EXPLORING NODE ATTRIBUTES FOR DATA MINING IN ATTRIBUTED GRAPHS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Jihwan Lee

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

May 2019

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

Dr. Sunil Prabhakar, Chair

Department of Computer Science

Dr. Jennifer Neville

Department of Computer Science

Dr. Bruno Ribeiro

Department of Computer Science

Dr. Dan Goldwasser

Department of Computer Science

Approved by:

Dr. Voicu S. Popescu

Head of the Department Graduate Program

To my family

For their endless love, support, and encouragement

ACKNOWLEDGMENTS

I am indebted to many people on finishing my Ph.D which was an unforgettable experience in my entire life. I would never have been able to finish my dissertation without the guidance of my advisor, committee members, help from friends, and support from my family.

First of all, I dedicate my sincere gratitude to my advisor, Dr. Sunil Prabhakar, for his constant support and continuous guidance to cultivate me as an independent researcher throughout my doctoral study. I appreciate that he decided to be willing to serve as my advisor when I first joined Purdue. It was fortunate to meet and work with him at the very important moment when starting my Ph.D degree. He gave me a lot of opportunities to explore various research topics while guiding me to move forward in the right direction. Whenever I struggled with making progress on my research, he always built me up with kindness and a sympathetic ear. He has been a great mentor and wonderful inspiration at all times during my pursuing Ph.D.

I also appreciate my committee members. Dr. Jennifer Neville gave me a great insight on the world of graphs/networks through classes and research discussions. Dr. Bruno Ribeiro guided me to keep focusing on fundamental properties of graphs when conducting research. Dr. Dan Goldwasser helped me widen my perspective on the intersection of various research areas.

I was also fortunate to have a lot of friends at Purdue who enjoyed delightful moments with me together - Romila Pradhan, Keehwan Park, Siarhei Bykau, Koray Manchuhan, Hogun Park, Jaewoo Lee, Jaewoo Shin, Hongjun Choi, Minkwang Choi, and many others. I will never forget all the memories we shared while spending tough times in the boring city.

Lastly, I would like to thank my lovely wife, Soobin Seo, for staying with me and being supportive during this long journey, and encouraging me to finish my

dissertation. She always makes me think positively and move forward consistently. I have no doubt that I would never have made it through this process without her endless devotion and love. I also wish to thank my daughter, Lena Lee, and my son, Eugene Lee, who are the greatest blessing and gift to me and my wife. They always bring delightful joys to our life and keep us smiling.

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

LIST OF FIGURES

SYMBOLS

$G = (V, E)$	attributed graph
$V = \{u_1, u_2, \ldots, u_{\|V\|}\}$	set of nodes in G
E	set of edges in G
$\mathcal{AG} = (\mathcal{V}, \mathcal{E})$	association graph
$\mathcal{V} = \{c_1, c_2, \ldots, c_{\|\mathcal{V}\|}\}$	set of clusters in $\mathcal{AG}$
$\mathcal{E}$	set of attribute associations in $\mathcal{AG}$
$\mathbf{a} = (a^1, a^2, \ldots, a^l)$	attribute vector of size l
Δ	attribute association
σ	*freq_support*
η	*size_support*
δ_G	density of graph G
Ψ_c	*p-value* of cluster c
$TS(u, v)$	tie-strength between node u and v
$\Gamma(\cdot)$	set of neighbors
$\tilde{G}_c$	subgraph of nodes within cluster c
U	number of nodes
C	number of communities
A	number of attributes
K	number of topics
θ	multinomial distribution over topics of a node
λ	multinomial distribution over topics of a community
π_u	community memberships of node u
ϕ_{az}	multinomial distribution over values of attribute a given topic z

μ_u	probability that node u is biased to communities
z_{ua}	node u's topic on attribute a
c_{ua}	node u's community that is relevant to attribute a
x_{ua}	node u's value on attribute a
y_{ua}	node u's bias for attribute a
$G = (V, E, X)$	attributed network
n	number of nodes
l	number of attributes
s_{ij}	weight of edge e_{ij}
τ	penalty term to the reconstruction error of non-zero values in attribute vectors
$\mathbf{y_i}$	attribute embedding of node v_i
$\mathbf{z_i}$	structural embedding of node v_i
$\mathbf{h_i}$	joint representation of node v_i
$W_1^{(k)}, \mathbf{b_1}^{(k)}$	k-th layer weights and biases in attribute modeling
$W_2^{(k)}, \mathbf{b_2}^{(k)}$	k-th layer weights and biases in structure modeling
$W_3^{(k)}, \mathbf{b_3}^{(k)}$	k-th layer weights and biases in joint modeling
m_1, m_2, m_3	number of layers for each modeling component
ω	hyperparameter that controls weights on the latent representation from modeling node attributes
ζ	negative penalty to control the importance of attribute association
$\mathcal{N}_i$	neighbors of node v_i
σ	activation function
R	regularization function
$\mathcal{L}_{\{sim,ass,net\}}$	loss functions for attribute similarity, attribute association, and network structure
p	in-community link probability in a synthetic graph
q	cross-community link probability in a synthetic graph

r	number of distinct attribute vectors in a community in a synthetic graph

ABSTRACT

Lee, Jihwan Ph.D., Purdue University, May 2019. Exploring Node Attributes for Data Mining in Attributed Graphs. Major Professor: Sunil Prabhakar.

Graphs have attracted researchers in various fields in that many different kinds of real-world entities and relationships between them can be represented and analyzed effectively and efficiently using graphs. In particular, researchers in data mining and machine learning areas have developed algorithms and models to understand the complex graph data better and perform various data mining tasks. While a large body of work exists on graph mining, most existing work does not fully exploit attributes attached to graph nodes or edges.

In this dissertation, we exploit node attributes to generate better solutions to several graph data mining problems addressed in the literature. First, we introduce the notion of statistically significant attribute associations in attribute graphs and propose an effective and efficient algorithm to discover those associations. The effectiveness analysis on the results shows that our proposed algorithm can reveal insightful attribute associations that cannot be identified using the earlier methods focused solely on frequency. Second, we build a probabilistic generative model for observed attributed graphs. Under the assumption that there exist hidden communities behind nodes in a graph, we adopt the idea of latent topic distributions to model a generative process of node attribute values and link structure more precisely. This model can be used to detect hidden communities and profile missing attribute values. Lastly, we investigate how to employ node attributes to learn latent representations of nodes in lower dimensional embedding spaces and use the learned representations to improve the performance of data mining tasks over attributed graphs.

1. INTRODUCTION

This dissertation investigates the problem of exploring node attributes and learning correlations between node attributes and graph structure from attributed graphs. It aims at discovering insightful and interesting attribute patterns and performing data mining tasks better by considering node attributes in complex network data.

Many different kinds of real-world objects establish relationships to each other and are usually represented by a certain type of abstract data type, called graph. A graph consists of a set of nodes and a set of edges connecting nodes, and its structural feature itself fits very well to the formation of connected objects. For example, in a social network, each individual user corresponds to a node and a connection or relationship between users is formed by an edge in a graph. In a citation network, a node represents a publication and any two publications forms a directed edge if one cites the other. There are a variety of graph types according to their structural/temporal/stochastic characteristics, as shown in Figure 1.1, and our main focus is on attributed graphs in this dissertation. Note that we use the terms *graph* and *network* interchangeably throughout this dissertation.

The explosive increase of relational data in terms of both the amount and the variety has led to the popularity of graphs in various fields. As graphs are more widely used in a broad range of commercial and scientific applications, the needs to analyze the graph data and find new valuable information from the graph data become more important. The availability of attributes associated with nodes and edges has the potential to enrich our learning and mining tasks to yield more insightful and accurate predictions. However, existing work has only recently begun to take these attributes into account. In this dissertation, we develop novel methods that exploit node attribute information to yield novel insights and improve the accuracy of predictions over existing methods.

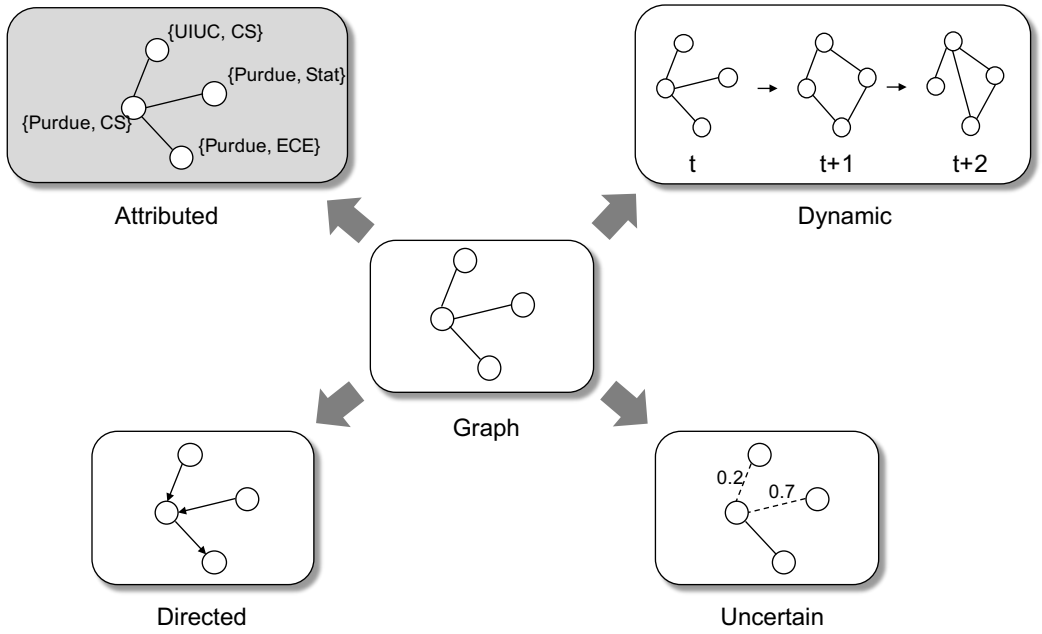


Fig. 1.1.: Taxonomy of graph. In attributed graphs, the nodes of a graph have some attribute values. If a graph evolves over time, it is called a dynamic graph. Also, The edges can be directional in directed graphs, and they may appear or not with some probability in uncertain graphs.

More specifically, this dissertation provides answers to the following questions: Given an attributed graph, (1) Which attribute values co-occurring through any connected nodes in the graph could bring us most significant information by themselves? (2) How are the node attributes correlated with link structure in generating the graph? (3) How should the node attributes be used together with graph structure for learning richer feature representations of the nodes?

1.1 Motivation

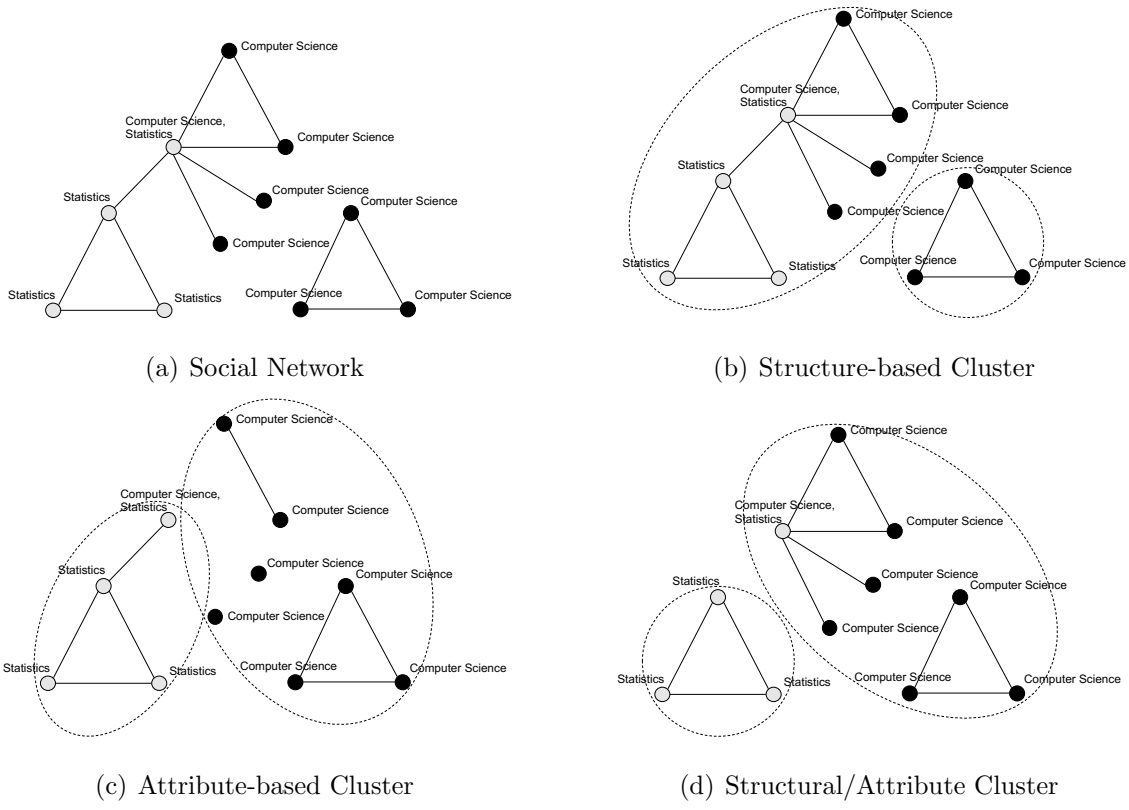
For the past decades, relational data has grown at a tremendous rate in a wide range of domains such as the Internet and the World-Wide Web [1–3], scientific citation and collaboration [4, 5], epidemiology [6–9], communication analysis [10],

metabolism [11, 12], ecosystems [13, 14], bioinformatics [15, 16], fraud and terrorist analysis [17, 18], and many others. The links in these data may represent citations, friendship, associations, metabolic functions, communications, co-locations, shared mechanisms, or many other explicit or implicit relationships.

Such relational data basically encode how objects relate to each other, and looking at the relationships of an object to its neighbors can help us understand the role of the object in the network (i.e., a person having a number of followers in a social network may play a role of hub in information propagation) and detect latent communities (i.e., people within a group interact with each other more frequently than with those outside the group). Sometimes the objects are associated with attributes describing their own properties. For example, each individual in a social network has its personal information such as gender, age, school, employer, and so on. Not only they reveal the characteristics of an object by themselves, but it also has been shown that the attributes affect the formation of relationships among individual objects in a network [19]. In other words, the attributes may be strongly correlated with the relationships, which gives us a chance to deal with relational data much better for machine learning tasks by taking into account both the relationships and the node attributes. As an illustrating example of the benefit of using node attributes, Figure 1.2 shows that the clusters of the nodes could be formed differently depending on which of structure and attribute is considered in graph clustering.

The proliferation of relational data in the real-world has given rise to the explosive growth of research on graphs. Researchers have come up with diverse interesting problems such as graph clustering, link prediction, node classification, community detection, and so on. These problems were first considered for graphs without node attributes and some have been extended to attributed graphs.

From the perspective of node attributes themselves, one might be interested in what insights we expect from seeing the node attributes over the entire graph. Given a pair of nodes that are connected to each other, one interesting observation on the relationship is which attribute values co-occur through the connection. Homophily



(a) Social Network

(b) Structure-based Cluster

(c) Attribute-based Cluster

(d) Structural/Attribute Cluster

Fig. 1.2.: Social network example with an attribute "major". Figure 1.2(a) shows a small social network graph where a node corresponds to an individual and an edge represents a friend relationship between two individuals. Each of the nodes is associated with an attribute which describes a subject she has majored. Figure 1.2(b) shows clusters based on node connectivity, i.e., friend relationship. Individuals within clusters are closely connected, but they could have different attribute values. Figure 1.2(c) shows another set of clusters based on attribute similarity, i.e., majors. However, the friend relationship may be lost due to the partitioning so that individuals are quite isolated in one of the clusters. Figure 1.2(d) shows clusters based on both structure and attribute information. This result balances the structural and attribute similarities. That is, individuals within clusters are closely connected, and meanwhile, they are homogeneous on major.

is the tendency of individuals to associate and bond with similar others. That is, if the relationship is homophilic, the connected nodes are likely to share the same or similar attribute values. Heterophily is the opposite case. The presence of those kinds of relationships has been discovered in a vast array of network studies by social scientists [19–21], yet they have only focused on associations between the same set of attributes. This problem can be generalized to associations between any two subsets of attributes and these associations may bring us more insightful patterns that are helpful for understanding the relationships between nodes, or even the entire network, better. Especially, we may want to discover only some associations which are really meaningful in terms of their statistical significance, but it is challenging to identify which associations are statistically significance in a naive way due to the extremely large number of possible associations in a large graph.

In addition to the node attribute patterns, it is worth considering the node attributes in different machine learning tasks because of their high applicability. Numerous machine learning tasks on graphs still rely nearly on relational learning in the graph structure. However, previous research has shown that the connectivity information in relational data is essentially not sufficient but necessary for achieving high performance in many applications. The relationships between nodes are not identical across different pairs of connected nodes. For example, in Facebook, user A and user B might make a friendship relationship because they used to be classmates while user A and user C are in a friend relationship because they work for the same company. Apparently, the different types of relationships have to be considered and handled differently while performing machine learning tasks since they actually play different roles depending on tasks. In most cases, the relationship types are implicit, and if this is the case then they need to be identified. Although it is non-trivial to discriminate the implicit relationship types, we can exploit node attributes to estimate the types if they are available. Fortunately, many relational data are provided with such extra information even if it is not necessary a form of node attribute, and it has been considered as critical to understand how node attributes and graph structure

are dependent of each other [19, 22–30]. Even though many previous research have been done on attributed graphs for various machine learning tasks, there is still a room for improvement.

1.2 Thesis Statement and Contributions

In this dissertation, we consider the problem of finding insightful attribute patterns and using node attributes with the purpose of improvement on existing approaches to some data mining tasks. First of all, we develop an effective algorithm for mining statistically significant attribute associations in attributed graphs. The results shed light on meaningful patterns of co-occurring node attribute values between connected nodes which might be hidden by frequent ones. Second, we propose a new probabilistic generative model for attributed graphs. The proposed model resolves some concerns raised from existing models by introducing a novel generative process with various hidden factors which are inferred through sampling and optimization techniques. We demonstrate the effectiveness of the model for community detection and profiling of missing attribute values. Lastly, we investigate the problem of learning network representation on attributed graphs in which no previous research has tried to incorporate node attributes. The learned representations in lower dimensional embedding spaces for nodes can be directly used for standard machine learning tasks on relational data.

The main thesis of this dissertation can be stated as follows:

> *It is important to uncover insightful patterns of node attributes for understanding the essence of attributed graphs better, and deep understanding on the dependency between node attributes and graph structure will help improve the performance of various machine learning tasks*

1.3 Outline

The remainder of this dissertation is organized as follows.

First, in Chapter 2, we define the concept of attribute association and introduce a new algorithm that can discover statistically significant attribute associations in attributed graphs.

In Chapter 3, we build a probabilistic generative model and show how the model is used to detect hidden communities in a network with node attributes and to estimate missing attribute values.

In Chapter 4, we propose a new approach to graph embedding, especially when the graph data contain node attributes. The nodes in a graph have new representations in lower dimensional embedding space that are obtained through an optimization process and then the learned representations are directly used as inputs for different machine learning tasks.

Lastly, in Chapter 5, we summarize our contributions made in the dissertation and present future works.

2. STATISTICALLY SIGNIFICANT ATTRIBUTE ASSOCIATIONS

Recently, graphs have been widely used to represent many different kinds of real world data or observations such as social networks, protein-protein networks, road networks, and so on. In many cases, each node in a graph is associated with a set of its attributes and it is critical to not only consider the link structure of a graph but also use the attribute information to achieve more meaningful results in various graph mining tasks. Most previous works with attributed graphs take into account attribute relationships only between individually connected nodes. However, it should be greatly valuable to find out which sets of attributes are associated with each other and whether they are statistically significant or not. Mining such significant associations, we can uncover novel relationships among the sets of attributes in the graph. We propose an algorithm that can find those attribute associations efficiently and effectively, and show experimental results that confirm the high applicability of the proposed algorithm.

2.1 Introduction

Nowadays graphs have emerged as a powerful abstract data type to represent and analyze complex data in a broad range of commercial and scientific applications including social networks [31,32], bioinformatics [33], world wide web [2,34], and so on. Mining structured patterns in graphs have been actively studied in the literature and such patterns including cliques [35], subgraphs [36–38], paths [39] and trees [40] help us better understand the intrinsic characteristics of graph data. Also, when the graph data come with auxiliary information such as node attributes, such information can be applied to various application areas, e.g., community detection, link prediction, graph

clustering, network modeling, and etc. Thus, attributed graphs are more important than ever before to complex mining tasks.

While node attributes can be successfully employed to augment various mining tasks, the node attributes themselves could give us interesting patterns for better understanding graphs. Given an attributed graph where each node is associated with its attribute values, one might be interested in a pattern of node attribute values which co-occur between connected nodes. Let us call such co-occurring attribute values between two connected nodes an attribute association. This information can tell us directly the attribute patterns shared by connected nodes over the entire graph. In large scale, one might be interested in which attribute associations are most frequently observed or which attribute vector is most expected to be observed given another attribute vector in attribute associations. Looking at frequent attribute associations reveals the most dominant attribute associations in the graph by simply taking into account how many times they are held by connected nodes. Even though the frequent attribute associations give us which ones are dominant over the entire graph, they do not tell us which ones are really significant. That is because the frequency of an attribute association often does not depart from what we expect and therefore may not be meaningful actually if we already know the distributions of attribute values in the graph. Rather, identifying the statistically significant attribute associations where the pattern of the attribute association deviates from the expected, can potentially infer undiscovered possible relationships between nodes in the graph. The statistical significance of a pattern has been emphasized in various data mining problems [25,36, 41–43] and the previous works already explored why a statistically significant pattern is more important rather than a frequent pattern. Thus, in this chapter we define a statistically significant attribute association and address the problem of uncovering it in attributed graphs.

Fig. 2.1 shows an example that shows a list of possible attribute associations in an attribute graph. An attribute association is *frequent* if the number of pairs of nodes is above a given threshold which is determined by *freq_support*. Unfortunately

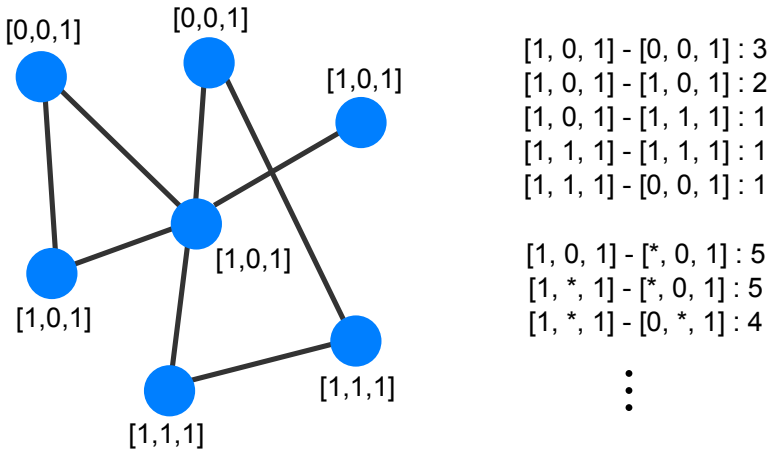


Fig. 2.1.: Attribute associations in attributed graph

the frequency is not sufficient to measure the statistical significance of an attribute association since the frequency eventually depends on the actual distributions of the attribute values in the graph. We will closely see the set difference between the two in Section 2.5. Also when obtaining significant associations, each attribute value does not always have to take discrete attribute value, e.g., 0 or 1 in binary case, as long as the association has enough statistical significance. Accordingly, we introduce wildcard attribute notation ($*$), which matches any value of the corresponding attribute.

The statistical significance of an attribute association with its frequency k is determined by the probability that it is observed at least k times or more, and the probability is called the *p-value* of the attribute association. By measuring *p-value*, we can identify the significant ones even though they are not frequent absolutely in the graph. Also, as shown in Fig. 2.1, we are interested in even associations of partial attribute values as long as they are statistically significant. The main challenge of the problem is how to estimate the probability that an attribute association occurs in a random graph. There are as many different attribute associations as the number of edges in a graph, and if we consider even the partial attribute associations then the number of possible attribute associations grows exponentially. We address the

challenge by transforming a graph G into an alternative graph $\mathcal{AG}$, called *association graph*, where each vertex contains a subset of nodes in G that have the same or similar attribute values and each edge corresponds to a certain attribute association between two set of attribute values, each of which is represented by a cluster. During the process of transformation, we build $\mathcal{AG}$ such that the edges (i.e., associations) are statistically significant.

To experimentally evaluate our work, we use two real world attributed graphs. One is the *DBLP co-authorship network* and the other is the *Yelp social network*. We present the statistically significant attribute associations extracted from the graphs and compare them against the frequent attribute associations qualitatively. In addition to that, we show quantitatively how the statistically significant attribute associations can be used for boosting the performance of the link prediction task.

We summarize the contributions of our work as follows:

- We formally define the novel problem of mining statistically significant attribute associations which aims to find patterns of co-occurring attribute values between nodes which deviate from the expected.
- We design and implement an algorithm that can find the statistically significant attribute associations efficiently and effectively.
- We conduct experiments using real world attributed graphs and show qualitative results as well as the actual application that can benefit from the results.

This chapter is organized as follows. In Section 2.2, we introduce previous works related to our problem and discuss how our problem differs from them. In Section 2.3, we define the problem of mining statistically significant attribute association and provide basic background concepts. The novel algorithm to solve our problem is discussed in Section 2.4 and we present our experimental findings in Section 2.5. Finally, we summarize the chapter in Section 2.6.

2.2 Related Work

There are a number of previous works that have explored the statistical significance of patterns in various data mining and knowledge discovery tasks and have proposed efficient methods for mining the statistically significant patterns. [36, 42] study the statistical significance of subgraphs where the nodes of the graph are labeled. [42] addresses the problem of finding statistically significant connected subgraphs in a vertex-labeled graph where the labels are discrete and continuous. The statistical significance is quantified by using the *chi-square* statistic, which makes the naïve algorithm impractical because of the exponential number of subgraphs. They propose an efficient algorithm which converts the graph into a super-graph. In [36], the authors propose a technique for computing the statistical significance of frequent subgraphs in a graph database. In order to solve the difficulty of estimating the *p-value* of a subgraph directly in the graph space due to the flexible structures of graphs, they tranform graphs into a feature space with predefined set of basis elements, and then approximate the significance of a feature vector in the feature space by using the binomial distribution. Although these two works explore the statistically significant patterns in graphs, they differ from our work in that they more focus on structured patterns, not attribute association patterns.

In addition to graphs, the statistical significance has been studied for other types of patterns as well. [41] extends the traditional association rule mining problem to searching statistically significant association rules such that some spurious rules are not included in the result set while considering statistical dependence. The significance of the observed frequency of an association rule is estimated by the binomial distribution. [43] solves the problem of mining statistically significant substrings in a string generated from a memoryless Bernoulli distribution and uses the chi-square statistic as a quantitative measure of statistical significance. The statistical significance is considered for the sequential pattern mining problem as well in [44]. The approach developed by the authors is able to efficiently mine unexpected patterns in

Table 2.1.: Basic notations

Notation	**Meaning**
$G = (V, E)$	attributed graph
$V = \{u_1, u_2, \ldots, u_{\|V\|}\}$	set of nodes in G
E	set of edges in G
$\mathcal{AG} = (\mathcal{V}, \mathcal{E})$	association graph
$\mathcal{V} = \{c_1, c_2, \ldots, c_{\|\mathcal{V}\|}\}$	set of clusters in $\mathcal{AG}$
$\mathcal{E}$	set of attribute associations in $\mathcal{AG}$
$\mathbf{a} = (a^1, a^2, \ldots, a^l)$	attribute vector of size l
Δ	attribute association
σ	*freq_support*
η	*size_support*
δ_G	density of graph G
Ψ_c	*p-value* of cluster c
$TS(u, v)$	tie-strength between node u and v
$\Gamma(\cdot)$	set of neighbors
$\tilde{G}_c$	subgraph of nodes within cluster c

sequence of itemsets without considering overlapping occurrences or conditioning the length of the sequence.

2.3 Problem Statement

In this section, we give basic definitions of the attribute association, frequent association, statistically significant association, and define the problem of mining statistically significant attribute associations. Table 2.1 introduces the notations we use throughout this chapter.

2.3.1 Attribute Associations

Suppose we have an attributed graph $G = (V, E, A)$ where $V = \{u_1, u_2, \ldots, u_{|V|}\}$ is a set of nodes, $E = V \times V$ is a set of edges, and $A = \{\mathbf{a_{u_1}}, \mathbf{a_{u_2}}, \ldots, \mathbf{a_{u_{|V|}}}\}$ is a set of attribute vectors, each of which is associated with a node in V. The attribute vector $\mathbf{a_u}$ of the node u that holds l different attributes is represented by a vector of l binary values in that each binary indicates whether the node u actually has a value for the corresponding attribute (in case of an m multi-valued attribute, it can be transformed into $m-1$ dichotomous variables each with binary). Then we define an *attribute association* between a pair of attribute vectors $\mathbf{a_1}$ and $\mathbf{a_2}$ as follows:

Definition 2.3.1 *Given two attribute vectors* $\mathbf{a_1} = (a_1^1, a_1^2, \ldots, a_1^l)$ *and* $\mathbf{a_2} = (a_2^1, a_2^2, \ldots, a_2^l)$*, the attribute association between them, denoted by* $\Delta_{\mathbf{a_1},\mathbf{a_2}}$*, is defined as a pair of two sets of attribute values,* $\{i|a_1^i = 1\}$ *and* $\{i|a_2^i = 1\}$ *where* $i \in \{1, 2, \ldots, l\}$.

Note that the attribute association is symmetric with respect to a given pair of attribute vectors $\mathbf{a_1}$ and $\mathbf{a_2}$, that is, $\Delta_{\mathbf{a_1},\mathbf{a_2}} = \Delta_{\mathbf{a_2},\mathbf{a_1}}$. Every pair of nodes has its attribute association and therefore there are as many attribute associations as the number of edges in G. The attribute association information is widely used in many different applications. For example, the link prediction algorithms that aim to predict whether a link will be newly formed between two unconnected nodes in the future usually employ the link structure information around the two nodes but it could leverage from using the attributes of the nodes as well. Many previous researches have shown that nodes in a graph tend to establish homophily or heterophily relationships in terms of their attributes [19–21]. Another example of using attribute information is the community detection problem. Many early approaches to detect latent communities rely on only the link structure of a graph [45–47]. That is, they detect communities such that nodes within the same community interact with each other more frequently than with those outside the community. However more recent studies use the node attributes as well as the link structure and show that the attribute information is helpful for community detection [23, 24, 48].

If an attribute association Δ is repeatedly observed and its frequency is over a given threshold σ that is referred as *freq_support*, then we say Δ is a frequent attribute association.

Definition 2.3.2 *Given an attribute association Δ and a support σ, Δ is called a frequent attribute association if $fr(\Delta) \geq \sigma \times |E|$ where $fr(\Delta)$ is the number of pairs of nodes with Δ.*

When a frequent attribute association is given, we can say that there are many pairs of nodes having the association but it does not necessarily mean that the attribute association is really interesting. For example, in a social network of *Purdue University Almuni*, it is not surprising to observe many connected nodes have the attribute association of {"`Purdue`", "`CS`"} – {"`Purdue`", "`CS`"}. So we are interested in statistically significant attribute associations rather than frequent ones, which will be discussed in the following section.

2.3.2 Statistically Significance

The statistical significance of an object can be quantified by estimating the probability of the observed or rarer objects under the null hypothesis. Let δ_G denote the density of G which is defined as the fraction of the number of edges in G over all pairs of nodes ($\delta_G = \frac{|E|}{1/2 \cdot |V| \cdot (|V|-1)}$). If we randomly select two groups of nodes no matter which attribute values they have, denoted by C_1 and C_2 respectively, then the expected number of edges between C_1 and C_2 is $e(C_1, C_2) = |C_1| \cdot |C_2| \cdot \delta_G$ by assuming the probability of a pair of randomly selected nodes being connected to each other follows δ_G. Also, assuming the edges are independent of each other, the actual number of edges M between C_1 and C_2 would follow the binomial distribution with parameters $n = |C_1| \cdot |C_2|$ and $p = \delta_G$, and thus the probability of getting exactly k edges among n possible edges is given by the following probability mass function:

$$f(k; n, p) = P[M = k] = \binom{n}{k} p^k (1-p)^{n-k} \tag{2.1}$$

If each of C_1 and C_2 is a group of nodes with the same attribute values in G which are specified by an attribute vector, then the attribute vectors $\mathbf{a_1}$ and $\mathbf{a_2}$ can be instantiated from C_1 and C_2 respectively and the attribute association between two attribute vectors is induced from the edges across the nodes of C_1 and the nodes of C_2. So we can measure the statistical significance of a given attribute association $\Delta_{\mathbf{a_1},\mathbf{a_2}}$ based on the probability $P[M \geq k]$ that the observed or higher number of edges occur between C_1 and C_2 in which the nodes have $\mathbf{a_1}$ and $\mathbf{a_2}$ respectively. The association is said to be statistically significant if the estimated probability $P[M \geq k]$ is very small.

$$P[M \geq k] = 1 - \sum_{i=0}^{k-1} \binom{n}{i} p^i (1-p)^{n-i} \tag{2.2}$$

Definition 2.3.3 *An attribute association $\Delta_{\mathbf{a_1},\mathbf{a_2}}$ between C_1 and C_2 is statistically significant if the probability of the observed or more number of edges between C_1 and C_2 is less than α which is called a significance level.*

In order to show the assumption that the number of edges between two groups of nodes follows the binomial distribution is reasonable, we randomly sampled two groups of 50 nodes from the *DBLP co-authorship network* (the details of the network is described in Section 2.5) 10,000 times and obtained the empirical distribution of the number of edges residing between the two groups. As shown in Fig. 2.2, the empirical distribution (blue bar, mean: 8.68 / stddev: 3.29) is very closed to the actual binomial distribution (red line, mean: 8.64 / stddev: 2.93), which is verified by the chi-squared testing on the two distributions.

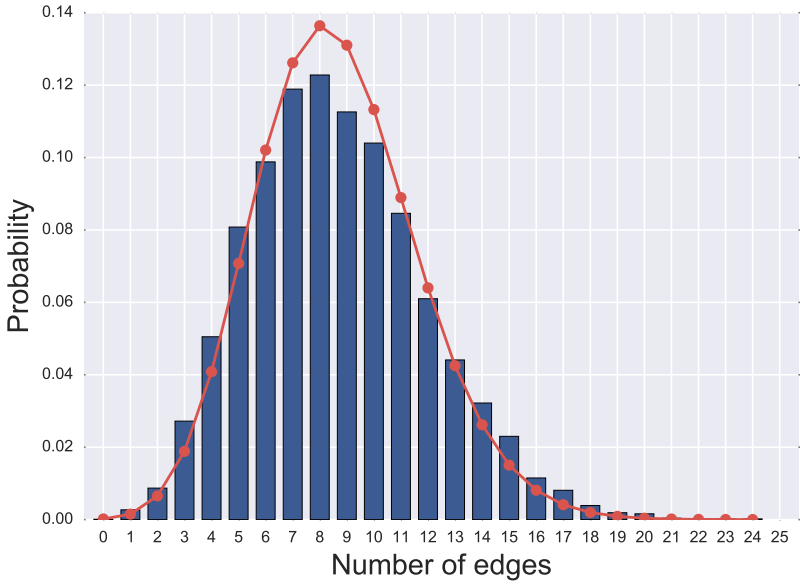


Fig. 2.2.: Distribution of the number of edges between two groups of nodes

2.3.3 Locality Preserving Significant Associations

An attribute association may reside in anywhere over the entire graph G. However, we expect that a certain attribute association could be observed more frequently among nodes which are closed to each other. For example, in the *DBLP co-authorship network*, some authors who have published papers in venues of data mining area are expected to have a certain attribute association with other authors in the same or similar area (e.g., the association of {`ICDM`, `KDD`} - {`ICDM`, `NIPS`, `ICML`}). Any pair of authors in a relationship with the association could be seen in several locations of G, but some of them may be located very closely in terms of the hop distance in the graph and form a densely connected subgraph or community. Different communties that have the same venue pattern many times would be corresponding to different schools in different countries. That is, some attribute association patterns come with locality in the graph and such a pattern can be more statistically significant locally rather than globally. Besides, some attribute association patterns that are statistically

significant locally may form another complex patterns (e.g., star or chain, not just pair) among them. One of the nice features of the algorithm we propose in Section 2.4 is that it is able to effectively find all the statistically significant attribute associations while preserving the locality.

2.4 Graph Transformation

In this section, we describe the algorithm that finds statistically significant attribute associations in a given attribute graph G. The basic approach for finding statistically significant attribute associations is to transform the original graph G into a new graph $\mathcal{AG} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$, which is called *Association Graph*, where each node in $\mathcal{V}$ corresponds to a group of nodes in V which have the same or similar attribute values, each edge in $\mathcal{E}$ is an attribute association Δ between two attribute vectors, and each attribute vector in $\mathcal{A}$ represents one shared by a group of nodes in V. To avoid confusion, from now on we call a node in $\mathcal{V}$ a cluster and call an edge in $\mathcal{E}$ an association. Each association Δ is assigned a weight, referred as its strength $w(\Delta)$, that is given by the number of edges between nodes in the clusters forming the association. For a given association Δ and its associated strength, defined as the number of edges between nodes in the clusters, we can determine whether Δ is significant or not by looking at the strength and the size of the clusters to which Δ is incident, which will be explained in detail in the following sections.

The graph transformation can be done through an iteration of two steps. We first start with a single cluster that contains all nodes of V in G and then the cluster is partitioned into several subclusters by applying two steps repeatedly and iteratively. For the first step, a cluster is split such that each subcluster contains a subset of V that have similar attribute values. This operation is able to be easily done using any clustering algorithms. In case of binary attributes, we just select one of the attributes and then do two-way split with respect to the attribute. In Section 2.4.1, we explain how to select the attribute. For the second step, we try to split a cluster such that

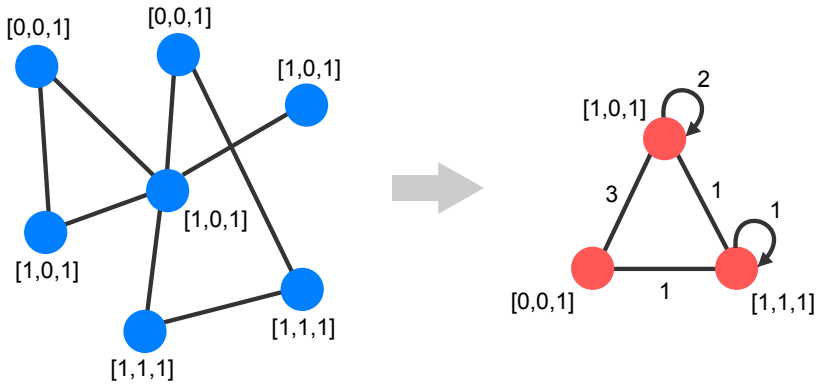


Fig. 2.3.: Graph transformation

each of the associations incident to the cluster has higher strength in order to obtain more significant associations between two sets of attributes. That is, the iteration of the two different splits alternate between performing the similarity-based split, which produces clusters with the same or similar attribute values, and the strength-based split, which makes associations more significant. It results in a new graph $\mathcal{AG}$ where we can see groups of nodes with certain attribute values and significant associations between them, as shown in Fig. 2.3.

Algorithm 1 shows the whole structure of the graph tranformation algorithm including the two steps of splits. and the following subsections describe how each split should be done in detail.

2.4.1 Similarity-based split

As mentioned already, the goal of the first step is to maximize the similarity among attribute values in each cluster so that each cluster can represent a certain set of attribute values. Thus we select one of the clusters in $\mathcal{AG}$ and then split it into two subclusters based on a certain attribute so that each subcluster contains a set of node that share the same value on the attribute. The way to select a cluster in $\mathcal{AG}$ is based on the following idea. Basically we do not only want to maximize the similarity

Algorithm 1 Algorithm for graph transformation

Input: $G = (V, E, A)$

Output: $\mathcal{AG} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$

Initialization :

1: $\mathcal{V} = \emptyset, \mathcal{E} = \emptyset$

2: c is initialized as a cluster containing all nodes in V

3: $\mathcal{V} = \mathcal{V} \cup c$

Iterative Process

4: **while** there exist at least one cluster to split **do**

5: $c =$ *findClusterForSimilaritySplit*($\mathcal{AG}$)

6: **if** (c exists) **then**

7: similaritySplit($\mathcal{AG}$, c)

8: **end if**

9: $c =$ *findClusterForStrengthSplit*($\mathcal{AG}$)

10: **if** (c exists) **then**

11: strengthSplit($\mathcal{AG}$, c)

12: **end if**

13: **end while**

14: **return** $\mathcal{AG}$

of attribute values in each subcluster after the split, but also want each subcluster to be statistically significant as much as possible in terms of the attribute values of its nodes.

To achieve the goal, we need to figure out which cluster should be split and which attribute should be used to split the cluster. Let p_i denote the probability that a value of 1 occurs at i-th attribute, which is the fraction of nodes with a value of 1 for the i-th attribute in G. So, p_i is considered an expectation of having the attribute value for a random node. First, an attribute on which a cluster should be split based is picked such that the probability of the attribute having the value of 1 in the cluster

is least deviated from its corresponding p_i. It allows the subclusters to not only have higher similar attribute values among the nodes in them but also have the highest significance gain through the split. Once we decide which attribute should be used for the split of the clusters, we select one of the clusters to split. While assuming that the attributes are independent of each other and the number of times the value of 1 appears at the i-th attribute follows the binomial distribution with the probability p_i, the statistical significance Ψ_c of a cluster c is defined based on the product of p-values of the attribute values of the nodes in the cluster as follows,

$$\Psi_c = 1 - \prod_{i=1}^{l} \left(1 - \sum_{j=0}^{k_i - 1} \binom{|c|}{j} p_i^j (1 - p_i)^{|c|-j}\right) \tag{2.3}$$

where k_i is the number of nodes having the value of 1 on the i-th attribute and $|c|$ is the number of nodes in the cluster c. So for each cluster c we compute $\Psi_{c'}$ of the subclusters c'. Remind that our goal is to split a cluster so that its subclusters are most statistically significant. However, since the subclusters may have different significances (one can be highly significant but the others can be very low), we take subclusters with the lowest significance from each of the clusters in $\mathcal{AG}$ and then select a cluster that will produce a subcluster with the highest significance among those subclusters, i.e.,

$$\arg\max_{c} \left(\min_{c' \in sb(c)} \Psi_{c'} \right) \tag{2.4}$$

where $sb(c)$ is a set of subclusters that will be created after the split. In this way, we can avoid to split a cluster that will produce the least significant subclusters. By repeating this kind of split, $\mathcal{AG}$ will have only clusters, in each of which the same attribute values are shared by its nodes, but we need to place one constraint while doing the split. Even though a cluster represents a certain set of attribute values shared in it, if it contains only a few nodes then its attribute values may not be meaningful at all when we look at an attribute association between clusters in $\mathcal{AG}$. Thus, we use *size_support*, denoted by η, to force a cluster not to split any more if

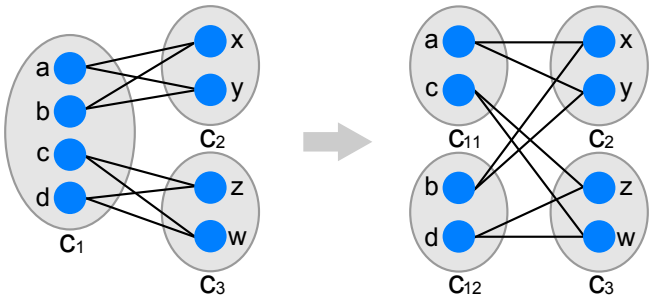
all the subclusters that will be obtained after splitting the cluster have the sizes less than $\eta \cdot |V|$. Thus, during the first step, we examine only clusters satisfying the η threshold to determine which cluster should be split. Also, it is obvious that a cluster in which all its nodes have the same attribute values does not need to be split.

We do not only want nodes in the same cluster to have the same attribute values but also allow them to have similar attribute values. In other words, even though every node in a cluster does not agree on a certain attribute, if the distribution of the values of the attribute is statistically significantly deviated from the expectation, then those nodes are considered to have an identical value for the attribute.

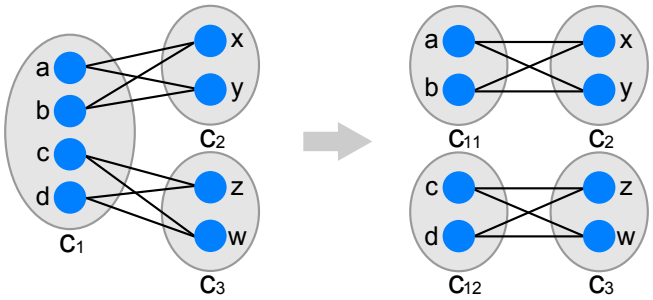
Once a cluster is split at the first step, we move on to the second step to increase the significances of the attribute associations between clusters.

2.4.2 Strength-based split

While the similarity-based split of the first step aims to increase the similarity of attribute values for a cluster, we try to maximize strengths of associations to which a cluster is incident through the strength-based split. Given an attribute association between two clusters, its strength is defined as the number of edges that connect the nodes of the clusters. The strength is not meaningful by itself because the significance depends on the sizes of the clusters as well as the strength. As we discussed the definition of a statistically significant attribute association in Section 2.3.2, the stronger strength an attribute association has and the smaller the associated clusters are, the higher statistically significant the association is. Thus, in order to make an association more significant, a cluster that is one of the end points of the association needs to be split into subclusters such that nodes which have many common neighbor clusters belong to the same subcluster. Fig. 2.4 illustrates the basic idea of the strength-based split. Suppose we want to maximize the significance of the associations held by the cluster c_1 and we consider two different splits to do that as presented in Fig. 2.4(a) and Fig. 2.4(b). The nodes a and b in c_1 have edges, all of which are incident to other



(a) (a, c) and (b, d)



(b) (a, b) and (c, d)

Fig. 2.4.: Two different strength-based splits

nodes in c_2 while the nodes c and d are adjacent to only other nodes in c_3. Thus, in order for the subclusters obtained from splitting c_1 to have associations of maximized significance, the split should produce two subclusters which contain the two nodes a and b, and the other two nodes c and d, respectively.

So we need to find the optimal split of a cluster so that its associations become more significant. For a given cluster c we try to split, we build a graph $\tilde{G} = (\tilde{V}, \tilde{E})$ where $\tilde{V} = \{u | u \in c\}$ and $\tilde{E} = \{(u, v) | u, v \in c \wedge \exists c' \text{ s.t. } (u, w_1), (u, w_2) \in E \text{ and } w_1, w_2 \in c'\}$, some of which are connected to each other if they have edges with some common neighbor clusters, $\Gamma(c)$. Those edges in $\tilde{E}$ are weighted based on the fraction of edges to common neighbors among all of their edges. Then, we partition the graph $\tilde{G}$ based on the weights of the edges in the graph and the subgraphs resulted from the partition become the subclusters we obtain through the

strength-based split. For this task, we need to come up with a proper way to assign weights to the edges. We borrow the idea of tie-strength between individuals in social network. In the social science community, there are many different ways to define the tie-strength of an interpersonal relationship [49], and one widely used measure is the Jaccard index. That is, a tie-strength between two individuals u and v is determined by $|\Gamma(u) \cap \Gamma(v)|/|\Gamma(u) \cup \Gamma(v)|$ where $\Gamma(\cdot)$ is a set of neighbors of a node. In our setting, two nodes u and v in the cluster c may not have common neighbor nodes in G but some of their neighbor nodes may belong to the same neighbor cluster c' in $\mathcal{AG}$. Similarly, when u and v in c are connected to some of the nodes in a common neighbor cluster c' of c, there might not be common nodes in c' which are incident to both u and v. Thus, we modify the Jaccard index slightly so as to measure the tie-strength between u and v while capturing the common neighbor clusters.

$$TS(u,v) = \frac{\sum_{c' \in \Gamma(c)} \min\{\phi(u,c'), \phi(v,c')\}}{\sum_{c' \in \Gamma(c)} \max\{\phi(u,c'), \phi(v,c')\}} \tag{2.5}$$

where $\phi(u,c') = |w|w \in c \wedge (u,v) \in E|$, that is the number of edges in E between u and any nodes in c'. Using this tie-strength measure, we can have nodes belong to the same subcluster after the split if they have many common neighbor clusters, regardless of whether they have common neighbor nodes in G or not (of course, it depends on the weight given by $TS(\cdot,\cdot)$).

Once we have $\tilde{G}$ for the cluster c then we perform graph partitioning on $\tilde{G}$ to find optimal subclusters that can make the associations between c and $c' \in \Gamma(c)$ more significant. Since all the edges $\tilde{E}$ of $\tilde{G}$ are assigned weights and $\tilde{G}$ should be partitioned based on the weights, we take an approach to maximize the modularity of $\tilde{G}$ [47]. The modularity $Q(\tilde{G})$ is defined as

$$Q(\tilde{G}) = \frac{1}{2m} \sum_{u,v} \Big[A_{uv} - \frac{k_u, k_v}{2m} \Big] \delta(c_u, c_v) \tag{2.6}$$

where $m = \tilde{E}$, k_u is the degree of u, c_u is the group to which u belongs, and A_{uv} is 1 if there is an edge in $\tilde{E}$ between u and v otherwise 0. That is, the modularity is

the fraction of the edges that fall within the given groups minus the expected such fraction if edges were distributed at random. If we split the cluster c through the graph partitioning method as described, a set of nodes that share many common neighbor clusters is likely to fall within the same subcluster as much as possible, and different nodes that share only few common neighbors would be distributed to different subclusters. Thus, we can increase the statistical significances of the attribute associations.

During the second step, we enforce a couple of conditions to prune some clusters and associations in $\mathcal{AG}$ and do not perform the strength-based split on them for both achieving computational efficiency and finding more meaningful results. As done in the first step, we use *size_support*, η because if the size of a cluster c is too small, we do not believe that c is representative of a certain set of attribute values. Thus, the strength-based split is run for a cluster c only when $|c| \geq \eta \cdot |V|$. In addition to that, if a cluster has an attribute association with too weak strength, then we can safely discard it for the rest of the algorithm. Note that the strength of an attribute association between two clusters monotonically decreases as the two splits are performed iteratively while the statistically significance is not monotonic in either way. Since we consider only attribute associations between clusters satisfying the *size_support* condition and the statistically significance of an association depends on its strength and the sizes of the clusters at the end points, we can prune an attribute association from $\mathcal{AG}$ as long as it meets the following condition.

Lemma 1 *Given an attribute association Δ_{c_1,c_2} and its two incident clusters c_1 and c_2, if the strength of Δ_{c_1,c_2} is less than $\Phi^{-1}\left(1-\alpha-\frac{C(p^2+q^2)}{\sqrt{npq}}\right)\sqrt{npq}+np$, then Δ_{c_1,c_2} does not have a chance to be statistically significant any more, where $n = |c_1| \cdot |c_2|$, $p = \delta_G$, $q = 1-p$, $\Phi(\cdot)$ is the error function, and C is a constant.*

Proof Given the size_support η, both the clusters c_1 and c_2 should have the size of at least $|V| \cdot \eta$ in order to make the attribute association Δ_{c_1,c_2} considered as statistically significant. Also, let k denote the strength of Δ_{c_1,c_2} and then according

to the (2.2), $P[X \geq k] \leq \alpha$. If we approximate the binomial distribution using the normal distribution,

$$\begin{aligned} P[X \geq k] &= P\big[\frac{X - np}{\sqrt{npq}} \geq \frac{k - np}{\sqrt{npq}}\big] \\ &= P\big[Z \geq \frac{k - np}{\sqrt{npq}}\big] \leq \alpha \end{aligned} \tag{2.7}$$

Now we have the standard normal distribution and need to find the lower bound of k which satisfies the inequality (2.7). Using the error function $\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{\frac{-t^2}{2}}\, \mathrm{d}t$ which is essentially identical to the standard normal cumulative distribution function [50],

$$\begin{aligned} \frac{k - np}{\sqrt{npq}} &\geq \Phi^{-1}(1 - \alpha) \\ k &\geq \Phi^{-1}(1 - \alpha)\sqrt{npq} + np \end{aligned} \tag{2.8}$$

The lower bound for k is originated from approximation based on the standard normal distribution and thus we need to get the error bound. According to the following *Berry-Essen theorem* [51],

$$\sup_{x \in \mathbb{R}} \Big| P\Big[\frac{B(p,n) - np}{\sqrt{npq}} - \Phi(x)\Big]\Big| \leq \frac{C(p^2 + q^2)}{\sqrt{npq}} \tag{2.9}$$

with $C < 0.4748$, we know that the error arising from the approximation is at most $\frac{C(p^2+q^2)}{\sqrt{npq}}$. As a result, if we relax the lower bound for k in (2.8) to the extent of the error, then we obtain

$$k \geq \Phi^{-1}\Big(1 - \alpha - \frac{C(p^2 + q^2)}{\sqrt{npq}}\Big)\sqrt{npq} + np \tag{2.10}$$

■

Since p is very small and n is large for given η, the error bound is small and we can still get a reasonably tight lower bound for k. Regarding the inverse of the error function, if we use $\alpha = 0.01$ as the significance level, then $\Phi^{-1}(1 - \alpha) = 1.8212$. According to Lemma 1, we drop attribute associations if they are too weak to be able to be significant later on. In fact, such associations are noise and do not bring us any meanings. Rather, it prevents the strength-based split from running optimally.

Table 2.2.: Dataset statistics

	Original graph			Association graph	
	Nodes	Edges	Density	Nodes	Edges
DBLP	4,672	37,726	0.00346	195	6,302
Yelp	4,454	44,906	0.00453	202	8,388

Table 2.3.: DBLP subareas in computer science

Subarea	Conferences
DM/ML	ICDM, NIPS, ICML
OS	SOSP, OSDI
Theory	FOCS, STOC, SODA
Security	IEEE Symposium on Security and Privacy (S&P), ACM Conference on Computer and Communications Security (CCS)

2.5 Experiments

2.5.1 Datasets

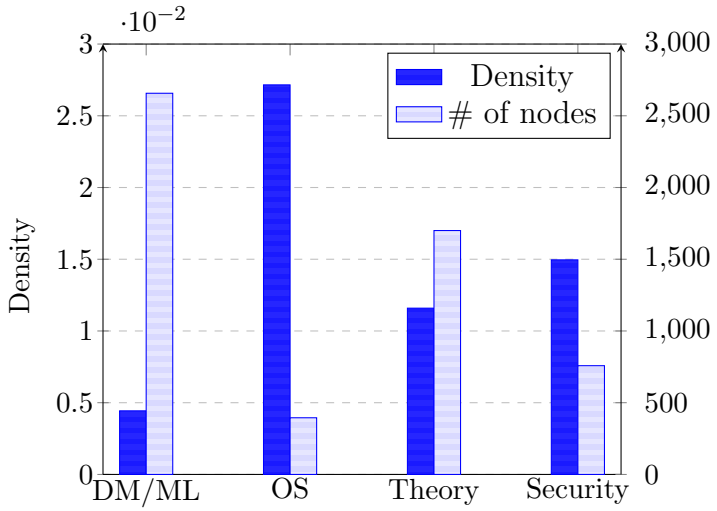
We ran the graph transformation algorithm on real co-authorship and social networks, and obtained the resulting association graphs. Using the association graphs, we analyzed qualitative differences between the statistically significant and frequent associations. Also we showed the application of the significant patterns to a link prediction problem, and synthetic graphs with attributes are considered to show the algorithm's scalability.

DBLP. We obtained a collection of bibliographic information from the DBLP website [52], an open bibliographic information provider of computer science journals and conferences. Each record of journal or conference paper has one or more authors, and the venue, on which it is published. We first filtered out any authors who appear in less than 3 papers. Then, we considered only papers published to the 10 conferences of 4 different subareas of computer science, i.e., data mining and machine learning (DM/ML), operating systems (OS), theory, and security. More details are shown in Table 2.3. Then we built an attribute vector of length 10 for each node, i.e., if an author (or a node) published a paper to a conference in Table 2.3, we set the corresponding vector value to 1. If not, we set the corresponding vector value to 0. Finally, an edge is formed if two authors (or nodes) have co-authored at least one paper in the dataset.

Yelp. Yelp is a provider of crowd-sourced reviews about local businesses, along with a social network. The Yelp challenge dataset [53] contains the social network, composed of the users (nodes) and their friend relations (edges). Also the sets of users' reviews are provided in the dataset. Each review is tied to a user and a business, and each business has a small set of business type categories. We first filtered out any users who have less than 10 reviews. Then, we considered only reviews for the restaurants, which has at least one of the 10 business categories, {Chinese, Japanese, Mediterranean, Thai, French, Greek, Vietnamese, Korean, Indian, British}. The node attributes are compiled similarly to the DBLP dataset. Note that we did not use some of the most popular restaurant categories, e.g., American, Mexican, and Italian. As the majority of users has left reviews on the restaurants of such categories, they seem to appear in most of the attribute associations and carry little or no information.

2.5.2 Effectiveness Analysis

To evaluate the effectiveness of our algorithm, we conducted the set difference between the statistically significant associations and the top-15 frequent associations.



Subarea	Nodes	Edges	Density
DM/ML	2,657	15,639	0.00443
OS	395	2,113	0.02715
Theory	1,700	16,738	0.01159
Security	758	4,291	0.01496

Fig. 2.5.: DBLP subgraph characteristics for different subareas

As the resulting significant associations contain wildcard attributes, it is not easy to make direct comparisons or set differences between the two. Thus, we took a conservative approach that as long as all attribute values of any top-15 frequent associations have exact or wildcard attribute match, we considered that there is a match. This approach is certainly in favor of the frequent associations, since it ignores that wildcard matches may lead to some other possible set of attribute values.

Table 2.4, we have the set difference between statistically significant and frequent associations for the DBLP dataset. First consider 4 subgraphs that only contains the nodes and their edges, whose attribute value for any conference of the corresponding subarea is 1. Fig. 2.5 describes the characteristics of each subgraph. The subgraph of DM/ML has a large number of nodes but its graph density is small, which means that

Table 2.4.: Significant associations minus Frequent associations for DBLP

#	Association
1	{SOSP, OSDI, S&P, CCS} – {SOSP, OSDI}
2	{SOSP, OSDI, S&P, CCS} – {S&P, CCS}
3	{ICML, ICDM, S&P(*)} – {ICML, ICDM}
4	{SOSP, OSDI, S&P, CCS} – {SOSP, S&P, CCS}
5	{FOCS(*), STOC(*), CCS} – {S&P, CCS}
6	{ICML, ICDM, S&P(*)} – {ICML(*), ICDM}
7	{SOSP, S&P, CCS} – {SOSP, OSDI}
8	{SOSP, S&P, CCS} – {S&P, CCS}
9	{ICML, ICDM, S&P(*)} – {ICDM, OSDI(*)}
10	{ICML, ICDM} – {ICML(*), ICDM}

the tie-strengths are weak. On the other hand, there are relatively small numbers of nodes in the subgraph of OS and security but their densities are high, which means that the tie-strengths are strong among the nodes. We can easily identify the OS and security-related associations, which contain {SOSP, OSDI} and {S&P, CCS}, are appearing on top of the difference list. Also note that many frequent associations are related to DM/ML conferences since its subgraph contains the most number of edges while its density is low.

From Table 2.4, we can infer many interesting significant associations, which do not appear in the frequent association list. The association number 1, 2, 4, 7, and 8 clearly shows that the nodes who have authorship in the OS-related conferences tend to co-work with the authors in the security-related conferences. The association number 3, 5 and 6 shows that the nodes who have authorship in the security-related conferences frequently co-work with the authors in DM/ML and theory-related conferences. Interestingly enough, the association number 9 shows how the authors in

DM/ML, security, and OS have frequent co-authorship relations in the graph. These results might look obvious to some of the readers who have a good understanding of co-authorship in computer science. However, when the relationships of attributes are little known, the discussed results may be intriguing.

Table 2.5 shows that the set difference between statistically significant and frequent associations for the Yelp dataset. Note that {Chinese, Japanese} appears very commonly in the association results due to their prevalence in node attributes. Thus, we will exclude them from the subsequent discussions. Also it turned out that the first 10 significant associations with the highest statistical significance are the same as the associations reported in Table 2.5. That is, none of the first 10 significant associations are reported in the top-15 frequent association results, since the significant associations do not occur often in terms of frequency but do occur often in the dataset in a statistically significant manner.

Among the frequent visitors of {Mediterranean, Thai}, the association number 2 and 7 shows that the nodes with {Greek} attribute are strongly associated with the nodes with {Vietnamese, Korean} attributes, and the association number 4, 6 and 10 shows that the nodes with {Vietnamese, Korean, Indian} are strongly associated with the nodes with {Vietnamese, Korean} and {Greek} attributes. Also the association number 5 and 8 describes that the nodes with {Mediterranean} have statistically significant associations with the nodes with {Mediterranean, Thai, Greek}.

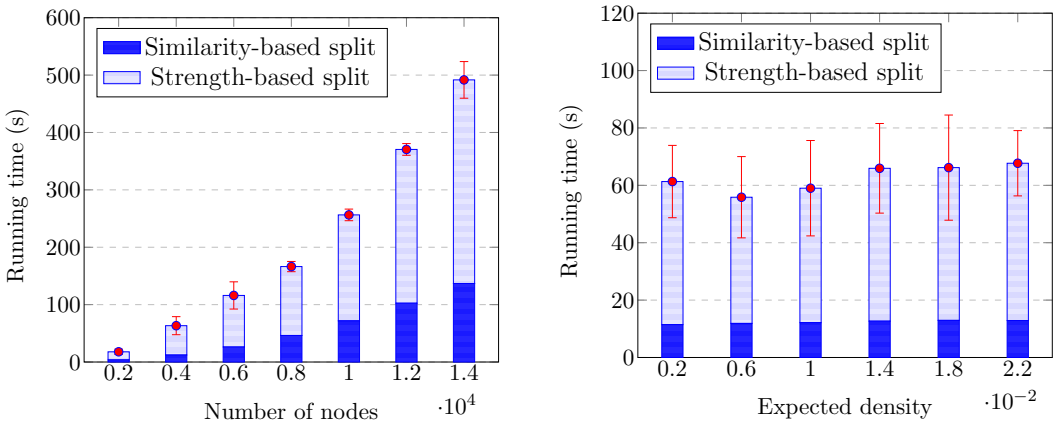
2.5.3 Scalability Analysis

We evaluated the computation cost of our algorithm on synthetic attributed graphs of different sizes and densities. The experiments were carried on a machine with an Intel Xeon 3.1GHz CPU and 32GB memory, running 64bit Ubuntu 14.04. All algorithms are implemented in Python 2.7.

The graphs are generated based on the simplified version of Multiplicative Attribute Graph (MAG) model [19]. MAG is widely used in the literature to generate

Table 2.5.: Significant associations minus Frequent associations for Yelp

#	Association
1	{Chinese, Japanese, Mediterranean, Thai, Greek} – {Chinese, Mediterranean, Thai(*), Greek}
2	{Chinese, Japanese, Mediterranean, Thai, Vietnamese, Korean} – {Chinese, Japanese, Mediterranean, Thai, Greek}
3	{Chinese, Japanese, Thai, Vietnamese, Korean} – {Chinese, Japanese, Vietnamese, Korean}
4	{Chinese, Japanese, Mediterranean, Thai, Vietnamese, Korean, Indian} – {Chinese, Japanese, Mediterranean, Thai, Vietnamese, Korean}
5	{Chinese, Mediterranean, Thai(*), Greek} – {Chinese, Mediterranean}
6	{Chinese, Japanese, Mediterranean, Thai, Vietnamese, Korean, Indian} – {Chinese, Japanese, Thai}
7	{Chinese, Japanese, Mediterranean, Thai, Vietnamese, Korean} – {Chinese, Mediterranean, Thai(*), Greek}
8	{Chinese, Japanese, Mediterranean, Thai, Greek} – {Chinese, Mediterranean}
9	{Chinese, Japanese, Thai, Vietnamese, Korean} – {Chinese, Japanese, Thai}
10	{Chinese, Japanese, Mediterranean, Thai, Vietnamese, Korean, Indian} – {Chinese, Japanese, Mediterranean, Thai, Greek}



(a) $\mu = 0.6$, density$= 0.010$ (b) $\mu = 0.6$, nodes$= 4,000$

Fig. 2.6.: Running time experiments on synthetic graph datasets

synthetic graphs with node attributes, and known to model real-world networks with flexibility. We conducted two sets of experiments with $l = 5$ and μ's fixed, the probability of each attribute value being 1, i.e., each node has five binary attributes and the attributes are drawn from the same distribution, retaining the node attribute distribution throughout the experiments.

Time complexity. Our algorithm is divisive in nature and it splits at least one node of *Association Graph* in every iteration. First, the similarity-based split step will run $\mathcal{O}(2^l)$ iterations. Usually the length of attribute vector is small, $l \ll n$, and the similarity-based split under reasonable settings takes much less time compared to that of the strength-based split. In the strength-based split step, it is not hard to see that the computation of tie-strengths between each pair of nodes, $\mathcal{O}(n^2)$, dominates the running time of the step. And we can notice that the algorithm will run $\log n$ iterations of the strength-based steps on average. Accordingly the overall average time complexity of the algorithm is $\mathcal{O}(n^2 \log n)$.

Results. Fig. 2.6(a) shows the computation time over the number of nodes. We fixed the attribute link-affinity matrix [19], which determines the probability of edge formation between two sets of node attributes. Note that since we kept all parameters

of the MAG model but the number of nodes, the graph density remained the same. We confirmed that the algorithm is of polynomial time in the number of nodes. This result is in line with the time complexity we discussed above.

In the second experiment, we fixed the number of nodes and the scale factor of the attribute matrix, which merely changes the expected number of edges. That is, we scaled the attribute matrix such that the resulting graphs have the graph densities as we desire, without changing any other properties of the graphs. In Fig. 2.6(b), we can easily observe that the algorithm's running time remains almost the same as we increase the expected graph density. The aforementioned time complexity should well explain the result.

Finally, both of the plots in Fig. 2.6 show that the running time of the strength-based split step dominates that of the similarity-based step. Also both plots describe that the running time of the similarity-based step remain the same as we add more edges with the number of nodes fixed, and the running time grows as we increase the number of nodes. This supports our intuition that the similarity-based split step is not relevant to the number of edges or graph density.

2.5.4 Application: Link prediction

As one of the application for which the statistically significant attribute associations are useful, the *link prediction* problem is considered. Many different approaches to the link prediction have been proposed for the past decade, but with the objective of showing the potential merit of the statistically significance attribute associations, we simply use the Jaccard coefficient proposed in [54] and compare the effects of using statistically significant attribute associations and frequent ones. Given a pair of nodes without an edge, we compute the prediction score by combining the Jaccard coefficient $J(u, v)$ and the score $S(u, v)$ resulted from either the significance or the normalized frequency of an attribute association between the nodes as follows

$$pred(u, v) = \tau \cdot J(u, v) + (1 - \tau) \cdot S(u, v) \tag{2.11}$$

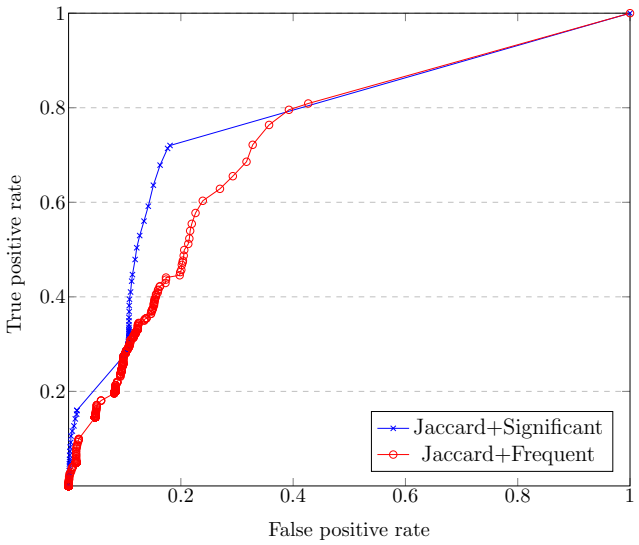


Fig. 2.7.: Link prediction performance

and if it is over a given threshold then we predict that u and v will form a new link. We take two snapshots of the *DBLP co-authorship network* (Mar 2015 and Mar 2016) and all the newly created links between the two snapshots are used for the positive samples. Similarly, a set of pairs of nodes that do not have an edge in both the snapshots are used for the negative samples. Since the number of negative samples far outweighs the number of positive samples, we do negative subsampling with the ratio of $1:5$ (five negatives per one positive). In Fig. 2.7, we report the ROC curves for two different methods, `Jaccard+Significant`, and `Jaccard+Frequent`. As shown in Fig. 2.7, the link prediction can more benefit from employing the attribute information and the statistically significant attribute associations can achieve higher performance rather than the frequent ones.

2.6 Summary

We defined a problem of mining statistically significant attribute associations using *Association Graph*, which keeps the locality of attribute associations and carries the significant relationships between the sets of attribute values. And we proposed a

novel, two-step iterative algorithm that efficiently and effectively generates an *Association Graph* from the original graph. The experiments are conducted on two real world datasets, and we ran some qualitative analysis on the results, confirming that our algorithm effectively finds the significant associations, which cannot be uncovered by conventional frequent association mining. Also we ran extensive scalability experiments on synthetic datasets, and confirmed that the algorithm is of polynomial running time in the number of nodes. Lastly, applying the results from one of the real world datasets to the link prediction task, and we showed how the statistically significant attribute associations can be used in practice.

For future work, we plan to investigate how we can exploit resulting *Association Graph* better, e.g., its locality preserving property, and if we can come up with a linear time algorithm or a distributed algorithm, which can be run on large-scale graphs.

3. COMMUNITY DETECTION

As many different kinds of complexe network data have become more and more available in these days, the community detection problem has become more important and its applications have been prevalent. Numerous methods have been proposed for solving the problem. One class of them considers only the network structure to detect latent communities while the other class of methods uses both the network structure and node attributes, if available. In this chapter, we propose a new community detection method, Probabilistic Topic-aware Community (PTC), based on a probabilistic generative model. The model tries to model an observed network along with node attributes by using the notion of hidden topics and individual node's bias. We provide the model description and the inference strategy for the hidden parameters in the model. The experiments with multiple Facebook ego networks show us not only the effectiveness of the proposed model but also the general limitation of using node attributes for community detection at the same time.

3.1 Introduction

The problem of community detection in networks has attracted a lot of interests from researchers in the literature for the past decades because the communities are helpful to better understand the underlying structures of the network and see how nodes form connections in an organized way. A community simply can be defined as a group of nodes which share common properties and/or interact with each other more frequently than with those outside the group. Communities can be observed easily in many different kinds of real-world networks such as social networks, collaboration networks, protein-protein interaction networks, and so on [55].

According to the definition of a community we saw earlier, we expect that there are many connections between nodes in a community and few connections between nodes across different communities. We can also consider the attributes of nodes if available. It is reasonable to think that a community is associated with some attributes and the nodes in the same community might have the same value on the attributes. For example, in social networks such as Facebook and Twitter, the members of the community `Purdue alumni` should have the value of "purdue" on the attribute *school* while the other community `Google employees` is strongly associated with the value of "google" on the attribute *work* but its members may have diverse values on the attribute *school*. Thus, we could consider two different sources that could be affected by communities: the network structure and the node attributes.

Due to the importance of the problem and a lot of possible applications, there are many methods that have been proposed for solving the community detection problem. Some of them have considered only either the network structure [46, 47, 56–58] or the node attributes [59]. However, both of the two sources are jointly correlated to the formation of communities [22]. That is, it is empirically shown that nodes across communities are loosely connected and nodes in a community share the same value with each other for certain attributes. Recently, many different methods that consider both the network structure and node attributes have been proposed [23, 24, 26, 27, 60–62].

There are many different ways to model the link structure in a network and the node attributes. We follow an idea of generating word-document corpus in LDA (Latent Dirichlet Allocation) model [63]. That is, we assume that every latent community has its own hidden topic distributions. Then each attribute is associated with probability distributions, each of which is associated with a topic, and the value of the attribute is decided according to a topic assigned to the attribute and its corresponding value distribution. In this way, we can model the attribute values such that two nodes in the same community are likely to have the same attribute value.

One interesting fact is that every attribute is not relevant to a community and it even depends on individual nodes. There can be some attributes whose values are not generated from a community's topics. Especially, if a node is loosely connected to other nodes so it does not belong to any community, then the attribute values possessed by the node should come from the node's personal topics. That is, each node has its own bias that represents how likely it takes attribute values from communities' topics or its own personal topics. So we aim to model topic-aware communities and each individual node's personal topics. Eventually the *Probabilistic Topic-aware Community* (PTC) model proposed in this chapter differs from existing works in that we consider hidden topics behind communties and nodes and non-community related attributes to generate the node attributes.

Our contributions of this chapter consist of

- For the problem of community detection, we propose a new probabilistic generative model which incorporates hidden topics for generating attribute values and individual biases for differentiating nodes' personal topics from communities' topics.

- We develop an inference algorithm based on the Gibbs sampling and the gradient method, which is used to estimate the hidden parameters in the proposed model.

- We apply the model to discover latent communities and show its effectiveness on real-world social networks through experimental results.

This chapter is organized as follows: In Section 3.2, we introduce some previous works that are related to the community detection problem or statistical network model. We propose a new probabilistic generative model for discovering communities in Section 3.3 and explain how the inference steps should be defined in Section 3.4. In Section 3.5, we conduct an experiment with real-world social network datasets and report the performances of our model and other baseline models. Lastly, we summarize the chapter and discuss the future work in Section 3.6.

3.2 Related Work

Many different kinds of approaches have been proposed to solve the problem of community detection. A probabilistic generative model can be used to find latent communities over a network [26, 27]. They basically assume that communities might exist initially with having nodes as their members and the node attributes and connections are generated according to the community memberships. So they try to jointly model the node attributes and connections which are observed in a network. [27] uses a logistic function based on the community memberships to model the node attributes and link-affinity matrices that represent how likely a pair of nodes has a link depending on the community memberships to model the occurrences of links. [26] devises a model that has pretty similar structure to [27] except that it tries to model the connections by using a logistic function with the community memberships as well. Our model differs from them in that we consider non-community related attributes and hidden topics behind communities and nodes to generate the node attributes.

The community detection could be considered as an application of the problem of graph clustering. The graph clustering divides a graph into groups of similar nodes based on a predefined objective function. [28–30] propose distance-based solutions to compute the similarity between nodes while taking into account both network structure and nodes attributes. Even though the methods in [28–30] are able to produce well-clustered nodes for a given attributed graph, they do not consider overlapping between communities so all nodes in a graph can belong to only one cluster.

3.3 Model Description

In this section, we describe a new probabilistic generative model that models how the network structure and node attributes are generated from latent communities behind nodes. The *Probabilistic Topic-aware Community* (PTC) model is built based on an idea of jointly modeling the links of the network and the node attributes while following the assumption that each community and each individual are associated

with hidden topic distributions and follow the distributions to draw node attribute values.

3.3.1 Model Overview

Given an attributed network G, we assume that there are U nodes in G and each of them is associated with a set of A attributes. Each node $u = 1, 2, \ldots, U$ has a non-negative real-valued community memberships π_u over the C latent communities that represent how strongly the node belongs to each community. Also, each node and each community $c = 1, 2, \ldots, C$ have topic distributions θ and λ respectively where each entry is a probability to draw a certain topic. A pair of an attribute $a = 1, 2, \ldots, A$ and a topic $z = 1, 2, \ldots, K$ is associated with a multinomial distribution ϕ_{az} over the attributes and an entry ϕ_{az}^{x} represents a probability that the value x of the attribute a is drawn given the topic z. Lastly, each node u has a bias probability μ_u that tells us how likely the node takes an attribute value based on its own interests or the topics of the communities to which the node belongs. Table 3.1 lists the notations that are used throughout this chapter.

3.3.2 Modeling the Links of the Network

An attribute network G can be represented by an adjacency matrix $M \in \{0, 1\}^{U \times U}$ where the binary value of each entry indicates whether there is a link between the node u and the node v or not. In order to model the connections between the nodes in the network G, we use the community memberships π of the nodes and employ the generative process used in CESNA [26]. As described earlier, π_{uc} indicates how strongly the node u is associated with the community c. Two nodes u and v who belong to the same community c are more likely to establish a link between them as π_{uc} and π_{vc} have more similar values. Thus the probability that two nodes in the same community c are connected to each other can be defined as follows:

Table 3.1.: Notations used in the PTC model

Symbol	Description
G	attribute network
U	number of nodes
C	number of communities
A	number of attributes
K	number of topics
θ	multinomial distribution over topics of a node
λ	multinomial distribution over topics of a community
π_u	community memberships of node u
ϕ_{az}	multinomial distribution over values of attribute a given topic z
μ_u	probability that node u is biased to communities
z_{ua}	node u's topic on attribute a
c_{ua}	node u's community that is relevant to attribute a
x_{ua}	node u's value on attribute a
y_{ua}	node u's bias for attribute a

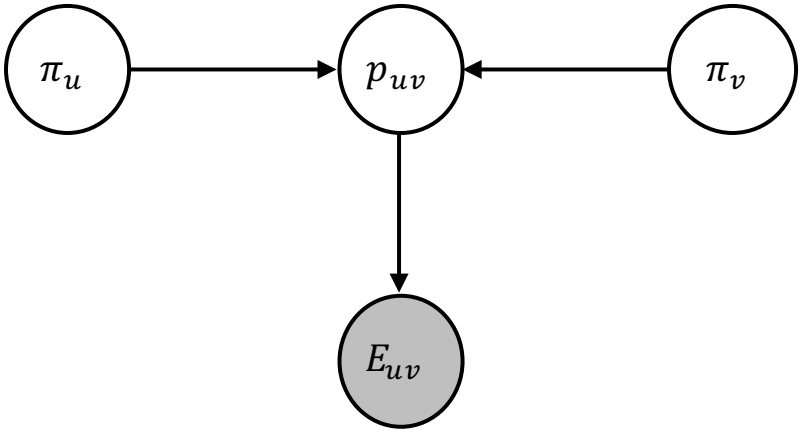


Fig. 3.1.: Modeling links

$$P_{uv}(c) = 1 - exp(-\pi_{uc} \cdot \pi_{vc})$$

Also, since we assume each node can belong to multiple communities and each community contributes to form a link independently, the probability of a link between u and v is obtained from the probability that u and v are not connected through any community:

$$P_{uv} = 1 - \prod_{c}(1 - P_{uv}(c)) = 1 - exp(\sum_{c} -\pi_{uc} \cdot \pi_{vc})$$

Then we can generate the adjacent matrix M by drawing a binary value for each entry m_{uv} according to the probability defined above. The corresponding graphical model representation is shown in Figure 3.1.

3.3.3 Modeling the Node Attributes

The attribute values can be modeled by borrowing the idea of generating words in LDA(Latent Dirichlet Allocation) model [63]. Each community and each node are associated with topic distributions (λ and θ). The latent topics could describe each of the attribute values held by nodes. Thus, if two nodes u and v belong to the same

community, then they are likely to have the same topic on a certain attribute and eventually have the same value on the attribute. However, all the attributes of nodes do not have to be relevant to communities. There might be some attributes that are generated regardless of the community memberships. For example, if a node has only few connections with other nodes and is actually not a member of any community in the network, then all the attribute values of the node might not be relevant to any community and not generated from communities' topics. Similarly, if a node belongs to only communities `Purdue alumni` and `Google employees` but has the value of "baseball" on the attribute *hobby*, then the attribute value might not be generated from the communities but from the node's personal interests. Therefore, for each attribute a, a node u first decides whether it is related to communities' topics or the node's personal topics by flipping a biased coin y(its bias to head is μ_u). If the attribute a is community-related($y = 1$), then the node u selects a community c according to π_u and draws a topic z from λ_c. If the attribute is not community-related($y = 0$), then the node draws a topic z from θ_u. Once a topic is decided for the attribute a, its value is decided based on the probability ϕ_{az} that is a multinomial distribution over the values of the attribute a.

While generating the node attribute values, we use the community memberships to select a community to which a node belongs. However, the node u's community membership π_u has non-negative real values which do not form a probability distribution. So we need to make it follow a probability distribution over a set of communities by normalizing the values. We use π'_u to denote the normalized community memberships of u ($\pi'_{uc} = \pi_{uc} / \sum_{c'} \pi_{uc'}$). The model has the plate notation as shown in Figure 3.2 and the generative process can be described as follows:

For a node u, if an attribute a follows a topic distribution of the node u, then the the joint distribution of the attribute's values X_{ua} of the node u is defined as:

$$\begin{aligned} &P(X_{ua}, y_{ua} = 0|\mu_u, \eta, \phi) = P(y_{ua} = 0|\mu_u) \\ &\int P(\theta_u|\eta)\Big(\sum_{z_{ua}} P(z_{ua}|\theta_u)\, P(x_{ua}|z_{ua}, \phi_a)\Big)\, \mathrm{d}\theta_u \end{aligned} \tag{3.1}$$

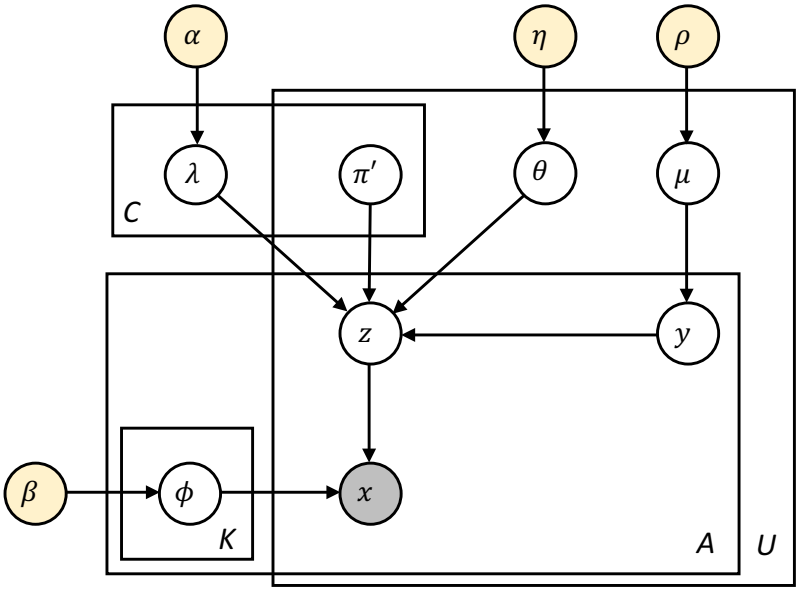


Fig. 3.2.: Modeling attributes

Algorithm 2 Generative process of node attribute values

for each attribute a and each topic z **do**
  draw $\phi_{az} \sim \text{Dir}(\beta_a)$
end for
for each community c **do**
  draw $\lambda \sim \text{Dir}(\alpha)$
end for
for each user u **do**
  draw $\mu_u \sim \text{Beta}(\rho)$
  for each attribute a **do**
    draw $y_{ua} \sim \text{Bernoulli}(\mu_u)$
    if $y_{ua} == 1$ **then**
      draw $c_{ua} \sim \text{Multi}(\pi'_u)$
      draw $z_{ua} \sim \text{Multi}(\lambda_{c_{ua}})$
    else
      draw $\theta_u \sim \text{Dir}(\eta)$
      draw $z_{ua} \sim \text{Multi}(\theta_u)$
    end if
    draw $x_{ua} \sim \text{Multi}(\phi_{az_{ua}})$
  end for
end for

For a node u, if an attribute a follows a topic distribution of a community c to which the node u belongs, then the the joint distribution of the attribute's values X_{ua} of the node u is defined as:

$$\begin{aligned} P(X_{ua}, y_{ua} = 1 | \mu_u, \alpha, \phi, \pi'_u) = \\ P(y_{ua} = 1|\mu_u) \int P(c|\pi_u) \\ \Big(\int P(\lambda_c|\alpha) \Big(\sum_{z_{ua}} P(z_{ua}|\lambda_c)\, P(x_{ua}|z_{ua}, \phi_a) \Big)\, \mathrm{d}\lambda_c \Big)\, \mathrm{d}\pi_u \end{aligned} \tag{3.2}$$

From the joint distributions defined above, we can obtain the full joint distribution of all observed attribute values X in the network G as follows:

$$P(X|\Theta) = \prod_u \int P(\mu_u|\rho)\Big(\prod_a \Big\{ P(X_{ua}|\mu_u, \eta, \phi) + P(X_{ua}|\mu_u, \alpha, \phi, \pi'_u)\Big\}\Big)\, \mathrm{d}\mu_u \tag{3.3}$$

where Θ is a set of all parameters in the model. We will discuss how the hidden variables should be estimated in the following section.

3.4 Inference

Given an observed attributed network G and associated node attribute values X, we need to estimate the hidden variables θ_u, θ_c, π, ϕ, μ, and ϕ. In PTC model, both G and X are generated from the community memberships π and they are conditionally indenpendent of each other given π. Also, as seen in Section 3.3.2, the links depend on only the community memberships, while the node attribute values depend on not only the community memberships but also other hidden variables such as topic distributions (θ and λ), attribute value distribution (ϕ), and bias distribution μ. Thus, our inference strategy is basically to run two separate inference steps alternately: one is for estimating the topic related parameters (θ, λ, ϕ, μ) on which the node attributes depend and the other is for estimating the community memberships (π).

3.4.1 Updating Topic Related Parameters

First, given the observed attribute values X we should estimate θ, λ, ϕ, and μ while assuming the values of π are predefined. Unfortunately, it is intractable to directly solve the distributions in Equations (3.1), (3.2), and (3.3) defined in Section 3.3.3. Thus, instead of estimating θ, λ, ϕ, and μ directly, the Gibbs sampling technique is

used for the model to learn the parameters. We sample biases and topics for each pair of user and attribute according to the conditional probabilities defined below:

$$P(y_{ua} = 0|\cdot) \propto \frac{n_{uy}(u,0)+\rho}{\sum_y n_{uy}(u,y)+2\rho} \cdot \frac{n_{uz}(u,z_{ua})+\eta}{\sum_k n_{uz}(u,k)+K\eta}$$

$$P(y_{ua} = 1|\cdot) \propto \frac{n_{uy}(u,1)+\rho}{\sum_y n_{uy}(u,y)+2\rho} \cdot \sum_c (\pi_{uc} \frac{n_{cz}(c,z_{ua})+\alpha}{\sum_k n_{cz}(c,k)+K\alpha})$$

$$P(z_{ua} = z|y_{ua} = 0,\cdot) \propto \frac{n_{uz}(u,z)+\eta}{\sum_k n_{uz}(u,k)+K\eta} \cdot \frac{n^{(a)}_{zx}(z,x_{ua})+\beta_a}{\sum_x n^{(a)}_{zx}(z,x)+X_a\beta_a}$$

$$P(z_{ua} = z|y_{ua} = 1,\cdot) \propto \frac{n_{cz}(c_{ua},z)+\alpha}{\sum_k n_{cz}(c,k)+K\alpha} \cdot \frac{n^{(a)}_{zx}(z,x_{ua})+\beta_a}{\sum_x n^{(a)}_{zx}(z,x)+X_a\beta_a}$$

where n_{uy} is the number of node u's attributes having bias y, n_{uz} is the number of times topic z is assigned to attributes based on node u, n_{cz} is the number of times topic z is assigned to attributes based on community c over all nodes, and $n^{(a)}_{zx}$ is the number of times topic z is assigned to the value x of attribute a.

We iterate the sampling until convergence and then estimate the parameters θ, λ, ϕ, and μ based on the samples:

$$\theta^{(k)}_u = \frac{n_{uz}(u,k)+\eta}{\sum_{k'} n_{uz}(u,k')+K\eta}$$
$$\theta^{(k)}_c = \frac{n_{cz}(c,k)+\alpha}{\sum_{k'} n_{cz}(c,k')+K\alpha}$$
$$\phi^{(x)}_{az} = \frac{n_{zx}(z,x)+\beta_a}{\sum_{x'} n_{zx}(z,x')+X_a\beta_a}$$
$$\mu^{(y)}_u = \frac{n_{uy}(u,y)+\rho}{\sum_{y'} n_{uy}(u,y')+2\rho}$$

3.4.2 Updating Community Memberships

Once the hidden parameters θ, λ, ϕ, and μ are estimated through the Gibbs sampling, we are ready to estimate the community memberships π and π can be estimated by the maximum likelihood estimation. That is, we can estimate π by finding the optimal π that maximizes the log-likelihood of the observed G and X:

$$\arg\max_{\pi} \log P(G, X|\pi, \Theta)$$

Since an observed network G and node attribute values X are conditionally independent of each other given the community memberships π and other parameter set Θ, the log-likelihood of G and X is decomposed into two parts as follows:

$$\log P(G, X|\pi, \Theta) = \log P(G|\pi) + \log P(X|\pi, \Theta)$$

Assuming every link in G is generated independently, the likelihood of G is

$$P(G|\pi) = \prod_{(u,v)\in E} (1 - exp(-\sum_{c} \pi_{uc}\pi_{vc})) \prod_{(u,v)\notin E} (exp(-\sum_{c} \pi_{uc}\pi_{vc})))$$

and thus the log-likelihood $\ell(G)$ is given by

$$\ell(G) = \sum_{(u,v)\in E} \log(1 - exp(-\sum_{c} \pi_{uc}\pi_{vc})) - \sum_{(u,v)\notin E} \sum_{c} \pi_{uc}\pi_{vc}$$

Likewise, the likelihood of X is

$$P(X|\pi, \Theta) = \prod_{u} \prod_{k} (Q_{uk}^{x_{uk}} (1 - Q_{uk})^{(1-x_{uk})})$$

where Q_{uk} is the probability that the node u takes the attribute value k, which is defined as follows:

$$Q_{uk} = \sum_{z} \phi_{az}^{(x_{uk})} (\mu_u \sum_{c} \frac{\pi_{uc'}}{\pi_u^0} \lambda_c^{(z)} + (1 - \mu_u)\theta_u^{(z)})$$

where $\pi_u^0 = \sum_c \pi_{uc}$. Then, the log-likelihood $\ell(X)$ is

$$\ell(X) = \sum_{u,k} (x_{uk} \log Q_{uk} + (1 - x_{uk}) \log(1 - Q_{uk}))$$

In order to find π that maximizes $\ell(G)$ and $\ell(X)$, we use the gradient method. Since the community memberships of the nodes in the network are independent, we can update new community memberships node by node. We denote the log-likelihood of G and the log-likelihood of X specific to only a node u by $\ell(G, \pi_u)$ and $\ell(X, \pi_u)$ respectively.

$$\ell(G, \pi_u) = \sum_{v \in N(u)} \log(1 - exp(-\sum_c \pi_{uc}\pi_{vc})) - \sum_{v \notin N(u)} \pi_{uc}\pi_{vc}$$

$$\ell(X, \pi_u) = \sum_k (x_{uk} \log Q_{uk} + (1 - x_{uk}) \log(1 - Q_{uk}))$$

where $N(u)$ is a set of neighbor nodes of u. From the partial derivatives of $\ell(G, \pi_u)$ and $\ell(X, \pi_u)$, each component of $\nabla\ell(G, \pi_u)$ and $\nabla\ell(X, \pi_u)$ are defined as follows

$$\frac{\partial \ell(G, \pi_u)}{\partial \pi_{uc}} = \sum_{v \in N(u)} \pi_{vc} \frac{exp(-\sum_c \pi_{uc}\pi_{vc})}{1 - exp(-\sum_c \pi_{uc}\pi_{vc})} - \sum_{v \notin N(u)} \pi_{vc}$$

$$\frac{\partial \ell(X, \pi_u)}{\partial \ell(\pi_{uc})} = \sum_k \frac{(x_{uk} - Q_{uk})Q'_{uk}}{(1 - Q_{uk})Q_{uk}}$$

where Q'_{uk} is the partial derivative of Q_{uk} with respect to π_{uc}. That is,

$$\frac{\partial Q_{uk}}{\partial \pi_{uc}} = \sum_z \phi_{az}^{(k)} \mu_u \Big(\frac{\lambda_c^{(z)} \pi_u^0 - \sum_{c'} \pi_{uc'} \lambda_{c'}^{(z)}}{(\pi_u^0)^2} \Big)$$

Since the community memberships are represented by non-negative real values, the new community memberships that are updated based on the gradients defined above should be projected onto positive value space.

$$\pi_{uc}^{new} = \max(0, \pi_{uc}^{old} + \alpha(\frac{\partial \ell(G, \pi_u)}{\partial \pi_{uc}} + \frac{\partial \ell(X, \pi_u)}{\partial \pi_{uc}}))$$

where α is a learning rate.

3.5 Experiments

We evaluate our generative model on the two different predictive tasks: 1) community detection and 2) profiling missing attribute values.

3.5.1 Detecting Hidden Communities

Dataset Basically PTC learns the community memberships based on both the network structure and the node attributes. In addition to the network structure and the node attributes, we need to know the actual community labels for each node as the ground-truth so that we can evaluate the results of community detection done by PTC and baseline methods. With the requirements, we consider the Facebook network dataset that is publicly accessible in the site of `Stanford Network Analysis Project`[1]. It consists of 10 different ego networks, each of which contains a set of communities. Since different ego networks are associated with different set of attributes, we evaluate our PTC for each of the networks separately. Tabel 3.2 shows some statistics of the ego networks.

Baselines CESNA [26] is state of the art model for the community detection problem and it uses both the network structure and the node attributes to detect latent communities as PTC does. One might be also interested in comparing PTC with other methods based on only the network structure. So we consider two other methods that focus on the network structure without taking into account any information about node attributes. One is MaxMod [57] and the other is InfoMap [56]. MaxMod basically discovers a set of communities in a network such that the modularity of the network is maximized. InfoMap works based on the probability flow through random walks. The implementations of those three baseline methods are provided by the SNAP (Stanford Network Analysis Platform) system[2].

Experimental Results There are various metrics to measure the performance of a community detection method. Among them we take an evaluation function that were used in [26, 46], which is defined as follows:

$$\frac{1}{2|C|}\sum_{C_i\in C}\max_{\tilde{C}_j\in\tilde{C}} J(C_i,\tilde{C}_j)+\frac{1}{2|\tilde{C}|}\sum_{\tilde{C}_j\in\tilde{C}}\max_{C_i\in C} J(\tilde{C}_j,C_i)$$

[1] `http://snap.stanford.edu/`

[2] `http://snap.stanford.edu/`

Table 3.2.: Facebook Network Statistics

Properties	ego1	ego2	ego3	ego4	ego5	ego6	ego7	ego8	ego9	ego10	Average
Number of nodes	348	1,045	790	755	547	227	59	159	170	66	416.60
Number of edges	5,038	53,498	28,048	60,050	9,626	6,384	292	3,386	3,312	540	17,017.40
Number of communities	24	9	17	46	32	14	17	7	14	13	19.30
Number of attributes	21	23	20	22	20	19	16	19	18	20	19.80
Number of values	224	576	319	480	262	161	42	105	63	48	228.00
Average community size	13.54	55.66	45.70	23.15	6.00	40.50	3.41	25.42	34.64	6.53	25.45
Communities per node	0.93	0.47	0.98	1.41	0.35	2.49	0.98	1.11	2.85	1.28	1.28

where $J(\cdot)$ is the Jaccard coefficient that measures similarity between a pair of sets, which is defined as the size of the intersection divided by the size of the union of the two sets. The evaluation function basically measures how much the ground-truth communities and the detected communities conform to each other. For each of the ground-truth communities, we find its most similar detected community based on the Jaccard coefficient between them. By taking the average of the maximal Jaccard coefficients over all ground-truth communities, we can see the degree of conformity between the two sets of communities. Note that there is no restriction on the number of detected communities. In other words, the number of detected communities depends on different methods and even on a single method because it can be specified as a parameter. Thus, we should measure the metric in the opposite direction as well, from the detected communities to the ground-truth communities. The average over the resulted measures should be reported as the performance of a method. Table 3.3 shows the performances of all the methods over the 10 different ego networks. The higher the value is, the better the performance is. We repeat our experiments five times with different samples and report the mean of the evaluation metric in five rounds, in order to make sure our experimental results are statistically meaningful.

There is no method that consistently outperforms other methods over the ego networks. PTC performs best for ego1, ego5, and ego6, CESNA performs best for ego2, and MaxMod performs best the rest of the ego networks. InfoMap does not win for any ego network. If we consider the average of the results as the overall performance then MaxMod could achieve the best performance. PTC can achieve better performance if a network is large enough in terms of the numbers of nodes, links, attributes, and hidden communities, since it can provide PTC with more evidences to estimate hidden variables that contribute to generate those components. One interesting observation is that MaxMod that uses only the network structure without considering the node attributes for community detection works outperforms methods that considers both the network structure and node attributes, which contradicts the results reported in [26]. This indicates either that using node attributes might

be information overloading so that it makes the model based on both information confusing while discovering communities or that Facebook ego networks simply do not have strong relationships between node attributes and latent communities.

3.5.2 Additional Task: Attribute Profiling

Dataset For the task of profiling missing attribute values, the Facebook ego networks used for community detection is not appropriate because users attribute values in the networks are tokenized and anonymized as binary vectors. Instead, we obtain LinkedIn dataset used in [22]. The authors of [22] collected the data by asking real users in LinkedIn to provide their attributes (e.g., *employer*, *college*, and *location*) and their ego networks (including the connections from the ego users to their friends and those among their friends) based on LinkedIn APIs. We merge all the ego networks to construct one big network where 19K users and 110K connections exist.

Baselines We use the following two competing methods as the baselines.

- **RN** The Relational Neighbor (RN) classifier [64] is based on the assumption that if two nodes are connected to each other then they are likely to share the same attribute value. It estimates the probability that a user holds an attribute value as the weighted average of the probabilities of its labeled neighbors (i.e., weighted majority voting). Despite of the simplicity of the method, previous studies show [64, 65] that it performs surprisingly well in many settings, even compared to complex models.

- **CP** Co-profiling (CP) approach [22] jointly learns users' attributes and relational types of their friends in a network. It profiles user attributes by propagation from friends in certain relational types, and profiles relational types for friends based on inferred attribute values and the network structure.

Experimental Results We use the accuracy to measure the performance of a method. It is the ratio of the number of the correctly profiled users to the total

Table 3.3.: Community Detection Performance

Method	ego1	ego2	ego3	ego4	ego5	ego6	ego7	ego8	ego9	ego10	Average
PTC	**0.220**	0.211	0.202	0.293	**0.175**	**0.540**	0.217	0.339	0.418	0.341	0.296
CESNA	0.198	**0.255**	0.388	0.322	0.087	0.375	0.246	0.425	0.286	0.424	0.301
MaxMod	0.196	0.151	**0.464**	**0.370**	0.097	0.382	**0.439**	**0.482**	**0.435**	**0.482**	**0.350**
InfoMap	0.181	0.191	0.321	0.246	0.095	0.321	0.367	0.365	0.210	0.424	0.272

Table 3.4.: Mean Accuracy of Attribute Profiling

	RN	CP	PTC
Employer	0.53	**0.60**	0.58
College	0.50	**0.61**	0.60
Location	0.62	0.65	**0.68**

number of test users. Since a user may have multiple values (e.g., Google, Facebook) for some attribute (e.g., employer), we define that a user is correctly profiled if the profiled value matches any of his true values. Also, as done in [22], we randomly take 20% of users as labeled samples and their attribute values are revealed to our algorithm and baseline methods, and then we profile missing attributes based on the attributes of the labeled users and the network structure with our algorithm and baseline methods.

Table 3.4 shows the mean accuracy of attribute profiling. RN achieves a reasonable accuracy (i.e., 53%) when using 20% of nodes as labeled samples, which demonstrates the usefulness of social connections for profiling missing attributes. However, it shows the worst performance among the three methods. The reasons could be 1) it fails to utilize additional information about connections available in networks and 2) the assumption that two connected users are likely to share the same attribute does not precisely capture the correlation between users attributes and their social connections. Both CP and PTC outperform RN over all considered attributes, which implies that it is obvious that joint learning of node attributes and network connections is helpful to better capture the correlation between the two components. While PTC is able to better profile the *location* attribute than CP, CP achieves higher accuracy than PTC for the other two attributes. It can be explained by the fact that there exist more various values on employer and college than location, which makes it hard for PTC to learn accurate probability distributions over attribute values.

3.6 Summary

In this chapter, we have devised a new generative model for an attributed network which learns latent communities behind nodes in the network. In order to discover the latent communities, we made three assumptions. First, community memberships affect the formation of the network structure and the node attribute values. That is, if two nodes belong to the same community then they are not only likely to be connected to each other but also likely to share the same value on certain attributes. Second, given an attribute each community and each node have their own topic distribution that represents the strength of their interests on latent topics. These latent topics determine which value a node would take for the attribute. Third, every attribute is relevant to a topic coming from either a community or an individual node and different nodes have different level of biases over the attributes. Based on the assumptions, PTC generates an attribute network consisting of both the network connections and node attributes in such a way that two nodes in the same community are likely to form a link between them and share the same value on only community-related attributes. Also, the attribute values are generated from hidden topics that represent interests of individual nodes and latent communities. We developed an inference algorithm that is used to estimate the hidden parameters as well as the community memberships.

The experiments for the performance evaluation were conducted on the Facebook ego networks. The results show that PTC performs well for some ego networks even though it does not outperform other baseline methods for every network. Per our analysis, PTC tends to better detect hidden communities in case where the size of network is large in terms of the number of nodes, the number of links, the number of attributes, and the number of hidden communities, because a large network can provide the model with more evidence to learn network structure attribute distributions. For future works, we will further investigate in which case node attributes effectively contribute to better detect latent communities. Definitely, every attribute is not helpful for discovering communities. If so, we may come up with how to selectively use

some of the attributes for better learning. Also, we chose the number of topics and the number of communities empirically but they could be obtained from a function of the network size including the number of nodes, the number of links, and the number of attributes. Lastly, since PTC is able to generate both the network structure and node attributes probabilistically, it can inherently predict missing attribute values or missing links. Given an attribute network with missing attribute values/links, once the model learns all the parameters then it will be able to compute probabilities that each of the missing information is generated based on observed information. The experimental results show that PTC is comparable to state-of-the-art models and it can better profile missing values of attributes whose values are less various. That is because too many different attribute values make it hard for PTC to learn accurate attribute distributions.

4. ATTRIBUTE ASSOCIATION AWARE NETWORK EMBEDDING

Network embedding aims to learn low-dimensional vector representations for nodes in a network that preserve structural characteristics. It has been shown that such representations are helpful in several graph mining tasks such as node classification, link prediction, and community detection. Some recent works have attempted to extend the approach to attributed networks in which each node is associated with a set of attribute values. They have focused on homophily relationships by forcing nodes with similar attribute values to obtain similar vector representations. This is unnecessarily restrictive and misses the opportunity to harness other types of relationships revealed by patterns in attribute values of connected nodes for learning insightful relationships. In this chapter, we propose a new network attributed embedding framework called *A3embed* that is aware of attribute associations. *A3embed* favors significant attribute associations, not merely homophily relationships, which contributes to its robustness to diverse attribute vectors and noisy links. The experimental results on real-world datasets demonstrate that the proposed framework achieves better performance on different graph mining tasks compared to existing models.

4.1 Introduction

Data mining and machine learning tasks that aim to exract insightful information from real-world data must increasingly handle complex network data. However, it is not realistic to apply standard machine learning models directly to network data because the network itself lacks fruitful feature representations that can provide informative patterns among nodes and links. To overcome this challenge, researchers have proposed methods for learning new network representations that are usually repre-

sented by low-dimensional continuous vectors. Such vectors in a continuous feature space represent nodes in a more abstract form while preserving structural proximities among the nodes and thus are better suitable for various data mining and machine learning tasks.

While recently proposed network embedding algorithms show acceptable performance on various tasks [66–71], they are limited to networks without attributes. However, an increasing number of real-world objects and applications are modeled with attributed networks and it has become increasingly important to analyze the attributes together with network structure. For example, in social networks such as Twitter and Facebook where user profile information is captured using attribute values, many users that have similar attributes are not connected to each other. That is, structural proximity is not sufficient to explain whether nodes in a network are similar or dissimilar. In that case, if available, node attributes can bring us a huge opportunity to capture the nodes' underlying similarity.

Alternative methods taking into account node attribute values in network embedding have been proposed more recently [72, 73]. They basically have the same motivation, where nodes with similar attribute values are located closely in the low-dimensional embedding space. It seems quite reasonable because many previous works have shown that nodes in a network tend to establish homophily relationship in terms of their attributes [19–21]. However, there actually exist more diverse relationships in real-world networks [74]. Also, even though such relationships may not be observed as frequently as homophily relationships, they can be more important for understanding various dynamics in complex networks and can be captured by considering statistical significance [74]. Unfortunately, the notion of attribute associations, defined as co-occurred attribute values between connected nodes, along with their significance has been ignored by existing network embedding methods despite its potential impact on network embedding.

Consider an attribute network where each node is associated with its attribute values. A pattern of node attribute values which co-occur between connected nodes

might be of interest because it can reveal the type of relationships among nodes clearly along with their structural proximity. As the number of attributes increases, it is unlikely that homophily relationships alone are dominant in the entire network. For example, in a social network, people working at `Google` may establish many links to co-workers but they may have different alma maters (e.g. connections between {`Google`, `Stanford`} – {`Google`, `UCLA`}). Moreover, if we consider other attributes such as *nationality* and *major*, one expects to observe much more diverse patterns of co-occurring attribute values on the connections among Google employees. That is, the homophily relationship may not be sufficient to capture underlying similarities among the nodes in a network. In such cases it is clear why it is important to consider such patterns which are called attribute associations and possibly significant, as well as attribute similarity represented by homophily relationship for successful attributed network embedding. Even if a particular attribute association is frequently observed among connected nodes, the frequency itself does not tell us how meaningful it is. Whether it is really meaningful or not depends more on how many nodes hold the attribute vectors involved with the attribute association and how many of them are connected to each other. The relative frequency of attribute association over the number of such nodes is more indicative of whether two nodes with the attribute association should be considered similar – and therefore be close to each other – in a low-dimensional embedding space.

In this chapter, we study the network embedding problem, especially for attributed networks, and propose a new embedding method that exploits attribute associations in learning low-dimensional representations. We experimentally evaluate our proposed method *A3embed* using two real-world attributed networks including BlogCatalog and Flickr. We compare the performance of *A3embed* with state-of-the-art network embedding methods [67–69, 73]. Our observations from the experiments demonstrate that new network representations learned by *A3embed* can be better generalized to various prediction and visualization tasks. Especially, *A3embed* is superior to not only some baselines that use only network structure but also others that

use augmented attribute information in addition to the structural information for all considered tasks.

We summarize the contributions of our proposed method as follows:

- We propose a novel network embedding method, called *A3embed* (**A**ttribute **A**ssociation **A**ware network **embed**ding). The method aims to obtain new representations of nodes in an attributed network by jointly modeling both structural and attribute information in the network while capturing attribute associations.

- We show why it is important to consider attribute associations on the task of network embedding.

- We empirically demonstrate how successfully *A3embed* learns new network representations in a low-dimensional space and how effective the learned representations are for downstream machine learning tasks on different real-world attributed networks.

This chapter is organized as follows. In Section 4.2, we introduce previous works related to our problem and discuss how our problem differs from them. In Section 4.3, we define the problem of network embedding and provide basic background concepts, and then introduce a novel method to solve the problem of attributed network embedding. We present our experimental observations over different network embedding methods on real-world datasets in Section 4.4. Finally, we summarize the chapter in Section 4.5.

4.2 Related Work

The problem of network embedding that aims to learn new representations for networks has been attracted by data mining and machine learning communities due to its practical importance in various applications such as node classification, link

prediction, visualization, network compression, and clustering. Especially, representational learning has been actively studied in the field of natural language processing and recently neural network based models have been proposed for feature learning of discrete objects such as words. In particular, the Skip-gram model [75, 76] has been applied to many different kinds of applications [66, 77–79]. The model basically aims to learn continuous feature representations for words in a corpus by optimizing an objective function based on the likelihood of observing surrounding words for a given word. The idea of the model is based on the distributional hypothesis which states that words in similar contexts tend to have similar meanings. That is, similar words tend to appear in similar word neighborhoods. More specifically, the model tries to embed each of the words of a document such that the word's features can predict its context that consists of the neighboring words appearing inside a window centered on the word. The feature representations are learned by optimizing the likelihood objective using stochastic gradient descent with negative sampling technique. The basic idea of the model is based on the distributional hypothesis which states that words in similar contexts tend to have similar meanings. That is, similar words tend to appear in similar word neighborhoods.

The data mining and machine learning communities have been attracted to the problem of network embedding that aims to learn new representations for networks due to its practical importance in various applications such as node classification, link prediction, visualization, network compression, and clustering. Recently, many researchers have developed methods to learn network representations which are based on the Skip-gram model [75, 76] that aims to learn continuous feature representations for words in a corpus. *DeepWalk* [66] is the first work that established an analogy for networks by representing a network as a document. While a document includes a sequence of words, nodes in a network do not have any ordered sequences among them. The idea to obtain a sequence of nodes from a network is to consider a set of short truncated random walks as its own corpus, and the nodes as its own words. Then the same optimization framework as one for the Skip-gram model can be applied to the

set of node sequences obtained from repeated random walks. In [68] the authors proposed a new algorithmic framework called *node2vec* for learning continuous feature representations for nodes in networks using a biased random walk procedure that smoothly interpolate between Breadth-First Search (BFS) and Depth-First Search (DFS). However, those models exploit only network structure when learning feature representations without taking into account any other information such as node attributes.

In the case of citation networks, where nodes come with text information, such auxiliary information can be useful for learning richer representations. [80] proposed text-associated DeepWalk (*TADW*) that incorporates text features of nodes into network representation learning under the framework of matrix factorization. *TriDNR* [81], a tri-party deep network representation model, is based on a coupled neural network that exploits inter-node relationships, node-content correlation, and node-label correspondence in a network to learn an optimal representation for each node in the network. Even though *TADW* and *TriDNR* use rich information in addition to network structure for learning network representations, the text information is inherently different from node attributes in that text information itself includes a sequence of words so as to be easily exploited by neural networks based on the Skip-gram model.

While all the methods introduced above work only for networks without node attributes, [72, 73] exploit node attribute values to get richer representations for networks if node attribute are available. *LANE* [72] is a semi-supervised model that incorporates node labels into embedding representation learning for attributed networks. *AANE* [73] also learns low-dimensional representations based on the decomposition of attribute affinity and the embedding difference between connected nodes in a distributed way at scale. Both of the methods jointly model the network structure and node attributes but they are limited to attribute similarity. That is, nodes have a chance to have similar representations only when their attribute values are similar. In contrast, our proposed model considers more diverse patterns of co-occurring attribute values.

4.3 Attribute Association Aware Network Embedding

In this section, we first define the network embedding problem and then introduce our proposed method that learns network representations for attributed networks. Table 4.1 presents the notations we use throughout this chapter.

4.3.1 Problem Definition

Consider an attributed network denoted by $G = (V, E, X)$ where $V = \{v_1, v_2, \cdots, v_n\}$ is a set of n number of nodes, $E = \{e_{ij}\}_{i,j=1}^{n}$ is a set of edges, and $X = \{\mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_n}\}$ is a set of attribute vectors, each of which is associated with a node in V. The attribute vector $\mathbf{x_i}$ of the node v_i that holds l different attributes is represented by a vector of l numerical values. An edge e_{ij} in E can be associated with its weight s_{ij} representing how strongly two individual nodes are connected to each other. If v_i and v_j are not connected by an edge, then $s_{ij} = 0$. In case of unweighted networks, $s_{ij} = 1$ for all edges e_{ij}.

Network embedding aims to learn new representations of nodes in a low-dimensional feature space by finding a mapping function $\mathcal{F} : V \mapsto \mathbb{R}^d$ where $d \ll n$ is the number of dimensions. Since a raw representation of nodes is too sparse, it is hard to observe interesting patterns, or obtain insightful knowledge, by applying standard machine learning models directly. However, the low-dimensional representations learned by network embedding may contain important underlying information over the network nodes in an abstract form. The key point to achieve good representations for attributed networks is to preserve the structural and attribute proximity of nodes [72,73]. However, the sparsity of attribute space and diversity of attribute vectors render attribute proximity insufficient to account for actual similarity of nodes. In Section 4.3.2, we will introduce the notion of attribute association and explain why it needs to be considered for the network embedding task.

Table 4.1.: Basic notations

Notation	Meaning
$G = (V, E, X)$	attributed network
n	number of nodes
l	number of attributes
s_{ij}	weight of edge e_{ij}
$\mathbf{y_i}$	attribute embedding of node v_i
$\mathbf{z_i}$	structural embedding of node v_i
$\mathbf{h_i}$	joint representation of node v_i
$W_1^{(k)}, \mathbf{b_1}^{(k)}$	k-th layer weights and biases in attribute modeling
$W_2^{(k)}, \mathbf{b_2}^{(k)}$	k-th layer weights and biases in structure modeling
$W_3^{(k)}, \mathbf{b_3}^{(k)}$	k-th layer weights and biases in joint modeling
m_1, m_2, m_3	number of layers for each modeling component
ω	hyperparameter that controls weights on the latent representation from modeling node attributes
ζ	negative penalty to control the importance of attribute association
$\mathcal{N}_i$	neighbors of node v_i
σ	activation function
R	regularization function
$\mathcal{L}_{\{sim,ass,net\}}$	loss functions for attribute similarity, attribute association, and network structure
p	in-community link probability in a synthetic graph
q	cross-community link probability in a synthetic graph
r	number of distinct attribute vectors in a community in a synthetic graph

4.3.2 Attribute Associations

We first define an attribute association as follows,

Definition 4.3.1 *Given two nodes v_i and v_j, the attribute association between them is defined as a relationship of co-occurred attribute values that appear in the pair of corresponding attribute vectors $\mathbf{x_i} = [x_i^1, x_i^2, \cdots x_i^l]$ and $\mathbf{x_j} = [x_j^1, x_j^2, \cdots x_j^l]$.*

Every pair of nodes has its attribute association, and thus there are as many attribute associations as the number of edges in E if all the nodes in V have distinct attribute vectors. Note that an attribute association of $\mathbf{x_i}$ and $\mathbf{x_j}$ is associated with not only the nodes v_i and v_j but also any pairs of nodes that have the same attribute vectors as $\mathbf{x_i}$ and $\mathbf{x_j}$. As the number of attributes increases in a network, the sparsity of attribute vectors would be higher and it is more likely for nodes to have diverse attribute vectors. However, even though two nodes have different attribute vectors, it does not mean necessarily that they are not similar. That is because it is also possible for some different attribute values to share similar topics or be correlated to each other. For example, in a social network where each individual is associated with their personal profile, some users may have `google`, `facebook`, `J.P Morgan`, `Goldman Sachs`, and so on for the attribute *employer*. In terms of their context, `google` and `facebook`, Internet service companies, are closer to each other rather than to the other two finance companies, and vice versa, and thus it is expected that users working at `google` (or `J.P Morgan`) are more likely to be linked with users working at `facebook` (or `Goldman Sachs`) even if they have different values for the attribute. In other words, dissimilarity of node attribute values may not neccessarily imply dissimilarity of nodes. This motivates us to consider various patterns of co-occured attribute values that are represented by attribute associations for network representation learning. Similarly, it is not always true that two nodes with exactly the same attribute vectors must be similar. Even though a number of nodes are associated with a particular attribute vector, if only a few of them are connected to each other, it is hard to say that all the nodes with the attribute vector are similar and should be located closely

in the embedding space. Thus, it is important to consider statistically significant attribute associations for more insightful network analysis [74]. In this paper, we do not compute the actual statistical significance of attribute associations but introduce the basic idea of jointly modeling node attributes and network structure for the task of learning network representations. That is, for a given attribute association of $\mathbf{x_i}$ and $\mathbf{x_j}$, we say the attribute association between them is more **significant** than another association of $\mathbf{x_m}$ and $\mathbf{x_n}$ if the nodes with the association of $\mathbf{x_i}$ and $\mathbf{x_j}$ are more densely connected to each other compared to the connections among the nodes with $\mathbf{x_m}$ and $\mathbf{x_n}$. Such nodes with more significant associations should be closer to each other than ones with less significant associations in the embedding space. We explain how the notion of significance should be considered in 4.3.3.

4.3.3 *A3embed*

We now propose a new network embedding method called *A3embed* using attribute associations for attributed networks. *A3embed* consists of two parts: one is for modeling attribute associations and the other is for modeling network structure. The idea is straightforward. If two nodes share similar attribute values and/or the attribute association between them is significant, then they should be close to each other in the low-dimensional embedding space. Likewise, if two nodes share many common neighbors and therefore are structurally similar to each other, then they should be located closely as well. In this way, we can preserve both attribute proximity and structural similarity while keeping patterns of significant attribute associations in the embeddings. The overall framework of *A3embed* is illustrated in Figure 4.1.

Modeling Attribute Associations

We basically want to not only preserve the attribute similarity but also employ significant attribute associations. First of all, in order to model the attribute similarity among network nodes, we apply a deep autoencoder [69, 82, 83] to the set of all

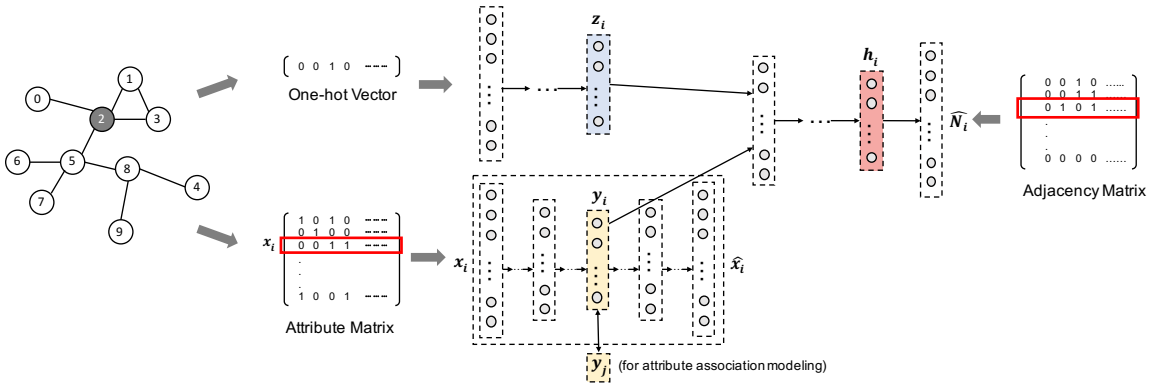


Fig. 4.1.: Framework of *A3embed*: For each node, its attribute vector and one-hot vector are fed into the deep model, and then its attribute and structural information are jointly modeled to predict its neighbors.

attribute vectors X. An autoencoder neural network, consisting of the encoder and decoder, is an unsupervised learning algorithm that applies backpropagation, setting the target values or outputs to be equal to the inputs. In other words, it tries to learn an approximation to the identity function, so as to output $\hat{\mathbf{x}}_\mathbf{i}$ that is similar $\mathbf{x}_\mathbf{i}$, by having a non-linear function that encodes $\mathbf{x}_\mathbf{i}$ to new representations $\mathbf{y}_\mathbf{i}$ and another non-linear function that reconstructs $\hat{\mathbf{x}}_\mathbf{i}$ from $\mathbf{y}_\mathbf{i}$. As a result, the learned $\mathbf{y}_\mathbf{i}$ in the middle of the autoencoder can be considered as compressed and latent representations of $\mathbf{x}_\mathbf{i}$. If we have multiple layers for the encoder and the decoder, then the latent representation ${\mathbf{y}_\mathbf{i}}^{(k)}$ is formulated as follows:

$$\begin{aligned} {\mathbf{y}_\mathbf{i}}^{(k)} &= \sigma(W_1^{(k)}{\mathbf{y}_\mathbf{i}}^{(k-1)} + {\mathbf{b}_\mathbf{1}}^{(k)}) \\ &= \sigma(W_1^{(k)}\sigma(W_1^{(k-1)}{\mathbf{y}_\mathbf{i}}^{(k-2)} + {\mathbf{b}_\mathbf{1}}^{(k-1)}) + {\mathbf{b}_\mathbf{1}}^{(k)}) \\ &= \sigma(W_1^{(k)}(\cdots\sigma(W_1^{(1)}\mathbf{x}_\mathbf{i} + {\mathbf{b}_\mathbf{1}}^{(1)})\cdots) + {\mathbf{b}_\mathbf{1}}^{(k)}) \end{aligned} \tag{4.1}$$

where σ is an element-wise activation function such as a sigmoid function or a rectified linear unit. Similarly, the reconstruction $\hat{\mathbf{x}}_\mathbf{i}$ that has the same shape as $\mathbf{x}_\mathbf{i}$ is mapped from $\mathbf{y}_\mathbf{i}$ by stacking hidden layers of non-linear functions on the top of $\mathbf{y}_\mathbf{i}$

in the reversed shape of the encoder. Then the autoencoder is trained to minimize reconstruction errors, represented by

$$\mathcal{L}_{sim} = \mathcal{L}(\mathbf{x}, \hat{\mathbf{x}}) = \sum_{i=1}^{n} \|\mathbf{x_i} - \hat{\mathbf{x}}_\mathbf{i}\|_2^2 \tag{4.2}$$

As discussed in [69], if an input vector, which is an attribute vector in our setting, is very sparse, then the autoencoder is prone to reconstruct zero values in an attribute vector rather than non-zero values. We avoid that by imposing a higher penalty to the reconstruction error of non-zero values than the error of zero-values as follows:

$$\mathcal{L}_{sim} = \sum_{i=1}^{n} \|(\mathbf{x_i} - \hat{\mathbf{x}}_\mathbf{i}) \odot \mathbf{t_i}\|_2^2 \tag{4.3}$$

where $\mathbf{t_i} = [t_{i1}, t_{i2}, \cdots, t_{il}]$ and $t_{ij} = \tau > 1$ for $j = 1, \cdots, l$ if v_i has a value for the j-th attribute, otherwise $t_{ij} = 1$. The $\odot$ operator performs an element-wise multiplication between two vectors.

As we put attribute vectors of nodes repeatedly into the autoencoder, nodes with similar attribute vectors must have similar latent representations $\mathbf{y}$. However, the loss function above is not enough to model potentially significant attribute associations that exist in many real-world applications as well.

The key idea of using significant attribute associations is to see, given an attribute association, how many nodes have the attribute vectors involved with the association and how frequently the association is observed over links among the nodes relatively. That is, if two nodes v_i and v_j are associated with a particular attribute association and its attribute vectors $\mathbf{x_i}$ and $\mathbf{x_j}$ appear many times on other connected nodes as well, then we make the corresponding latent representations $\mathbf{y_i}$ and $\mathbf{y_j}$ similar, no matter if they have dissimilar attribute values or not. Note that the frequency of attribute vectors itself may not be important. In contrast, if there are many nodes that hold either $\mathbf{x_i}$ or $\mathbf{x_j}$ but only few of them are connected, then we make $\mathbf{y_i}$ and $\mathbf{y_j}$ away from each other. The following loss function takes care of attribute associations with the idea above:

$$\mathcal{L}_{ass} = \sum_{i,j=1}^{n} s_{ij} \cdot \|\mathbf{y_i} - \mathbf{y_j}\|_2^2 \tag{4.4}$$

where s_{ij} is the edge weight between v_i and v_j and $s_{ij} = \zeta < 0$ if there is no edge between v_i and v_j. For nodes v_i and v_j that are not connected, we give a negative penalty ζ to their corresponding latent representations $\mathbf{y_i}$ and $\mathbf{y_j}$ such that they are not close. The effect of the negative penalty term ζ guarantees that 1) even two nodes with the same attribute vectors could be apart if such attribute association is very rare in the network and 2) even two nodes with different attribute vectors could be close if other nodes with such vectors are connected more densely than usual. The choice of ζ also controls how aggressively *A3embed* models attribute associations. If ζ is very low, we rarely penalize attribute associations that appear between unconnected nodes. Thus, only node pairs with very significant attribute associations will be mapped in close proximity to each other in the embedding space.

Modeling Structural Proximity

While some existing works take into account preserving the first-order and second-order proximity simultaneously [67, 69], we focus on the second-order proximity, i.e., common neighbor structure. This is acceptable because the first-order proximity is already considered to some extent when modeling attribute associations. Of course, even though two nodes are directly connected, they may not have similar low-dimensional representations if the attribute association is too weak. This sounds reasonable unless a pair of connected nodes necessarily share common values on most attributes.

For a given a node v_i, we use its one-hot vector $\mathbf{v_i}$ as an input to *A3embed*. It is fed-forward into a multi-layer perceptron and its latent representation $\mathbf{z_i}$ is combined with another latent representation $\mathbf{y_i}$ to produce a joint representation $\mathbf{h_i}$ of both network structure and node attribute values. Then the joint representation $\mathbf{h_i}$ is further fed-forward into following hidden layers and the final joint representation is used to

predict neighbors of the node v_i. The formulations of the latent representations at each layer are as follows:

$$\begin{aligned} \mathbf{z_i}^{(k)} &= \sigma(W_2^{(k)}\mathbf{z_i}^{(k-1)} + \mathbf{b_2}^{(k)}) \\ &= \sigma(W_2^{(k)}(\cdots\sigma(W_2^{(1)}\mathbf{v_i} + \mathbf{b_2}^{(1)})\cdots) + \mathbf{b_2}^{(k)}) \end{aligned} \tag{4.5}$$

$$\begin{aligned} \mathbf{h_i}^{(0)} &= [\omega\mathbf{y_i}, \mathbf{z_i}] \\ \mathbf{h_i}^{(k)} &= \sigma(W_3^{(k)}\mathbf{h_i}^{(k-1)} + \mathbf{b_3}^{(k)}) \end{aligned} \tag{4.6}$$

where ω is a hyperparameter that controls weights on the latent representation from modeling node attributes when constructing the first joint representation by concatenation.

Lastly, we predict the neighbors $\mathcal{N}_i$ of the input node v_i using the final joint representation $\mathbf{h_i}$. The output vector $\hat{\mathcal{N}}_i$ in *A3embed* should be close to a row or column vector of an adjacency matrix indicating neighbor nodes, and thus it is a multi-label classification task. The predictive probabilities for the neighbor nodes in $\mathcal{N}_i$ are obtained independently by placing a vector of sigmoids. Then the output vector $\hat{\mathcal{N}}_i$ is computed as follows:

$$\begin{aligned} \hat{\mathcal{N}}_i &= [p(v_1|v_i), p(v_2|v_i), \cdots, p(v_n|v_i)] \\ &= [\frac{1}{1+e^{-\mathbf{u_1}\cdot\mathbf{h_i}}}, \frac{1}{1+e^{-\mathbf{u_2}\cdot\mathbf{h_i}}}, \cdots, \frac{1}{1+e^{-\mathbf{u_n}\cdot\mathbf{h_i}}}] \end{aligned} \tag{4.7}$$

where $\mathbf{u_j}$ is a column vector of the weight matrix between the last two layers, which corresponds to a contextual vector of the neighbor v_j. We then construct the loss function as:

$$\begin{aligned} \mathcal{L}_{net} &= -\sum_{i=1}^{n} \log p(v_1, v_2, \cdots, v_n|v_i) \\ &= -\sum_{i=1}^{n}\sum_{v_j\in\mathcal{N}_i} \log p(v_j|v_i) \end{aligned} \tag{4.8}$$

Modeling jointly the network structure based on neighbors and the attribute information including attribute similarity and significant attribute associations, our

proposed method *A3embed* is trained while aiming to find optimal weight parameters in the following final objective function:

$$\arg\min_{f_1,f_2,f_3} \sum_{i=1}^{3} \lambda_i \cdot R(f_i) + \alpha \mathcal{L}_{sim} + \gamma \mathcal{L}_{ass} + \mathcal{L}_{net} \tag{4.9}$$

where f_i is a set of weight matrices and biases for each component in the deep neural network framework of *A3embed*, R is a regularization function which is defined as $R(f_i) = \frac{1}{2} \sum_{k=1}^{m_i} \|W_i^{(k)}\|_F^2$, and λ_i is a regularization term.

Optimization

Our goal is to find the optimal f_1, f_2, and f_3 that minimize the objective function formulated in Eq 4.9. We train *A3embed* using stochastic gradient descent. Specifically, we adopt RMSProp [84], an adaptive learning rate method, to update gradients during training. We omit the mathematical formulation of the partial derivative for each of the loss function because it is straightforward.

4.4 Experiments

4.4.1 Datasets

We evaluate our proposed method as well as competitors using one synthetic and three real-world attributed networks. Each of the networks contains a set of nodes forming network edges and associated attribute vectors for each node. Table 4.2 presents the statistics of the network datasets.

Synthetic Attributed Network We generated synthetic attributed networks to show the robustness of our model to diverse attribute associations, not only homophily. The networks are generated using the stochastic block model [85, 86]. We first generate several disjoint connected components, each of which corresponds to a community representing a group of nodes with the same class label, where nodes in the same community are connected to each other with p probability and nodes

Table 4.2.: Dataset Statistics

Name	Synthetic	BlogCatalog	Flickr
No. Nodes	1,024	5,196	7,575
No. Edges	varied	171,743	239,738
No. Attributes	1,000	8,189	12,047
No. Labels	varied	6	9

are connected with q probability across different communities. Every node belonging to the same community shares the same attribute vector. We then adjust the connection probabilities and perturb attribute vectors to introduce non-homophily attribute associations between nodes in the same community as well as noisy links. Embeddings for such contrived attributed networks can reveal how a model is able to capture diverse attribute associations as well as underlying node similarities from noisy connections of network nodes. See more details in Section 4.4.5.

BlogCatalog [72, 73] BlogCatalog is a blogging platform where users can form a network connecting each other. Each blog has a short description and the keywords in the description are considered as attributes. Users can assign to their blogs a category that represents a class label.

Flickr [72,73] Flickr is an online photo management and sharing website where users can establish connections to others. For attributes, we use as attributes a set of tags that describe users' specific interests in their photos. The groups to which users subscribe in the platform are considered as class labels.

4.4.2 Baselines

We evaluate the following baseline methods as well as *A3embed* for comparison. All the baselines were published recently and are known as good performers for network embedding. They are categorized into two groups. *node2vec*, *SDNE*, and *LINE*

use only the network structure information while *AANE* uses both structural and attribute information. The brief descriptions of the baselines are as follows:

node2vec [68] *node2vec*, extending *DeepWalk* [66], is one of the state-of-the art methods for network embedding and it uses only structural information. It exploits truncated random walk sequences to obtain context nodes for a given node while allowing flexibility between homophily and structural equivalence, and then computes node embeddings by maximizing the likelihood of observing context nodes.

SDNE [69] This method uses structural information only as well but focuses on the first-order and second-order proximity among nodes to preserve the network structure.

LINE [67] As in *SDNE*, *LINE* preserves the first-order and second-order proximity, but it does not model them jointly. They are considered separately to learn low-dimensional representations for each, and then concatenated. In our experiments, only the second-order proximity is used because it does not differ much from the concatenated representations, in terms of the effectiveness on downstream tasks.

AANE [73] *AANE* models and incorporates node attribute proximity into network embedding in a distributed way. It learns a low-dimensional representation based on the decomposition of attribute affinity and the embedding between connected nodes. The key difference from *A3embed* is that the node attribute information is learned in *A3embed* allowing implicit similarity and diverse relationships of attribute values whereas the node attributes have to be explicitly similar in *AANE*.

4.4.3 Experimental Setup

All experiments were conducted on a machine with Intel i5-4690K 3.50GHz CPU, 32 GB memory and GTX Titan X GPU, running 64bit Ubuntu 14.04. *A3embed* is implemented using TensorFlow 1.2.1 [1] in Python 2.7, and for the implementations of all the baselines, we use the source code from the authors.

[1] `https://www.tensorflow.org`

All the methods we evaluate include various hyperparameters that may affect the performances of the methods significantly and thus need to be tuned. We basically seek optimal hyperparameter values through grid-search and run each of the baseline algorithms with multiple epochs until we achieve the best results. Note that all the notations we use in the following discussion are ones from the original papers for the methods. For *AANE*, we use the parameter values that are already specified in the source code written by the author for each dataset ($\lambda \in \{1e-6, 0.0425\}$ and $\rho \in \{4, 5\}$). For *node2vec*, the search strategy parameters p and q are set to 2 and 0.5 respectively, and we use typical values for any other parameters such as the length of random walk ($l = 80$) and the size of network neighborhoods ($k = 10$). For *SDNE*, we also use $\alpha = 1, \beta = 5, \gamma = 5$, and the shape of its autoencoder structure is the same as ones described in its paper. All the parameters in *LINE* are set as used in the paper, except that we vary the number of samples used for optimization to find the best performance. For *A3embed*, we found that the following parameter setting works best for both BlogCatalog and Flickr: $\alpha = 1$, $\tau = 5$, $\gamma = 5$, $\zeta = -0.5$, $\omega = 0.5$, $\lambda_1 = \lambda_2 = \lambda_3 = 1$, and the learning rate is set to 0.001, For fair comparisons, the dimensionality of the embeddings is set to 200 for BlogCatalog and Flickr and 100 for synthetic networks for all the methods. In addition to $d = 200$, the impact of different embedding dimensions is discussed in Section 4.4.6.

4.4.4 Multi-Label Classification

One of the most common analytics tasks in network data is node classification, and so we evaluate the effectiveness of different network representations obtained from considered network embedding algorithms through a multi-label classification task on the real-world datasets. Every node in the network data is associated with one or more labels. Given a set of low-dimensional representations of the nodes generated by a network embedding algorithm, we randomly split them into training and test sets with varied ratios and train a classification model over the training nodes and their

Table 4.3.: Node classification performance of different methods over different training-test split ratios on BlogCatalog

Metric	Algorithm	10%	20%	30%	40%	50%	60%	70%	80%	90%
Macro F_1	LINE	0.656	0.672	0.686	0.691	0.684	0.691	0.679	0.683	0.689
	node2vec	0.514	0.541	0.609	0.624	0.635	0.641	0.639	0.649	0.655
	SDNE	0.551	0.617	0.651	0.678	0.679	0.693	0.684	0.690	0.692
	AANE	0.755	0.858	**0.884**	0.885	0.886	0.883	0.876	0.889	0.889
	A3embed	**0.837**	**0.866**	0.881	**0.888**	**0.888**	**0.894**	**0.901**	**0.912**	**0.917**
Micro F_1	LINE	0.661	0.676	0.691	0.696	0.691	0.697	0.684	0.689	0.704
	node2vec	0.521	0.545	0.614	0.631	0.642	0.648	0.646	0.657	0.669
	SDNE	0.556	0.620	0.654	0.682	0.686	0.698	0.688	0.697	0.702
	AANE	0.783	0.865	**0.889**	0.890	0.890	0.887	0.879	0.893	0.892
	A3embed	**0.841**	**0.868**	0.883	**0.891**	**0.891**	**0.897**	**0.902**	**0.913**	**0.915**

Table 4.4.: Node classification performance of different methods over different training-test split ratios on Flickr

Metric	Algorithm	10%	20%	30%	40%	50%	60%	70%	80%	90%
Macro F_1	LINE	0.576	0.601	0.604	0.610	0.617	0.621	0.627	0.626	0.624
	node2vec	0.358	0.438	0.470	0.479	0.497	0.508	0.514	0.517	0.521
	SDNE	0.506	0.559	0.582	0.592	0.600	0.609	0.609	0.610	0.609
	AANE	0.754	0.781	0.818	0.843	0.847	0.858	0.861	0.865	0.872
	A3embed	**0.816**	**0.840**	**0.849**	**0.855**	**0.856**	**0.864**	**0.865**	**0.874**	**0.890**
Micro F_1	LINE	0.585	0.608	0.611	0.619	0.626	0.630	0.638	0.639	0.640
	node2vec	0.363	0.444	0.475	0.486	0.507	0.518	0.524	0.529	0.533
	SDNE	0.508	0.562	0.586	0.597	0.607	0.617	0.617	0.617	0.620
	AANE	0.781	0.806	0.831	0.850	0.855	0.863	0.865	0.869	0.877
	A3embed	**0.819**	**0.843**	**0.852**	**0.856**	**0.860**	**0.867**	**0.868**	**0.877**	**0.894**

labels using the learned representations as features. Then, we see how accurately the models predict the labels of test nodes using Macro- and Micro-F_1 metrics. Here, for the classification model, we use a one-vs-rest support vector machine classifier provided by scikit-learn library [2].

Table 4.3 and Table 4.4 show the classification results of different methods on the two real-world attributed network, BlogCatalog and Flickr. By looking at the performance difference between structure-only based methods and joint models, it is clear that using attribute information, if available, is critical to learn better representations. Especially, *A3embed* almost consistently outperforms all the other competitive methods over different split ratios of training and test samples, which demonstrates the network representations learned from our proposed model can capture more meaningful underlying characteristics of the network nodes. Moreover, *A3embed* is very robust to even small training sample sizes. As we decreases the size of the training set, the improvement margin of *A3embed* over the baseline methods increases. This observation can tell us that our proposed method is better suited to many different real-world applications where only few nodes actually have labeles.

4.4.5 Capturing Attribute Associations

A3embed not only takes into account attribute proximity, but also diverse attribute associations between different attribute vectors, whereas existing methods work only on homophily relationship. This property implies that the representation learning of *A3embed* is more robust and generalizes to various patterns of relationships between nodes. In order to highlight how robust *A3embed* is compared to the baselines, we generate synthetic attributed networks in such a way that we can control certain properties held by network data by changing link probabilities and attribute values, as described in 4.4.1. We start with a naive network where every node in the same community is assigned an identical attribute vector ($r = 1$) and nodes are more likely

[2] `https://scikit-learn.org/`

Table 4.5.: Node classification performance on synthetic attributed networks

	$p = 0.3$, $q = 0.1$, $r = 5$		$p = 0.3$, $q = 0.3$, $r = 5$	
Algorithm	Macro F_1	Micro F_1	Macro F_1	Micro F_1
LINE	0.921	0.924	0.088	0.091
node2vec	0.911	0.912	0.061	0.068
SDNE	0.918	0.920	0.083	0.085
AANE	1.0	1.0	0.701	0.712
A3embed	1.0	1.0	1.0	1.0

to be linked with others in the same community ($p > q$). We then change q such that nodes are connected across different communities. We also randomly divide the nodes in each community into r disjoint subsets and perturb the attribute values of the nodes such that there are r different attribute vectors in the same community. In this way, we have diverse attribute associations, not only homophily relationships.

Having different values of p, q, and r, we generate various synthetic attributed networks with ten communities and predict which communities the nodes in the test set belong to by using learned low-dimensional representations. Changing the value of r does not affect the behaviors of *LINE*, *node2vec*, and *SDNE* at all because it preserves the structural proximity only without using the node attributes. It is also not surprising that both *A3embed* and *AANE* perform the classification task with high accuracy if r is low, that is, diverse attribute associations are rare. Table 4.5 shows the methods' robustness to existence of diverse attribute associations. When $p = 0.3$, $q = 0.1$, and $r = 5$, while *LINE*, *node2vec*, and *SDNE* lose some accuracy due to the noisy links, *A3embed* and *AANE* classify every node perfectly. If nodes in the same community are tightly connected with a small fraction of noisy links, then the structural proximity can be a strong signal for such nodes to stay close in the low-dimensional embedding space even if r is high. However, it does not mean *AANE* is able to capture diverse attribute associations. We discuss more details in

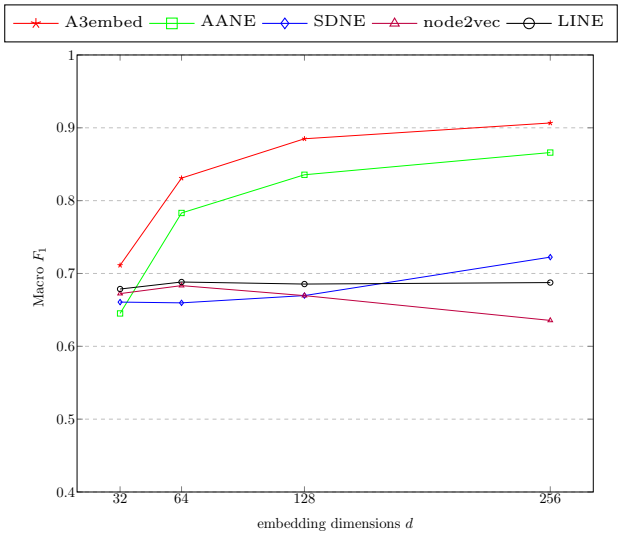
Section 4.4.7. If $p = 0.3$, $q = 0.3$, and $r = 5$, then the network structure is not helpful anymore (explaining poor performance of *LINE*, *node2vec*, and *SDNE*) and it becomes very important to be able to capture and model attribute associations. *A3embed* still achieves 100% accuracy but *AANE*'s performance gets worsen due to lack of its ability to model attribute associations.

4.4.6 Impact of Embedding Dimensions

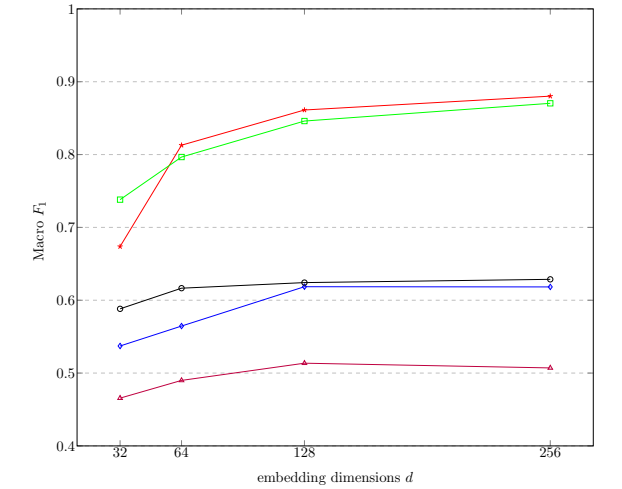
We study how the classification performance of learned representations changes with respect to varying embedding dimensions $d \in \{32, 64, 128, 256\}$. Ideally, a network embedding method is expected to be able to learn good representations regardless of the embedding dimensions. Figure 4.2 illustrates the effect of embedding dimensions on node classification with the BlogCatalog and Flickr datasets. We here report only Macro-F_1 because we observed Macro-F_1 and Micro-F_1 have almost the same trend in this experiment. As demonstrated in Figure 4.2, *A3embed* and *AANE* work better as d increases while the other methods based on only network structure saturate or deteriorate after certain number of dimensions. Since *A3embed* and *AANE* use both network structure and node attributes for joint modeling, they have greater capacity to embed latent features compared to the other three. *node2vec* goes even worse when $d = 128$ or 256, which implies overfitting.

4.4.7 Visualization

In addition to measuring effectivenesses over different downstream tasks we have discussed so far, it is also very important to visualize a network because such visualizations can help us more intuitively understand how the network nodes are distributed and interact with each other. Since different network embedding methods preserve different properties of a network, they have different ability and interpretation of node visualization. We use the synthetic networks with different parameter settings as discussed in 4.4.5 and learn new representations of nodes using *A3embed*, *AANE*,



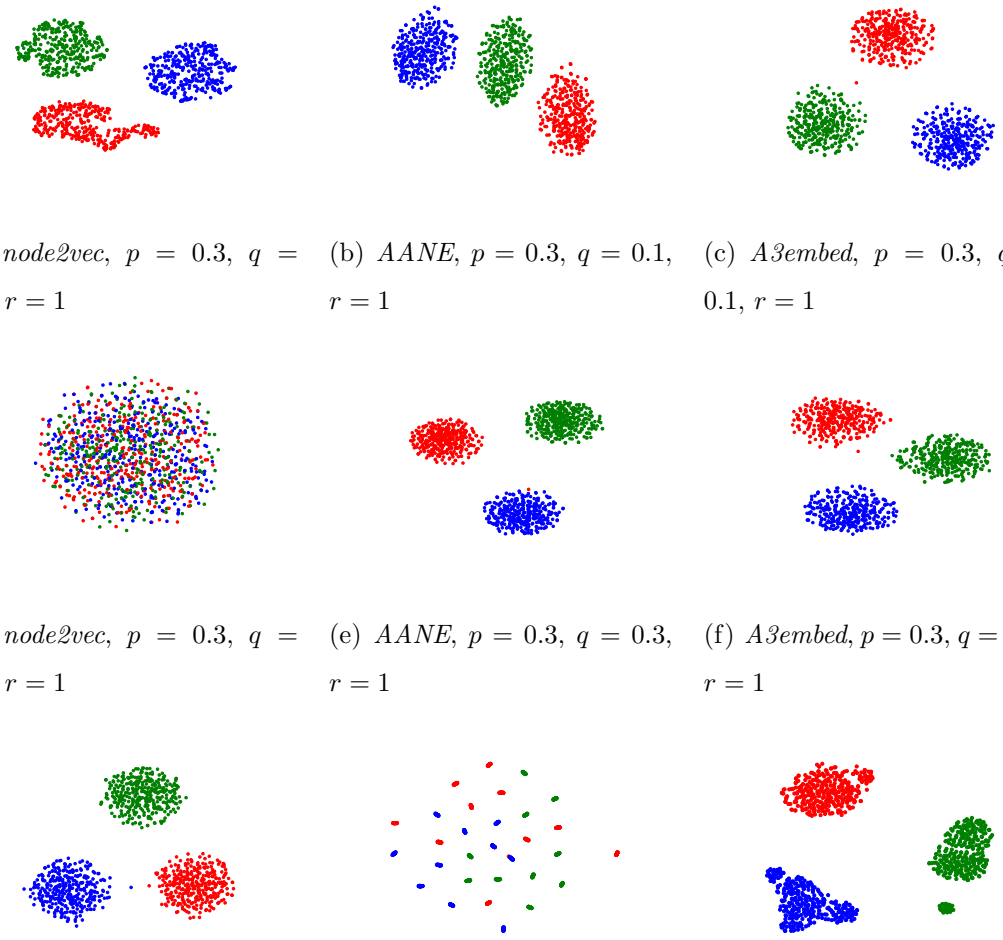
(a) BlogCatalog



(b) Flickr

Fig. 4.2.: Classification performance of learned representation over different embedding dimensions

and *node2vec*. We omit *SDNE* and *LINE* for the visualization task because they are basically not much different from *node2vec* in that all of them model only network structure. The learned representations are used as input to *t-SNE* [87] with its default parameter values.



(a) *node2vec*, $p = 0.3$, $q = 0.1$, $r = 1$

(b) *AANE*, $p = 0.3$, $q = 0.1$, $r = 1$

(c) *A3embed*, $p = 0.3$, $q = 0.1$, $r = 1$

(d) *node2vec*, $p = 0.3$, $q = 0.3$, $r = 1$

(e) *AANE*, $p = 0.3$, $q = 0.3$, $r = 1$

(f) *A3embed*, $p = 0.3$, $q = 0.3$, $r = 1$

(g) *node2vec*, $p = 0.3$, $q = 0.1$, $r = 10$

(h) *AANE*, $p = 0.3$, $q = 0.1$, $r = 10$

(i) *A3embed*, $p = 0.3$, $q = 0.1$, $r = 10$

Fig. 4.3.: Visualization of synthetic attributed networks. Color of a point indicates its community. (p: in-community link probability, q: cross-community link probability, r: number of distinct attribute vectors in a community)

Figure 4.3 illustrates visualization of low-dimensional representations of various synthetic attributed networks. The synthetic attributed networks are built with different parameter settings (p, q, and r) and all of them include three communities, each of which is indicated by a color. When $p = 0.3$, $q = 0.1$, and $r = 1$, all the methods

produce nice visualization where every community is well-separated (Figure 4.3(a), 4.3(b), and 4.3(c)). However, if we have large numbers of noisy links (those that cross different communities), it must be critical to benefit from modeling node attributes. While *A3embed* and *AANE* can visualize the network in perfect shape (Figure 4.3(e) and 4.3(f)), *node2vec* fails to visualize the nodes correctly due to the absence of any clues to differentiate the in-community and cross-community links in the network (Figure 4.3(d)). If the number of distinct attribute vectors in each community increases ($r = 10$) and thus there are diverse attribute associations, *AANE* fails to keep every node in a community close to each other (Figure 4.3(h)). It makes r disjoint groups for each community in its visualization because *AANE* optimizes its objective based on only homophily relationship. In contrast, *A3embed* captures the attribute associations between even different attribute vectors and the nodes in the same community are better clustered together than *AANE* (Figure 4.3(i)). *node2vec* is not affected by changing r due to its inability to model node attributes (Figure 4.3(g)).

4.5 Summary

In this chapter, we propose a novel network embedding method, called *A3embed*, for attributed networks. *A3embed* learns new network representations by jointly exploiting network structural information and node attribute values. While preserving the network structure, it also uses various attribute associations, not limited to homophily relationships, among nodes. The existence of non-homophily but significant attribute associations in networks can play an important role for finding well-represented embeddings. The experiments are conducted on two real-world attributed networks to demonstrate the effectiveness of *A3embed* in some downstream tasks such as multi-label classification and visualization. We also use synthetic attributed networks to show how well *A3embed* is able to capture diverse attribute associations. The experimental results show that our proposed method outperforms other network

embedding methods in different downstream machine learning tasks, which confirms the importance of using attribute associations in representation learning.

5. CONCLUSION

In this dissertation, we addressed various data mining problems that arise in attributed graphs with a focus on how to exploit attribute values as well as graph structure to improve the effectiveness of the solutions. The proposed solutions are relevant and effective in numerous real-world applications where relational entities possess attribute values representing their properties and providing a rich source of valuable information.

First of all, in Chapter 2, we introduced the notion of attribute associations in attributed graphs and explored a novel algorithm that extracts statistically significant attribute associations efficiently and effectively. The information we can obtain from such statistically significant attribute associations can provide us with insights that help us better understand underlying patterns in attributed graphs, and no other existing works are able to perform the same task. The qualitative analysis on the experimental results demonstrates the effectiveness of our proposed algorithms for the task and we also showed that the algorithms scales well to large attributed graphs. The statistically significant attribute associations are also shown to be helpful for the link prediction problem.

We also proposed a probabilistic generative model for attributed graphs to solve the problem of community detection. Detection of hidden communities in graphs is widely used in various applications such as clustering users together in order to increase the efficacy of predictive models, estimating unknown features of users/entities in social networks, detecting networks of fraudulent/rogue websites, and so on. The generative model was built based on the assumption that both node connections and attributes are generated from communities. To estimate parameters of the model, we applied an inference algorithm based on Gibbs sampling which is similar to Latent

Dirichlet Allocation. Our proposed model enables us to perform community detection as well as profiling missing attribute values due to the nature of generative model.

Finally, we studied the utility of attribute association patterns to the task of representation learning for nodes in attributed graphs. Although there are plenty of existing algorithms that learn node embeddings by considering graph structure and/or node attribute values, none of them considered the notion of attribute associations we introduced in Chapter 2. We adopted deep neural networks to perform joint learning of graph structure and node attributes in an unsupervised setting. We established that our model was able to capture both attribute associations and node proximity during the process of learning node embedding vectors. We evaluated our proposed model, A3embed, on real-world attributed graphs based on node classification task and visualization. The experiment results showed that A3embed outperforms state-of-the-art models and it is able to effectively capture diverse patterns of attribute associations.

To sum up, this dissertation presents various approaches to practical uses of node attributed values to solve different data mining problems in attributed graphs. Compared to previous methods, our proposed solutions are shown to achieve better performances for all the considered tasks.

REFERENCES

REFERENCES

[1] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the internet topology," in *ACM SIGCOMM computer communication review*, vol. 29, no. 4. ACM, 1999, pp. 251–262.

[2] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener, "Graph structure in the web," *Computer networks*, vol. 33, no. 1, pp. 309–320, 2000.

[3] R. Albert, H. Jeong, and A.-L. Barabási, "Internet: Diameter of the world-wide web," *Nature*, vol. 401, no. 6749, pp. 130–131, 1999.

[4] A. McGovern, L. Friedland, M. Hay, B. Gallagher, A. Fast, J. Neville, and D. Jensen, "Exploiting relational structure to understand publication patterns in high-energy physics," *ACM SIGKDD Explorations Newsletter*, vol. 5, no. 2, pp. 165–172, 2003.

[5] M. E. Newman, "The structure of scientific collaboration networks," *Proceedings of the National Academy of Sciences*, vol. 98, no. 2, pp. 404–409, 2001.

[6] R. Pastor-Satorras and A. Vespignani, "Epidemic spreading in scale-free networks," *Physical review letters*, vol. 86, no. 14, p. 3200, 2001.

[7] C. Moore and M. E. Newman, "Epidemics and percolation in small-world networks," *Physical Review E*, vol. 61, no. 5, p. 5678, 2000.

[8] R. M. May and A. L. Lloyd, "Infection dynamics on scale-free networks," *Physical Review E*, vol. 64, no. 6, p. 066112, 2001.

[9] A. Kleczkowski and B. T. Grenfell, "Mean-field-type equations for spread of epidemics: The small worldmodel," *Physica A: Statistical Mechanics and its Applications*, vol. 274, no. 1, pp. 355–360, 1999.

[10] R. Rossi and J. Neville, "Modeling the evolution of discussion topics and communication to improve relational classification," in *Proceedings of the First Workshop on Social Media Analytics*. ACM, 2010, pp. 89–97.

[11] H. Jeong, B. Tombor, R. Albert, Z. N. Oltvai, and A.-L. Barabási, "The large-scale organization of metabolic networks," *Nature*, vol. 407, no. 6804, pp. 651–654, 2000.

[12] A. Wagner and D. A. Fell, "The small world inside large metabolic networks," *Proceedings of the Royal Society of London B: Biological Sciences*, vol. 268, no. 1478, pp. 1803–1810, 2001.

[13] J. A. Dunne, R. J. Williams, and N. D. Martinez, "Food-web structure and network theory: the role of connectance and size," *Proceedings of the National Academy of Sciences*, vol. 99, no. 20, pp. 12 917–12 922, 2002.

[14] J. Camacho, R. Guimerà, and L. A. N. Amaral, "Robust patterns in food web structure," *Physical Review Letters*, vol. 88, no. 22, p. 228102, 2002.

[15] S. Maslov and K. Sneppen, "Specificity and stability in topology of protein networks," *Science*, vol. 296, no. 5569, pp. 910–913, 2002.

[16] H. Jeong, S. P. Mason, A.-L. Barabási, and Z. N. Oltvai, "Lethality and centrality in protein networks," *Nature*, vol. 411, no. 6833, pp. 41–42, 2001.

[17] J. Neville, Ö. Şimşek, D. Jensen, J. Komoroske, K. Palmer, and H. Goldberg, "Using relational knowledge discovery to prevent securities fraud," in *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM, 2005, pp. 449–458.

[18] V. E. Krebs, "Mapping networks of terrorist cells," *Connections*, vol. 24, no. 3, pp. 43–52, 2002.

[19] M. Kim and J. Leskovec, "Multiplicative attribute graph model of real-world networks," *Internet Mathematics*, vol. 8, no. 1-2, pp. 113–160, 2012.

[20] E. M. Rogers and D. K. Bhowmik, "Homophily-heterophily: Relational concepts for communication research," *Public opinion quarterly*, vol. 34, no. 4, pp. 523–538, 1970.

[21] M. McPherson, L. Smith-Lovin, and J. M. Cook, "Birds of a feather: Homophily in social networks," *Annual review of sociology*, pp. 415–444, 2001.

[22] R. Li, C. Wang, and K. C.-C. Chang, "User profiling in an ego network: co-profiling attributes and relationships," in *Proceedings of the 23rd international conference on World wide web*. ACM, 2014, pp. 819–830.

[23] R. Balasubramanyan and W. W. Cohen, "Block-lda: Jointly modeling entity-annotated text and entity-entity links." in *SDM*, vol. 11. SIAM, 2011, pp. 450–461.

[24] S. Günnemann, B. Boden, I. Färber, and T. Seidl, "Efficient mining of combined subspace and subgraph clusters in graphs with feature vectors," in *Advances in Knowledge Discovery and Data Mining*. Springer, 2013, pp. 261–275.

[25] S. Gunnemann, P. Dao, M. Jamali, and M. Ester, "Assessing the significance of data mining results on graphs with feature vectors," in *2012 IEEE 12th International Conference on Data Mining (ICDM)*. IEEE, 2012, pp. 270–279.

[26] J. Yang, J. McAuley, and J. Leskovec, "Community detection in networks with node attributes," in *ICDM*. IEEE, 2013, pp. 1151–1156.

[27] M. Kim and J. Leskovec, "Latent multi-group membership graph model," *ICML*, 2012.

[28] Y. Zhou, H. Cheng, and J. X. Yu, "Clustering large attributed graphs: An efficient incremental approach," in *ICDM*. IEEE, 2010, pp. 689–698.

[29] H. Cheng, Y. Zhou, and J. X. Yu, "Clustering large attributed graphs: A balance between structural and attribute similarities," *TKDD*, vol. 5, no. 2, p. 12, 2011.

[30] Y. Zhou, H. Cheng, and J. X. Yu, "Graph clustering based on structural/attribute similarities," *VLDB*, vol. 2, no. 1, pp. 718–729, 2009.

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

[32] J. Scott, *Social network analysis*. Sage, 2012.

[33] J. Hu, X. Shen, Y. Shao, C. Bystroff, and M. J. Zaki, "Mining protein contact maps." in *BIOKDD*, 2002, pp. 3–10.

[34] J. M. Kleinberg, R. Kumar, P. Raghavan, S. Rajagopalan, and A. S. Tomkins, "The web as a graph: measurements, models, and methods," in *Computing and combinatorics*. Springer, 1999, pp. 1–17.

[35] J. Pei, D. Jiang, and A. Zhang, "Mining cross-graph quasi-cliques in gene expression and protein interaction data," in *21st International Conference on Data Engineering (ICDE'05)*. IEEE, 2005, pp. 353–356.

[36] H. He and A. K. Singh, "Graphrank: Statistical modeling and mining of significant subgraphs in the feature space," in *ICDM'06. Sixth International Conference on Data Mining, 2006*. IEEE, 2006, pp. 885–890.

[37] S. Ranu and A. K. Singh, "Graphsig: A scalable approach to mining significant subgraphs in large graph databases," in *2009 IEEE 25th International Conference on Data Engineering*. IEEE, 2009, pp. 844–855.

[38] X. Yan, H. Cheng, J. Han, and P. S. Yu, "Mining significant graph patterns by leap search," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. ACM, 2008, pp. 433–444.

[39] J. Scott, T. Ideker, R. M. Karp, and R. Sharan, "Efficient algorithms for detecting signaling pathways in protein interaction networks," *Journal of Computational Biology*, vol. 13, no. 2, pp. 133–144, 2006.

[40] Y. Chi, Y. Yang, and R. R. Muntz, "Indexing and mining free trees," in *Third IEEE International Conference on Data Mining, 2003. ICDM 2003*. IEEE, 2003, pp. 509–512.

[41] W. Hamalainen and M. Nykanen, "Efficient discovery of statistically significant association rules," in *ICDM'08. IEEE International Conference on Data Mining, 2008*. IEEE, 2008, pp. 203–212.

[42] A. Arora, M. Sachan, and A. Bhattacharya, "Mining statistically significant connected subgraphs in vertex labeled graphs," in *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. ACM, 2014, pp. 1003–1014.

[43] M. Sachan and A. Bhattacharya, "Mining statistically significant substrings using the chi-square statistic," *Proceedings of the VLDB Endowment*, vol. 5, no. 10, pp. 1052–1063, 2012.

[44] C. Low-Kam, C. Raïssi, M. Kaytoue, and J. Pei, "Mining statistically significant sequential patterns," in *2013 IEEE 13th International Conference on Data Mining (ICDM)*. IEEE, 2013, pp. 488–497.

[45] M. Coscia, G. Rossetti, F. Giannotti, and D. Pedreschi, "Demon: a local-first discovery method for overlapping communities," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2012, pp. 615–623.

[46] J. Yang and J. Leskovec, "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.

[47] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, p. P10008, 2008.

[48] Y. Ruan, D. Fuhry, and S. Parthasarathy, "Efficient community detection in large networks using content and links," in *Proceedings of the 22nd international conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2013, pp. 1089–1098.

[49] M. Gupte and T. Eliassi-Rad, "Measuring tie strength in implicit social networks," in *Proceedings of the 4th Annual ACM Web Science Conference*. ACM, 2012, pp. 109–118.

[50] W. H. Greene, *Econometric analysis*. Pearson Education India, 2003.

[51] S. Nagaev and V. Chebotarev, "On the bound of proximity of the binomial distribution to the normal one," *Theory of Probability & Its Applications*, vol. 56, no. 2, pp. 213–239, 2012.

[52] "DBLP: computer science bibliography," http://dblp.uni-trier.de/xml/, April 2016.

[53] "Yelp dataset challenge," https://www.yelp.com/, July 2014.

[54] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *Journal of the American society for information science and technology*, vol. 58, no. 7, pp. 1019–1031, 2007.

[55] S. Fortunato, "Community detection in graphs," *Physics reports*, vol. 486, no. 3-5, pp. 75–174, 2010.

[56] M. Rosvall and C. T. Bergstrom, "Maps of random walks on complex networks reveal community structure," *Proceedings of the National Academy of Sciences*, vol. 105, no. 4, pp. 1118–1123, 2008.

[57] A. Clauset, M. E. Newman, and C. Moore, "Finding community structure in very large networks," *Physical review E*, vol. 70, no. 6, p. 066111, 2004.

[58] M. Girvan and M. E. Newman, "Community structure in social and biological networks," *Proceedings of the national academy of sciences*, vol. 99, no. 12, pp. 7821–7826, 2002.

[59] A. P. Streich, M. Frank, D. Basin, and J. M. Buhmann, "Multi-assignment clustering for boolean data," in *ICML*. ACM, 2009, pp. 969–976.

[60] J. Chang and D. M. Blei, "Relational topic models for document networks," in *International conference on artificial intelligence and statistics*, 2009, pp. 81–88.

[61] Y. Liu, A. Niculescu-Mizil, and W. Gryc, "Topic-link lda: joint models of topic and author community," in *proceedings of the 26th annual international conference on machine learning*. ACM, 2009, pp. 665–672.

[62] J. J. McAuley and J. Leskovec, "Learning to discover social circles in ego networks." in *NIPS*, vol. 2012, 2012, pp. 548–56.

[63] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *the Journal of machine Learning research*, vol. 3, pp. 993–1022, 2003.

[64] S. A. Macskassy and F. Provost, "A simple relational classifier," NEW YORK UNIV NY STERN SCHOOL OF BUSINESS, Tech. Rep., 2003.

[65] M. Pennacchiotti and A.-M. Popescu, "Democrats, republicans and starbucks afficionados: user classification in twitter," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2011, pp. 430–438.

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

[67] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2015, pp. 1067–1077.

[68] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: ACM, 2016, pp. 855–864. [Online]. Available: http://doi.acm.org/10.1145/2939672.2939754

[69] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: ACM, 2016, pp. 1225–1234. [Online]. Available: http://doi.acm.org/10.1145/2939672.2939753

[70] S. Pan, J. Wu, X. Zhu, C. Zhang, and Y. Wang, "Tri-party deep network representation," in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, ser. IJCAI'16. AAAI Press, 2016, pp. 1895–1901. [Online]. Available: http://dl.acm.org/citation.cfm?id=3060832.3060886

[71] S. Cao, W. Lu, and Q. Xu, "Deep neural networks for learning graph representations," 2016.

[72] X. Huang, J. Li, and X. Hu, "Label informed attributed network embedding," in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. ACM, 2017, pp. 731–739.

[73] ——, "Accelerated attributed network embedding," in *Proceedings of the 2017 SIAM International Conference on Data Mining*. SIAM, 2017, pp. 633–641.

[74] J. Lee, K. Park, and S. Prabhakar, "Mining statistically significant attribute associations in attributed graphs," in *IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 2016, pp. 991–996.

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

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

[77] Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents." in *ICML*, vol. 14, 2014, pp. 1188–1196.

[78] P. Yanardag and S. Vishwanathan, "Deep graph kernels," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 1365–1374.

[79] A. Tomar, F. Godin, B. Vandersmissen, W. De Neve, and R. Van de Walle, "Towards twitter hashtag recommendation using distributed word representations and a deep feed forward neural network," in *International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2014*. IEEE, 2014, pp. 362–368.

[80] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Y. Chang, "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.

[81] S. Pan, J. Wu, X. Zhu, C. Zhang, and Y. Wang, "Tri-party deep network representation," in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July2016*, 2016, pp. 1895–1901. [Online]. Available: http://www.ijcai.org/Abstract/16/271

[82] R. Salakhutdinov and G. Hinton, "Semantic hashing," *International Journal of Approximate Reasoning*, vol. 50, no. 7, pp. 969–978, 2009.

[83] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[84] T. Tieleman and G. Hinton, "Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude," COURSERA: Neural Networks for Machine Learning, 2012.

[85] Y. J. Wang and G. Y. Wong, "Stochastic blockmodels for directed graphs," *Journal of the American Statistical Association*, vol. 82, no. 397, pp. 8–19, 1987.

[86] K. Nowicki and T. A. B. Snijders, "Estimation and prediction for stochastic blockstructures," *Journal of the American Statistical Association*, vol. 96, no. 455, pp. 1077–1087, 2001.

[87] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

VITA

VITA

Jihwan Lee obtained B.S degree from the Department of Computer Engineering in Kyungpook National University, South Korea, in 2008. In 2009, He spent about six months doing Database and Data Mining research as a research intern under the guidance of Prof. Wook-Shin Han who is a professor at POSTECH now. He joined the Department of Computer Science in University of California Los Angeles and was actively involved with various research projects of Data Stream Mining under the direction of Prof. Carlo Zaniolo, and obtained M.S degree in 2011. After that he decided to pursue Ph.D degree and joined the Department of Computer Science in Purdue University. His research areas during the doctoral program includes graph data mining, machine learning, and statistical relational learning. He did two summer internships at Hewrett Packard Enterprise and Amazon in 2015 and 2016, respectively, and he has been working as Applied Scientist in Amazon Alexa AI while keeping pursuing his Ph.D degree.