Wide Web*. ACM, 2008, pp. 101–110.

[17] Makhzani, Alireza and Shlens, Jonathon and Jaitly, Navdeep and Goodfellow, Ian and Frey, Brendan, "Adversarial autoencoders," *arXiv preprint arXiv:1511.05644*, 2015.

[18] Luscombe, Nicholas M and Laskowski, Roman A and Thornton, Janet M, "Amino acid–base interactions: a three-dimensional analysis of protein–dna interactions at an atomic level," *Nucleic acids research*, vol. 29, no. 13, pp. 2860–2874, 2001.

[19] Hummon, Norman P and Dereian, Patrick, "Connectivity in a citation network: The development of dna theory," *Social networks*, vol. 11, no. 1, pp. 39–63, 1989.

[20] Rice, Ronald E and Borgman, Christine L and Reeves, Byron, "Citation networks of communication journals, 1977–1985 cliques and positions, citations made and citations received," *Human communication research*, vol. 15, no. 2, pp. 256–283, 1988.

[21] Verspagen, Bart, "Mapping technological trajectories as patent citation networks: A study on the history of fuel cell research," *Advances in Complex*

Systems, vol. 10, no. 01, pp. 93–115, 2007.

[22] Mikolov, Tomas and Chen, Kai and Corrado, Greg and Dean, Jeffrey, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[23] Le, Quoc and Mikolov, Tomas, "Distributed representations of sentences and documents." in *International Conference on Machine Learning*, 2014, pp. 1188–1196.

[24] Blei, David M and Ng, Andrew Y and Jordan, Michael I, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[25] Pan, Shirui and Wu, Jia and Zhu, Xingquan and Zhang, Chengqi and Wang, Yang, "Tri-party deep network representation," *Network*, vol. 11, no. 9, p. 12, 2016.

[26] Rasmus, Antti and Berglund, Mathias and Honkala, Mikko and Valpola, Harri and Raiko, Tapani, "Semi-supervised learning with ladder networks," in *Advances in neural information processing systems*, 2015, pp. 3546–3554.

[27] Zhang, Daokun and Yin, Jie and Zhu, Xingquan and Zhang, Chengqi, "Network representation learning: A survey," *arXiv preprint arXiv:1801.05852*, 2017.

[28] Cai, Hongyun, Vincent W. Zheng, and Kevin Chang, "A comprehensive survey of graph embedding: Problems, techniques and applications," *IEEE Transactions on Knowledge and Data Engineering*, 2017.

[29] Goyal, Palash and Ferrara, Emilio, "Graph embedding techniques, applications, and performance: A survey," *arXiv preprint arXiv:1705.02801*, 2017.

[30] Grover, Aditya, and Jure Leskovec, "node2vec: Scalable feature learning for networks," in *In Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, 2016, pp. 855–864.

[31] Jian Tang and Meng Qu and Mingzhe Wang and Ming Zhang and Jun Yan and Qiaozhu Mei, "LINE: large-scale information network embedding," in *In Proceedings of the 24th International Conference on World Wide Web*, 2015, pp. 1067–1077.

[32] Hofmann, Thomas, "Probabilistic latent semantic analysis," in *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence.* Morgan Kaufmann Publishers Inc., 1999, pp. 289–296.

[33] Tang, Lei and Liu, Huan, "Leveraging social media networks for classification," *Data Mining and Knowledge Discovery*, vol. 23, no. 3, pp. 447–478, 2011.

[34] Cao, Shaosheng, Wei Lu, and Qiongkai Xu, "Grarep: Learning graph representations with global structural information," in *In Proceedings of the 24th ACM International on Conference on Information and Knowledge Management.* ACM, 2015, pp. 891–900.

[35] Ou, Mingdong, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu, "Asymmetric transitivity preserving graph embedding." in *In Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 1105–1114.

[36] Wang, Xiao, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang, "Community preserving network embedding." in *Association for the Advancement of Artificial Intelligence*, 2017, pp. 203–209.

[37] Cai, Deng and He, Xiaofei and Wu, Xiaoyun and Han, Jiawei, "Non-negative matrix factorization on manifold," in *Data Mining, 2008. ICDM'08. Eighth*

IEEE International Conference on. IEEE, 2008, pp. 63–72.

[38] Qiu, Jiezhong and Dong, Yuxiao and Ma, Hao and Li, Jian and Wang, Kuansan and Tang, Jie, "Network embedding as matrix factorization: Unifyingdeepwalk, line, pte, and node2vec," *arXiv preprint arXiv:1710.02971*, 2017.

[39] L. Gao, H. Yang, C. Zhou, J. Wu, S. Pan, and Y. Hu, "Active discriminative network representation learning," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*. International Joint Conferences on Artificial Intelligence Organization, 7 2018, pp. 2142–2148. [Online]. Available: https://doi.org/10.24963/ijcai.2018/296

[40] X. Shen, S. Pan, W. Liu, Y.-S. Ong, and Q.-S. Sun, "Discrete network embedding," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*. International Joint Conferences on Artificial Intelligence Organization, 7 2018, pp. 3549–3555. [Online]. Available: https://doi.org/10.24963/ijcai.2018/493

[41] Wang, Daixin, Peng Cui, and Wenwu Zhu, "Structural deep network embedding," in *In Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2016, pp. 1225–1234.

[42] Cao, Shaosheng and Lu, Wei and Xu, Qiongkai, "Deep neural networks for learning graph representations." in *Association for the Advancement of Artificial Intelligence*, 2016, pp. 1145–1152.

[43] Kipf, Thomas N and Welling, Max, "Variational graph auto-encoders," 2016.

[44] Kingma, Diederik P and Welling, Max, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[45] Damashek, Marc, "Gauging similarity with n-grams: Language-independent categorization of text," *Science*, vol. 267, no. 5199, p. 843, 1995.

[46] Chi, Lianhua and Li, Bin and Zhu, Xingquan, "Context-preserving hashing for fast text classification." in *Proceedings of the 2014 SIAM International Conference on Data Mining*. SIAM, 2014, pp. 100–108.

[47] Chim, Hung and Deng, Xiaotie, "Efficient phrase-based document similarity for clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 9, pp. 1217–1229, 2008.

[48] Pan, Shirui and Zhu, Xingquan, "Graph classification with imbalanced class distributions and noise," in *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*. AAAI Press, 2013, pp. 1586–1592.

[49] Hochreiter, Sepp and Schmidhuber, Jürgen, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[50] Kiros, Ryan and Zhu, Yukun and Salakhutdinov, Ruslan R and Zemel, Richard and Urtasun, Raquel and Torralba, Antonio and Fidler, Sanja, "Skip-thought vectors," in *Advances in neural information processing systems*, 2015, pp. 3294–3302.

[51] Kalchbrenner, Nal and Grefenstette, Edward and Blunsom, Phil, "A convolutional neural network for modelling sentences," *arXiv preprint arXiv:1404.2188*, 2014.

[52] Defferrard, Michaël and Bresson, Xavier and Vandergheynst, Pierre, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in neural information processing systems*, 2016, pp. 3837–3845.

[53] Veropoulos, Konstantinos and Campbell, Colin and Cristianini, Nello and others, "Controlling the sensitivity of support vector machines," in *Proceedings of the international joint conference on AI*, 1999, pp. 55–60.

[54] Valpola, Harri, "From neural pca to deep unsupervised learning," pp. 143–171, 2015.

[55] Belkin, Mikhail and Niyogi, Partha and Sindhwani, Vikass, "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples," *Journal of machine learning research*, vol. 7, no. Nov, pp. 2399–2434, 2006.

[56] Liu, Xianming and Zhai, Deming and Zhao, Debin and Zhai, Guangtao and Gao, Wen, "Progressive image denoising through hybrid graph laplacian regularization: a unified framework," *IEEE Transactions on image processing*, vol. 23, no. 4, pp. 1491–1503, 2014.

[57] Donahue, Jeff and Krähenbühl, Philipp and Darrell, Trevor, "Adversarial feature learning," *arXiv preprint arXiv:1605.09782*, 2016.

[58] Zhao, Junbo and Mathieu, Michael and LeCun, Yann, "Energy-based generative adversarial network," *arXiv preprint arXiv:1609.03126*, 2016.

[59] Dumoulin, Vincent and Belghazi, Ishmael and Poole, Ben and Mastropietro, Olivier and Lamb, Alex and Arjovsky, Martin and Courville, Aaron, "Adversarially learned inference," *arXiv preprint arXiv:1606.00704*, 2016.

[60] Radford, Alec, Luke Metz, and Soumith Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[61] Vincent, Pascal, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of machine learning research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

[62] Glover, John, "Modeling documents with generative adversarial networks," *arXiv preprint arXiv:1612.09122*, 2016.

[63] Chung, Junyoung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[64] Zhang, Daokun and Yin, Jie and Zhu, Xingquan and Zhang, Chengqi, "User profile preserving social network embedding," in *In Proceedings of IJCAI*. AAAI Press, 2017, pp. 3378–3384.

[65] Goodfellow, Ian and Pouget-Abadie, Jean and Mirza, Mehdi and Xu, Bing and Warde-Farley, David and Ozair, Sherjil and Courville, Aaron and Bengio, Yoshua, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[66] Dai, Quanyu, Qiang Li, Jian Tang, and Dan Wang, "Adversarial network embedding," *arXiv preprint arXiv:1711.07838*, 2017.

[67] Tian, Fei and Gao, Bin and Cui, Qing and Chen, Enhong and Liu, Tie-Yan, "Learning deep representations for graph clustering." in *Association for the Advancement of Artificial Intelligence*, 2014, pp. 1293–1299.

[68] Leskovec, Jure and Mcauley, Julian J, "Learning to discover social circles in ego networks," in *Advances in neural information processing systems*, 2012, pp. 539–547.

[69] Chang, Jonathan and Blei, David M, "Relational topic models for document networks," in *International Conference on Artificial Intelligence and Statistics*, 2009, pp. 81–88.

[70] Xia, Rongkai and Pan, Yan and Du, Lei and Yin, Jian, "Robust multi-view

spectral clustering via low-rank and sparse decomposition." in *Association for the Advancement of Artificial Intelligence*, 2014, pp. 2149–2155.

[71] Van Der Maaten, Laurens, "Accelerating t-sne using tree-based algorithms." *Journal of machine learning research*, vol. 15, no. 1, pp. 3221–3245, 2014.

[72] Bottou, Léon, "Stochastic gradient learning in neural networks," *Proceedings of Neuro-Nımes*, vol. 91, no. 8, 1991.

[73] Morin, Frederic and Bengio, Yoshua, "Hierarchical probabilistic neural network language model." in *Aistats*, vol. 5. Citeseer, 2005, pp. 246–252.

[74] Mikolov, Tomas and Joulin, Armand and Chopra, Sumit and Mathieu, Michael and Ranzato, Marc'Aurelio, "Learning longer memory in recurrent neural networks," *arXiv preprint arXiv:1412.7753*, 2014.

[75] Ley, Michael, "The dblp computer science bibliography: Evolution, research issues, perspectives," in *International Symposium on String Processing and Information Retrieval*. Springer, 2002, pp. 1–10.

[76] Yang, Yiming, "An evaluation of statistical approaches to text categorization," *Information retrieval*, vol. 1, no. 1-2, pp. 69–90, 1999.

[77] Turban, Efraim and Bolloju, Narasimha and Liang, Ting-Peng, "Enterprise social networking: Opportunities, adoption, and risk mitigation," *Journal of Organizational Computing and Electronic Commerce*, vol. 21, no. 3, pp. 202–220, 2011.

[78] Barnetta, A, "Fortune 500 companies in second life–activities, their success measurement and the satisfaction level of their projects," Ph.D. dissertation, Master thesis ETH Zürich, 2009.

[79] Butow, Eric and Taylor, Kathleen, *How to Succeed in Business Using LinkedIn: Making Connections and Capturing Opportunities on the World's# 1 Business Networking Site.* AMACOM Div American Mgmt Assn, 2008.

[80] Drury, Glen, "Opinion piece: Social media: Should marketers engage and how can it be done effectively?" *Journal of Direct, Data and Digital Marketing Practice*, vol. 9, no. 3, pp. 274–277, 2008.

[81] UK, Social Enterprise, *State of Social Enterprise Survey 2013.* Social Enterprise UK, 2013.

[82] Yang, Wan-Shiou and Dia, Jia-Ben and Cheng, Hung-Chi and Lin, Hsing-Tzu, "Mining social networks for targeted advertising," in *System Sciences, 2006. HICSS'06. Proceedings of the 39th Annual Hawaii International Conference on*, vol. 6. IEEE, 2006, pp. 137a–137a.

[83] Sadovykh, Valeria and Sundaram, David and Piramuthu, Selwyn, "Do online social networks support decision-making?" *Decision Support Systems*, vol. 70, pp. 15–30, 2015.

[84] Huang, Hong and Tang, Jie and Wu, Sen and Liu, Lu and others, "Mining triadic closure patterns in social networks," in *Proceedings of the 23rd international conference on World wide web.* ACM, 2014, pp. 499–504.

[85] Heller Baird, Carolyn and Parasnis, Gautam, "From social media to social customer relationship management," *Strategy & Leadership*, vol. 39, no. 5, pp. 30–37, 2011.

[86] Huang, Hong and Tang, Jie and Liu, Lu and Luo, JarDer and Fu, Xiaoming, "Triadic closure pattern analysis and prediction in social networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 12, pp. 3374–3389, 2015.

[87] Yang, Jaewon and Leskovec, Jure, "Community-affiliation graph model for overlapping network community detection," in *Data Mining (ICDM), 2012 IEEE 12th International Conference on.* IEEE, 2012, pp. 1170–1175.

[88] Yang, Jaewon and Leskovec, Jure, "Overlapping community detection at scale: a nonnegative matrix factorization approach," in *Proceedings of the sixth ACM international conference on Web search and data mining.* ACM, 2013, pp. 587–596.

[89] Balasubramanyan, Ramnath and Cohen, William W, "Block-LDA: Jointly modeling entity-annotated text and entity-entity links." in *Proceedings of the 2011 SIAM International Conference on Data Mining*, vol. 11. SIAM, 2011, pp. 450–461.

[90] Sun, Yizhou and Han, Jiawei and Gao, Jing and Yu, Yintao, "itopic-model: Information network-integrated topic modeling," in *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on.* IEEE, 2009, pp. 493–502.

[91] Gu, Quanquan and Zhou, Jie, "Co-clustering on manifolds," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, 2009, pp. 359–368.

[92] Wang, Hua and Nie, Feiping and Huang, Heng and Makedon, Fillia, "Fast non-negative matrix tri-factorization for large-scale data co-clustering," in *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, vol. 22, no. 1, 2011, p. 1553.

[93] Li, Yusheng and Shang, Yilun and Yang, Yiting, "Clustering coefficients of large networks," *Information Sciences*, 2016.

[94] Pan, Shirui and Wu, Jia and Zhu, Xingquan, "Cogboost: Boosting for fast cost-sensitive graph classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 11, pp. 2933–2946, 2015.

[95] Wang, Haishuai and Wu, Jia and Pan, Shirui and Zhang, Peng and Chen, Ling, "Towards large-scale social networks with online diffusion provenance detection," *Computer Networks*, vol. 114, pp. 154–166, 2017.

[96] Pan, Shirui and Wu, Jia and Zhu, Xingquan and Long, Guodong and Zhang, Chengqi, "Finding the best not the most: regularized loss minimization subgraph selection for graph classification," *Pattern Recognition*, vol. 48, no. 11, pp. 3783–3796, 2015.

[97] Pan, Shirui and Wu, Jia and Zhu, Xingquan and Long, Guodong and Zhang, Chengqi, "Task sensitive feature exploration and learning for multitask graph classification," *IEEE transactions on cybernetics*, vol. 47, no. 3, pp. 744–758, 2017.

[98] Domingos, Pedro, "Mining social networks for viral marketing," *IEEE Intelligent Systems*, vol. 20, no. 1, pp. 80–82, 2005.

[99] Ferreira, Leonardo N and Zhao, Liang, "Time series clustering via community detection in networks," *Information Sciences*, vol. 326, pp. 227–242, 2016.

[100] Tuten, Tracy L, *Advertising 2.0: social media marketing in a web 2.0 world*. Greenwood Publishing Group, 2008.

[101] Lo, Siaw Ling and Chiong, Raymond and Cornforth, David, "Ranking of high-value social audiences on twitter," *Decision Support Systems*, vol. 85, pp. 34–48, 2016.

[102] van Dam, Jan-Willem and van de Velden, Michel, "Online profiling and clustering of facebook users," *Decision Support Systems*, vol. 70, pp. 60–72, 2015.

[103] Cai, Qing and Gong, Maoguo and Ma, Lijia and Ruan, Shasha and Yuan, Fuyan and Jiao, Licheng, "Greedy discrete particle swarm optimization for large-scale social network clustering," *Information Sciences*, vol. 316, pp. 503–516, 2015.

[104] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, "Adversarially regularized graph autoencoder for graph embedding." in *IJCAI*, 2018, pp. 2609–2615.

[105] Yang, Jaewon and McAuley, Julian and Leskovec, Jure, "Community detection in networks with node attributes," in *Data Mining (ICDM), 2013 IEEE 13th international conference on.* IEEE, 2013, pp. 1151–1156.

[106] Lee, Daniel D and Seung, H Sebastian, "Algorithms for non-negative matrix factorization," in *Advances in neural information processing systems*, 2001, pp. 556–562.

[107] Xu, Wei and Liu, Xin and Gong, Yihong, "Document clustering based on non-negative matrix factorization," in *Proceedings of international ACM SIGIR conference on Research and development in informaion retrieval.* ACM, 2003, pp. 267–273.

[108] Ding, Chris and Li, Tao and Peng, Wei and Park, Haesun, "Orthogonal non-negative matrix t-factorizations for clustering," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, 2006, pp. 126–135.

[109] Ding, Chris and Li, Tao and Jordan, Michael and others, "Convex and semi-nonnegative matrix factorizations," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, no. 1, pp. 45–55, 2010.

[110] Niyogi, X, "Locality preserving projections," in *Neural information processing systems*, vol. 16. MIT, 2004, p. 153.

[111] Hartigan, John A and Wong, Manchek A, "Algorithm as 136: A k-means clustering algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[112] Li, Li-Xin and Wu, Lin and Zhang, Hui-Sheng and Wu, Fang-Xiang, "A fast algorithm for nonnegative matrix factorization and its convergence," *IEEE transactions on neural networks and learning systems*, vol. 25, no. 10, pp. 1855–1863, 2014.

[113] Lin, Chih-Jen, "On the convergence of multiplicative update algorithms for nonnegative matrix factorization," *IEEE Transactions on Neural Networks*, vol. 18, no. 6, pp. 1589–1596, 2007.

[114] Chen, Yi-Wei and Lin, Chih-Jen, "Combining svms with various feature selection strategies," in *Feature extraction*. Springer, 2006, pp. 315–324.

[115] Powers, David Martin, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," *Journal of Machine Learning Technologiesn*, vol. 2, no. 1, pp. 37–63, 2011.