Local Modeling of Attributed Graphs: Algorithms and Applications

A Dissertation presented

by

Bryan Perozzi

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

in

Computer Science

Stony Brook University

May 2016

Stony Brook University

The Graduate School

Bryan Perozzi

We, the dissertation committe for the above candidate for the

Doctor of Philosophy degree, hereby recommend

acceptance of this dissertation

Steven Skiena - Dissertation Advisor
Distinguished Teaching Professor, Computer Science

Leman Akoglu - Chairperson of Defense
Assistant Professor, Computer Science

Jie Gao
Associate Professor, Computer Science

Vahab Mirrokni
Principal Research Scientist, Google, Inc.

This dissertation is accepted by the Graduate School

Charles Taber
Dean of the Graduate School

Abstract of the Dissertation

Local Modeling of Attributed Graphs: Algorithms and Applications

by

Bryan Perozzi

Doctor of Philosophy

in

Computer Science

Stony Brook University

2016

It is increasingly common to encounter real-world graphs which have attributes associated with the nodes, in addition to their raw connectivity information. For example, social networks contain both the friendship relations as well as user attributes such as interests and demographics. A protein-protein interaction network may not only have the interaction relations but the expression levels associated with the proteins. Such information can be described by a graph in which nodes represent the objects, edges represent the relations between them, and feature vectors associated with the nodes represent the attributes. This graph data is often referred to as an attributed graph.

This thesis focuses on developing scalable algorithms and models for attributed graphs. This data can be viewed as either discrete (set of edges), or continuous (distances between embedded nodes), and I examine the issue from both sides. Specifically, I present an online learning algorithm which utilizes recent advances in *deep learning* to create rich graph embeddings. The multiple scales of social relationships encoded by this novel approach are useful for multi-label classification and regression tasks in networks. I also present local algorithms for anomalous community scoring in discrete graphs. These algorithms discover subsets of the graph's attributes which cause communities to form (e.g. shared interests on a social network).

The scalability of all the methods in this thesis is ensured by building from a restricted set of graph primitives, such as ego-networks and truncated random walks, which exploit the *local* information around each vertex. In addition, limiting the scope of graph dependencies we consider enables my approaches to be trivially parallelized using commodity tools for big data processing, like MapReduce or Spark.

The applications of this work are broad and far reaching across the fields of data mining and information retrieval, including user profiling/demographic inference, online advertising, and fraud detection.

Dedicated to my mother

"Computers are the future."

Contents

List of Figures

List of Tables

Acknowledgements

The PhD is a marathon, not a sprint. In the course of running it, I was helped by many individuals along the way. I would like to take a moment to thank the individuals who have helped me along the way:

First, I would like to thank my advisor Steven Skiena. Through a refined blend of high standards, skepticism, and intellectual freedom he guided me to a more successful graduate school existence than any of us expected. I hope to replicate his success should I mentor others in my career.

I was blessed with other great collaborators while at Stony Brook. I would like to thank Leman Akoglu for great technical discussions and the mentoring (especially w.r.t. paper writing) which went far beyond the call of duty. I would like to thank Rami Al-Rfou for all the late nights spent pair programming and pushing for deadlines. (It was fun riding the representation learning wave with you, Rami!) I would also like to thank all the other collaborators from our lab (Yanqing, Vivek, Yingtao, Haochen, and many others). I would also like to thank the many mentors I had during my summer internships (including Mary, Jon, Matt, Michael, Hila, Mayur, Hao, and Tasos).

I would also like to thank the funding agencies which supported this work. Specifically, I thank the Institute for Advanced Computational Science at Stony Brook (IACS) and its director, Robert Harrison. The IACS Junior Researcher Fellowship allowed me to participate at a much larger number of academic venues than otherwise would have been possible. I must also thank the National Science Foundation, and Google for a faculty research award to Steve.

As a social network researcher, it would be amiss for me to not also thank the rest of my network. I thank Aleks for always having a second opinion, the Stony Brook Soirée (Connor, Sean, Rishab, Sam, Shikha, Amogh, and others) for conversation, the League (Chris, Spence, JT, Jason, Tom) for their career advice, and the Hammers (Lee, Mike, Steven, Ann, Becky) for their unwavering friendship. As a catch all clause for people who were accidentally omitted – thanks to anyone who gave me a meal, wrote me a code review, or let me sleep on their couch.

Last but certainly not least, I thank my wife Christy, without whom none of this would have been possible.

Publications

1. Bryan Perozzi, Vivek Kulkarni, and Steven Skiena. Walklets: Multiscale graph embeddings for interpretable network classification. In *(submitted)*, 2016

2. Yingtao Tian, Vivek Kulkarni, Bryan Perozzi, and Steven Skiena. On the convergent properties of word embedding methods. In *(submitted)*, 2016

3. Yingtao Tian, Bryan Perozzi, Vivek Kulkarni, and Steven Skiena. Piecewise linear embedding alignment for multilingual learning. In *(submitted)*, 2016

4. Bryan Perozzi, Michael Schueppert, Jack Saalweachter, and Mayur Thakur. When recommendation goes wrong - Anomalous Link Discovery in Recommendation Networks. In *Proceedings of the 22th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, 2016

5. Vivek Kulkarni, Bryan Perozzi, and Steven Skiena. Freshman or fresher? Quantifying the Geographic Variation of Internet Language. In *10th International AAAI Conference on Web and Social Media*, ICWSM'16, 2016

6. Bryan Perozzi and Leman Akoglu. Scalable anomaly ranking of attributed neighborhoods. In *Proceedings of 2016 SIAM International Conference on Data Mining*, SDM '16, 2016

7. Yanqing Chen, Bryan Perozzi, and Steven Skiena. Vector-based similarity measurements for historical figures. In *Similarity Search and Applications*, pages 179–190. Springer International Publishing, 2015

8. Vivek Kulkarni, Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. Statistically significant detection of linguistic change. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, New York, NY, USA, 2015. ACM

9. Bryan Perozzi and Steven Skiena. Exact age prediction in social networks. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, 2015

10. Rami Al-Rfou, Vivek Kulkarni, Bryan Perozzi, and Steven Skiena. Polyglot-ner: Massive multilingual named entity recognition. In *Proceedings of 2015 SIAM International Conference on Data Mining*, SDM '15, 2015

11. Bryan Perozzi, Christopher McCubbin, and J.T. Halbert. Scalable graph clustering with parallel approximate pagerank. *Social Network Analysis and Mining*, 4(1):179, 2014. ISSN 1869-5450

12. Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 701–710, New York, NY, USA, 2014. ACM

13. Bryan Perozzi, Leman Akoglu, Patricia Iglesias Sánchez, and Emmanuel Müller. Focused clustering and outlier detection in large attributed graphs. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 1346–1355, New York, NY, USA, 2014. ACM

14. Bryan Perozzi, Rami Al-Rfou, Vivek Kulkarni, and Steven Skiena. Inducing language networks from continuous space word representations. In *Complex Networks V*, volume 549 of *Studies in Computational Intelligence*, pages 261–273. 2014

15. Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. Polyglot: Distributed word representations for multilingual nlp. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 183–192, Sofia, Bulgaria, August 2013. ACL

16. Yanqing Chen, Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. The expressive power of word embeddings. In *ICML 2013 Workshop on Deep Learning for Audio, Speech, and Language Processing*, Atlanta, GA, USA, July 2013

17. Bryan Perozzi, Christopher McCubbin, Spencer Beecher, and J.T. Halbert. Scalable graph clustering with pregel. In *Complex Networks IV*, volume 476 of *Studies in Computational Intelligence*, pages 133–144. 2013

18. C. McCubbin, B. Perozzi, A. Levine, and A. Rahman. Finding the 'needle': Locating interesting nodes using the k-shortest paths algorithm in mapreduce. In *Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on*, pages 180–187, Dec 2011

Chapter 1

Introduction

This thesis is concerned with the scalable modeling of attributed graphs. In a large attributed network such as the Netflix user/movie graph or the Facebook social graph, how can we use a local view of the graph structure to predict node attributes? Do homogeneous communities occur? How can we efficiently find the relevant subspaces which cause them? Can we query these communities directly? Which structural links are anomalous and shouldn't be trusted?

Network data can be viewed with either a discrete (set of edges) or continuous (distances between embedded nodes) representation. In this thesis, I examine the issue from both sides. Specifically, I present scalable online learning algorithms which create useful graph embeddings. I also develop local algorithms for anomalous community scoring in discrete graphs.

In the course of evaluation, I study many different networks, including word co-occurrence graphs from Wikipedia, product co-purchase graphs from Amazon, collaboration networks such as Citeseer and DBLP, and many social networks such as Facebook, Twitter, Google+, YouTube, LastFM, and Flickr. I show the utility of my proposed methods through a number of real world applications including multi-label classification, user demographic inference, and anomalous community mining.

1.1 The Local Modeling Philosophy

Let G be a graph, $G = (V, E4)$, where V represent the vertices of the network, and E the set of connections between them, $E \subseteq (V \times V)$. In many graphs of practical concern, the cardinalities of V and E can be incredibly large, with $|V| \sim 1,000,000,000$ and $|E| \sim 100,000,000,000$. Figure 1.1 presents the sizes of several such real world graphs. We note that each of these graphs drive commercial products with billions of dollars in annual revenue.

Graph	Number of Vertices, $\|V\|$	Number of Edges, $\|E\|$	Notes
Facebook, 2011	721,000,000	68,200,000,000	From [140]
Facebook, 2016	1,590,000,000	—	Monthly Active Users, December 2015[1]
Amazon, 2015	480,000,000	—	US Product Co-purchase Graph [2]
Internet, 2016	4,630,000,000	—	Indexed Web Graph[3]

Table 1.1: Some graphs of commercial importance

To cope with the massive size of these graphs, I propose approaches based on *local* modeling throughout this thesis. More of a philosophy (or perhaps a heuristic) than a hard rule, I believe that the only successful approach possible when dealing with graphs comprising such 'big data' is to purposely limit the scope of the relationship which you are willing to consider. By purposely analyzing less data per vertex, one can thereby do more.

To ensure these local properties of methods which I present in this thesis, I utilize a restricted set of graph primitives – truncated random walks and ego networks.

1.2 Thesis Overview

This thesis focuses on my work in local algorithms for graph embedding and anomaly detection in attributed graphs. I begin with Chapter 2, where I introduce my method for deep learning for social networks. This method, DeepWalk, uses short random walks to learn a mapping function Φ which uses a small number (d) of latent dimensions to describe each vertex v in a graph, $\Phi(v) \in \mathbb{R}^{|V| \times d}$. These representations are useful for multi-label classification tasks. Next, in Chapter 3, I describe a modification of DeepWalk's sampling procedure to produce representations which capture multiple scales of community information. I then present a method to detect groups of attributes (a subspace) which causes observed community structure to form in Chapter 4. This method allows efficient optimization, and can scale to large communities. I follow this in Chapter 5 with a method to allow user focused querying of attributed graphs, with an integrated anomaly detection. Finally, I present some observations about the nearest neighbor graphs of these embedding spaces in Chapter 6, and close with conclusions and future work in Chapter 7. I briefly describe these chapters further below:

DeepWalk - Online Learning of Social Representations [117] We present DEEPWALK, a novel approach for learning latent representations of vertices in a network. These latent representations encode social relations in a continuous vector space, which is easily exploited by statistical models. DEEPWALK generalizes recent advancements in language modeling and unsupervised feature learning (or *deep learning*) from sequences of words to graphs.

DEEPWALK uses local information obtained from truncated random walks to *learn* latent representations by treating walks as the equivalent of sentences. We demonstrate DEEPWALK's latent representations on several multi-label network classification tasks for social networks such as BlogCatalog, Flickr, and YouTube. Our results show that DEEPWALK outperforms challenging baselines which are allowed a global view of the network, especially in the presence of missing information. DEEPWALK's representations can provide F_1 scores up to 10% higher than competing methods when labeled data is sparse. In some experiments, DEEPWALK's representations are able to outperform all baseline methods while using 60% less training data.

DEEPWALK is also scalable. It is an online learning algorithm which builds useful incremental results, and is trivially parallelizable. These qualities make it suitable for a broad class of real world applications such as network classification, and anomaly detection.

Walklets: Multiscale Graph Embeddings for Interpretable Network Classification [119] We present Multiscale Network Embeddings (WALKLETS), a novel approach for learning multiscale

representations of vertices in a network. These representations clearly encode multiscale vertex relationships in a continuous vector space suitable for multi-label classification problems. Unlike previous work, the latent features generated using WALKLETS are analytically derivable, and human interpretable.

WALKLETS uses the offsets between vertices observed in a random walk to learn a series of latent representations, each which captures successively larger relationships. This variety of dependency information allows the same representation strategy to model phenomenon which occur at different scales.

We demonstrate WALKLETS's latent representations on several multi-label network classification tasks for social networks such as BlogCatalog, Flickr, and YouTube. Our results show that WALKLETS outperforms new methods based on neural matrix factorization, and can scale to graphs with millions of vertices and edges.

AMEN [111] Given a graph with node attributes, what neighborhoods[4] are anomalous? To answer this question, one needs a quality score that utilizes both structure and attributes. Popular existing measures either quantify the structure only and ignore the attributes (e.g., conductance), or only consider the connectedness of the nodes inside the neighborhood and ignore the cross-edges at the boundary (e.g., density).

In this work we propose `normality`, a new quality measure for attributed neighborhoods. `Normality` utilizes structure and attributes *together* to quantify both internal consistency and external separability. It exhibits two key advantages over other measures: (1) It allows many boundary-edges as long as they can be "exonerated"; i.e., either (i) are expected under a null model, and/or (ii) the boundary nodes do not exhibit the subset of attributes shared by the neighborhood members. Existing measures, in contrast, penalize boundary edges irrespectively. (2) `Normality` can be efficiently maximized to automatically infer the shared attribute subspace (and respective weights) that characterize a neighborhood. This efficient optimization allows us to process graphs with millions of attributes.

We capitalize on our measure to present a novel approach for Anomaly Mining of Entity Neighborhoods (AMEN). Experiments on real-world attributed graphs illustrate the effectiveness of our measure at anomaly detection, outperforming popular approaches including conductance, density, OddBall, and SODA. In addition to anomaly detection, our qualitative analysis demonstrates the utility of `normality` as a powerful tool to contrast the correlation between structure and attributes across different graphs.

Focused Clustering [114] : Graph clustering and graph outlier detection have been studied extensively on plain graphs, with various applications. Recently, algorithms have been extended to graphs with attributes as often observed in the real-world. However, all of these techniques fail to incorporate the user preference into graph mining, and thus, lack the ability to steer algorithms to more interesting parts of the attributed graph.

[4]A neighborhood is used as a general term throughout text and refers to any connected subgraph; such as a cluster, community, social circle, ego network, etc.

In this work, we overcome this limitation and introduce a novel user-oriented approach for mining attributed graphs. The key aspect of our approach is to infer user preference by the so-called focus attributes through a set of user-provided exemplar nodes. In this new problem setting, clusters and outliers are then simultaneously mined according to this user preference. Specifically, our FOCUSCO algorithm identifies the focus, extracts focused clusters and detects outliers. Moreover, FOCUSCO scales well with graph size, since we perform a local clustering of interest to the user rather than global partitioning of the entire graph. We show the effectiveness and scalability of our method on synthetic and real-world graphs, as compared to both existing graph clustering and outlier detection approaches.

Language Networks [115] Recent advancements in unsupervised feature learning have developed powerful latent representations of words. Understanding the structure of latent spaces attained is key to any future advancement in unsupervised learning.

In this work, we introduce a new view of continuous space word representations as language networks. We explore two techniques to create language networks from learned features by inducing them for two popular word representation methods and examining the properties of their resulting networks. We find that the induced networks differ from other methods of creating language networks, and that they contain meaningful community structure.

1.3 Additional Work

I have also worked on a number of additional projects, omitted for the sake of brevity:

My work in computational social science has developed a method for tracking linguistic shift through time utilizing neural network matrix factorization [71]. This statistically sound method is capable of tracking linguistic change across years of micro-blogging using Twitter, a decade of product reviews using a corpus of movie reviews from Amazon, and a century of written books using the Google Book Ngrams. I have also worked on predicting user age information with latent representations [112]. I show that DEEPWALK offers an attractive alternative to iterative methods for estimating user age in social networks.

My work in natural language processing has focused around representation learning for building massively multilingual NLP systems. I have analyzed the capabilities of word embedding methods [29], generated representations from the 137 languages on Wikipedia which had more than 10,000 articles [5], used a knowledge base and positive label learning to build named entity annotators for 40 languages [6], and analyzed the community structure present in the Polyglot embedded representations [115].

I have also done work in distributed graph algorithms for Hadoop [88] and Pregel [113, 118]. Finally, I have also worked on using anomalous link discovery to find bad recommendations in a very large entity recommendation network (The Google Related Places Graph) [120].

Chapter 2

Deep Learning for Social Networks

2.1 Introduction

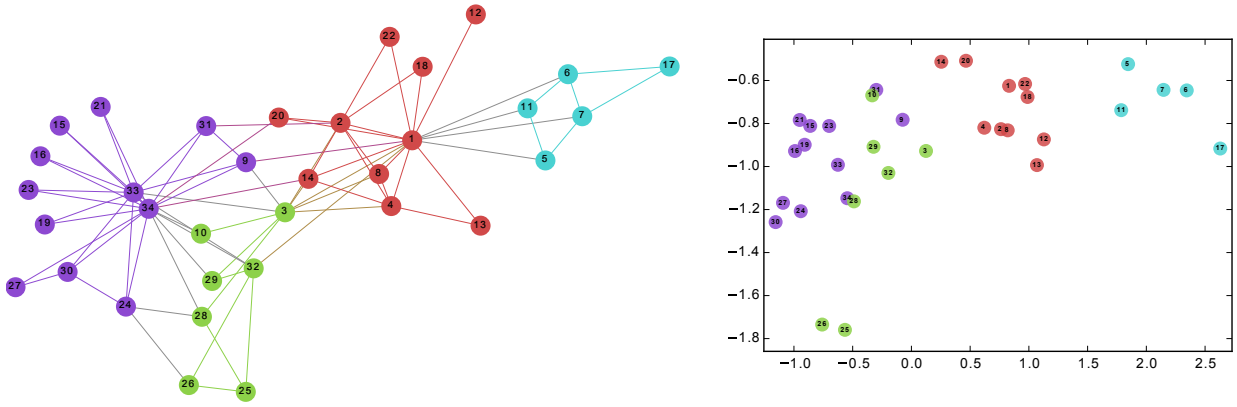
The sparsity of a network representation is both a strength and a weakness. Sparsity enables the design of efficient discrete algorithms, but can make it harder to generalize in statistical learning. Machine learning applications in networks (such as network classification [53, 125], content recommendation [46], anomaly detection [27], and missing link prediction [80]) must be able to deal with this sparsity in order to survive.

In this paper we introduce *deep learning* (unsupervised feature learning) [17] techniques, which have proven successful in natural language processing, into network analysis for the first time. We develop an algorithm (DEEPWALK) that learns *social representations* of a graph's vertices, by modeling a stream of short random walks. Social representations are latent features of the vertices that capture neighborhood similarity and community membership. These latent representations encode social relations in a continuous vector space with a relatively small number of dimensions. DEEPWALK generalizes neural language models to process a special language composed of a set of randomly-generated walks. These neural language models have been used to capture the semantic and syntactic structure of human language[33], and even logical analogies [94].

DEEPWALK takes a graph as input and produces a latent representation as an output. The result of applying our method to the well-studied Karate network is shown in Figure 2.1. The graph, as typically presented by force-directed layouts, is shown in Figure 2.1a. Figure 2.1b shows the output of our method with 2 latent dimensions. Beyond the striking similarity, we note that linearly separable portions of (2.1b) correspond to clusters found through modularity maximization in the input graph (2.1a) (shown as vertex colors).

To demonstrate DEEPWALK's potential in real world scenarios, we evaluate its performance on challenging multi-label network classification problems in large heterogeneous graphs. In the relational classification problem, the links between feature vectors violate the traditional *i.i.d.* assumption. Techniques to address this problem typically use approximate inference techniques

This work originally appeared as "Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 701–710, New York, NY, USA, 2014. ACM."



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

Figure 2.1: Our proposed method *learns* a latent space representation of social interactions in $\mathbb{R}^d$. The learned representation encodes community structure so it can be easily exploited by standard classification methods. Here, our method is used on Zachary's Karate network [153] to generate a latent representation in $\mathbb{R}^2$. Note the correspondence between community structure in the input graph and the embedding. Vertex colors represent a modularity-based clustering of the input graph.

[102] to leverage the dependency information to improve classification results. We distance ourselves from these approaches by learning label-independent representations of the graph. Our representation quality is not influenced by the choice of labeled vertices, so they can be shared among tasks.

DEEPWALK outperforms other latent representation methods for creating *social dimensions* [132, 134], especially when labeled nodes are scarce. Strong performance with our representations is possible with very simple linear classifiers (e.g. logistic regression). Our representations are general, and can be combined with any classification method (including iterative inference methods). DEEPWALK achieves all of that while being an online algorithm that is trivially parallelizable.

Our contributions are as follows:

- We introduce deep learning as a tool to analyze graphs, to build robust representations that are suitable for statistical modeling. DEEPWALK learns structural regularities present within short random walks.

- We extensively evaluate our representations on multi-label classification tasks on several social networks. We show significantly increased classification performance in the presence of label sparsity, getting improvements 5%-10% of Micro F_1, on the sparsest problems we consider. In some cases, DEEPWALK's representations can outperform its competitors even when given 60% less training data.

- We demonstrate the scalability of our algorithm by building representations of web-scale graphs, (such as YouTube) using a parallel implementation. Moreover, we describe the minimal changes necessary to build a streaming version of our approach.

The rest of the paper is arranged as follows. In Sections 4.3 and 2.3, we discuss the problem formulation of classification in data networks, and how it relates to our work. In Section 2.4 we present DEEPWALK, our approach for Social Representation Learning. We outline ours experiments in Section 2.5, and present their results in Section 5.4. We close with a discussion of related work in Section 5.2, and our conclusions.

2.2 Problem Definition

We consider the problem of classifying members of a social network into one or more categories. Let $G = (V, E)$, where V represent the members of the network, E are their connections, $E \subseteq (V \times V)$, and $G_L = (V, E, X, Y)$ is a partially labeled social network, with attributes $X \in \mathbb{R}^{|V| \times S}$ where S is the size of the feature space for each attribute vector, and $Y \in \mathbb{R}^{|V| \times |\mathcal{Y}|}$, $\mathcal{Y}$ is the set of labels.

In a traditional machine learning classification setting, we aim to learn a hypothesis H that maps elements of X to the labels set $\mathcal{Y}$. In our case, we can utilize the significant information about the dependence of the examples embedded in the structure of G to achieve superior performance.

In the literature, this is known as the relational classification (or the *collective classification* problem [125]). Traditional approaches to relational classification pose the problem as an inference in an undirected Markov network, and then use iterative approximate inference algorithms (such as the iterative classification algorithm [102], Gibbs Sampling [52], or label relaxation [65]) to compute the posterior distribution of labels given the network structure.

We propose a different approach to capture the network topology information. Instead of mixing the label space as part of the feature space, we propose an unsupervised method which learns features that capture the graph structure *independent* of the labels' distribution.

This separation between the structural representation and the labeling task avoids cascading errors, which can occur in iterative methods [104]. Moreover, the same representation can be used for multiple classification problems concerning that network.

Our goal is to learn $X_E \in \mathbb{R}^{|V| \times d}$, where d is small number of latent dimensions. These low-dimensional representations are distributed; meaning each social phenomena is expressed by a subset of the dimensions and each dimension contributes to a subset of the social concepts expressed by the space.

Using these structural features, we will augment the attributes space to help the classification decision. These features are general, and can be used with any classification algorithm (including iterative methods). However, we believe that the greatest utility of these features is their easy integration with simple machine learning algorithms. They scale appropriately in real-world networks, as we will show in Section 5.4.

2.3 Learning Social Representations

We seek to learn social representations with the following characteristics:

- **Adaptability** - Real social networks are constantly evolving; new social relations should not require repeating the learning process all over again.

- **Community aware** - The distance between latent dimensions should represent a metric for evaluating social similarity between the corresponding members of the network. This allows generalization in networks with homophily.
- **Low dimensional** - When labeled data is scarce low-dimensional models generalize better, and speed up convergence and inference.
- **Continuous** - We require latent representations to model partial community membership in continuous space. In addition to providing a nuanced view of community membership, a continuous representation has smooth decision boundaries between communities which allows more robust classification.

Our method satisfies these requirements by learning representation for vertices from a stream of short random walks, using optimization techniques originally designed for language modeling. Here, we review the basics of both random walks and language modeling, and describe how their combination satisfies our requirements.

2.3.1 Random Walks

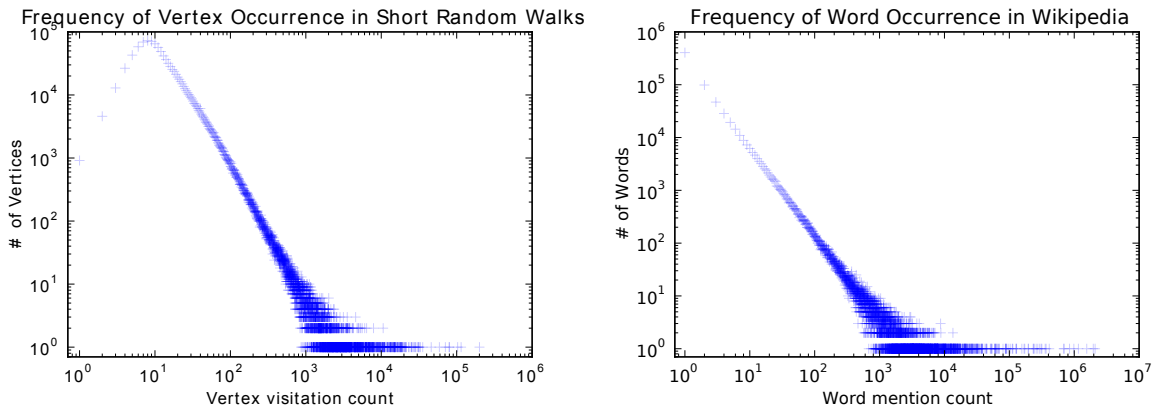
We denote a random walk rooted at vertex v_i as $\mathcal{W}_{v_i}$. It is a stochastic process with random variables $\mathcal{W}_{v_i}^1, \mathcal{W}_{v_i}^2, \ldots, \mathcal{W}_{v_i}^k$ such that $\mathcal{W}_{v_i}^{k+1}$ is a vertex chosen at random from the neighbors of vertex v_k. Random walks have been used as a similarity measure for a variety of problems in content recommendation [46] and community detection [8]. They are also the foundation of a class of *output sensitive* algorithms which use them to compute local community structure information in time sublinear to the size of the input graph [129].

It is this connection to local structure that motivates us to use a *stream* of short random walks as our basic tool for extracting information from a network. In addition to capturing community information, using random walks as the basis for our algorithm gives us two other desirable properties. First, local exploration is easy to parallelize. Several random walkers (in different threads, processes, or machines) can simultaneously explore different parts of the same graph. Secondly, relying on information obtained from short random walks make it possible to accommodate small changes in the graph structure without the need for global recomputation. We can iteratively update the learned model with new random walks from the changed region in time sub-linear to the entire graph.

2.3.2 Connection: Power laws

Having chosen online random walks as our primitive for capturing graph structure, we now need a suitable method to capture this information. If the degree distribution of a connected graph follows a power law (i.e. *scale-free*), we observe that the frequency which vertices appear in the short random walks will also follow a power-law distribution.

Word frequency in natural language follows a similar distribution, and techniques from language modeling account for this distributional behavior. To emphasize this similarity we show two different power-law distributions in Figure 2.2. The first comes from a series of short random walks



(a) YouTube Social Graph **(b) Wikipedia Article Text**

Figure 2.2: The distribution of vertices appearing in short random walks (2.2a) follows a power-law, much like the distribution of words in natural language (2.2b).

on a scale-free graph, and the second comes from the text of 100,000 articles from the English Wikipedia.

A core contribution of our work is the idea that techniques which have been used to model natural language (where the symbol frequency follows a power law distribution (or *Zipf's law*)) can be re-purposed to model community structure in networks. We spend the rest of this section reviewing the growing work in language modeling, and transforming it to learn representations of vertices which satisfy our criteria.

2.3.3 Language Modeling

The goal of language modeling is to estimate the likelihood of a specific sequence of words appearing in a corpus. More formally, given a sequence of words $W_1^n = (w_0, w_1, \cdots, w_n)$, where $w_i \in \mathcal{V}$ ($\mathcal{V}$ is the vocabulary), we would like to maximize the $\Pr(w_n | w_0, w_1, \cdots, w_{n-1})$ over all the training corpus. Recent work in representation learning has focused on using probabilistic neural networks to build general representations of words which extend the scope of language modeling beyond its original goals.

In this work, we present a generalization of language modeling to explore the graph through a stream of short random walks. These walks can be thought of as short sentences and phrases in a special language; the direct analog is to estimate the likelihood of observing vertex v_i given all the previous vertices visited so far in the random walk, i.e.

$$\Pr\left(v_i \mid (v_1, v_2, \cdots, v_{i-1})\right) \tag{2.1}$$

Our goal is to learn a latent representation, not only a probability distribution of node co-occurrences, and so we introduce a mapping function $\Phi \colon v \in V \mapsto \mathbb{R}^{|V| \times d}$. This mapping Φ represents the latent social representation associated with each vertex v in the graph. (In practice,

we represent Φ by a $|V| \times d$ matrix of free parameters, which will serve later on as our X_E). The problem then, is to estimate the likelihood:

$$\Pr\Big(v_i \mid \big(\Phi(v_1), \Phi(v_2), \cdots, \Phi(v_{i-1})\big)\Big) \tag{2.2}$$

However, as the walk length grows, computing this conditional probability becomes unfeasible.

A recent relaxation in language modeling [92, 93] turns the prediction problem on its head. First, instead of using the context to predict a missing word, it uses one word to predict the context. Secondly, the context is composed of the words appearing to both the right and left of the given word. Finally, it removes the ordering constraint on the problem, instead, requiring the model to maximize the probability of any word appearing in the context without the knowledge of its offset from the given word. In terms of vertex representation modeling, this yields the optimization problem:

$$\underset{\Phi}{\text{minimize}} \quad -\log \Pr\big(\{v_{i-w}, \cdots, v_{i+w}\} \setminus v_i \mid \Phi(v_i)\big) \tag{2.3}$$

We find these relaxations are particularly desirable for social representation learning. First, the order independence assumption better captures a sense of 'nearness' that is provided by random walks. Moreover, this relaxation is quite useful for speeding up the training time by building small models as one vertex is given at a time.

Solving the optimization problem from Eq. 2.3 builds representations that capture the shared similarities in local graph structure between vertices. Vertices which have similar neighborhoods will acquire similar representations (encoding co-citation similarity), allowing generalization on machine learning tasks.

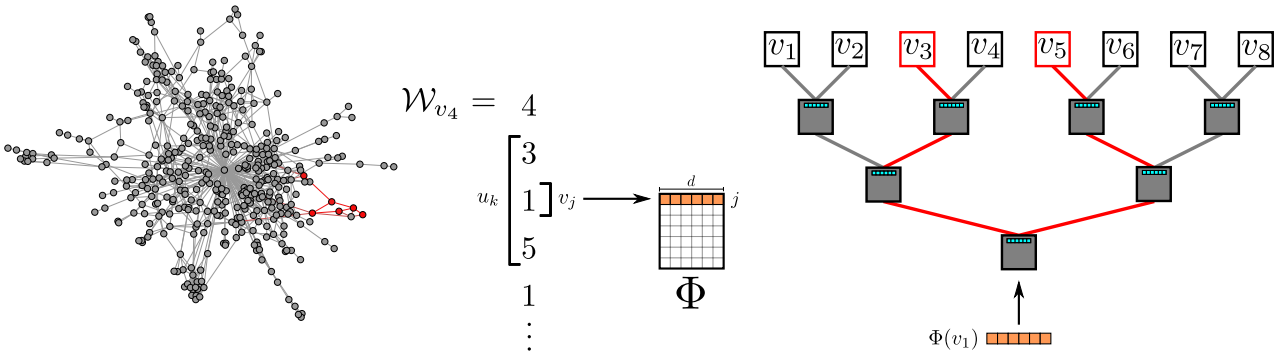
By combining both truncated random walks and language models we formulate a method which satisfies all of our desired properties. This method generates representations of social networks that are low-dimensional, and exist in a continuous vector space. Its representations encode latent forms of community membership, and because the method outputs useful intermediate representations, it can adapt to changing network topology.

2.4 Method

In this section we discuss the main components of our algorithm. We also present several variants of our approach and discuss their merits.

2.4.1 Overview

As in any language modeling algorithm, the only required input is a corpus and a vocabulary $\mathcal{V}$. DEEPWALK considers a set of short truncated random walks its own corpus, and the graph vertices as its own vocabulary ($\mathcal{V} = V$). While it is beneficial to know V and the frequency distribution of vertices in the random walks ahead of the training, it is not necessary for the algorithm to work as we will show in 2.4.2.



(a) Random walk generation. **(b) Representation mapping.** **(c) Hierarchical Softmax.**

Figure 2.3: Overview of DEEPWALK. We slide a window of length $2w + 1$ over the random walk $\mathcal{W}_{v_4}$, mapping the central vertex v_1 to its representation $\Phi(v_1)$. Hierarchical Softmax factors out $\Pr(v_3 \mid \Phi(v_1))$ and $\Pr(v_5 \mid \Phi(v_1))$ over sequences of probability distributions corresponding to the paths starting at the root and ending at v_3 and v_5. The representation Φ is updated to maximize the probability of v_1 co-occurring with its context $\{v_3, v_5\}$.

2.4.2 Algorithm: DEEPWALK

The algorithm consists of two main components; first a random walk generator, and second, an update procedure. The random walk generator takes a graph G and samples uniformly a random vertex v_i as the root of the random walk $\mathcal{W}_{v_i}$. A walk samples uniformly from the neighbors of the last vertex visited until the maximum length (t) is reached. While we set the length of our random walks in the experiments to be fixed, there is no restriction for the random walks to be of the same length. These walks could have restarts (i.e. a teleport probability of returning back to their root), but our preliminary results did not show any advantage of using restarts. In practice, our implementation specifies a number of random walks γ of length t to start at each vertex.

Lines 3-9 in Algorithm 1 shows the core of our approach. The outer loop specifies the number of times, γ, which we should start random walks at each vertex. We think of each iteration as making a 'pass' over the data and sample one walk per node during this pass. At the start of each pass we generate a random ordering to traverse the vertices. This is not strictly required, but is well-known to speed up the convergence of stochastic gradient descent.

In the inner loop, we iterate over all the vertices of the graph. For each vertex v_i we generate a random walk $|\mathcal{W}_{v_i}| = t$, and then use it to update our representations (Line 7). We use the SkipGram algorithm [92] to update these representations in accordance with our objective function in Eq. 2.3.

SkipGram

SkipGram is a language model that maximizes the co-occurrence probability among the words that appear within a window, w, in a sentence. It approximates the conditional probability in Equation

Algorithm 1 DEEPWALK(G, w, d, γ, t)

Input: graph $G(V, E)$
window size w
embedding size d
walks per vertex γ
walk length t
Output: matrix of vertex representations $\Phi \in \mathbb{R}^{|V| \times d}$

1: Initialization: Sample Φ from $\mathcal{U}^{|V| \times d}$
2: Build a binary Tree T from V
3: **for** $i = 0$ to γ **do**
4: $\quad \mathcal{O} = \text{Shuffle}(V)$
5: $\quad$ **for each** $v_i \in \mathcal{O}$ **do**
6: $\quad\quad \mathcal{W}_{v_i} = RandomWalk(G, v_i, \text{t})$
7: $\quad\quad$ SkipGram(Φ, $\mathcal{W}_{v_i}$, w)
8: $\quad$ **end for**
9: **end for**

Algorithm 2 SkipGram(Φ, $\mathcal{W}_{v_i}$, w)

1: **for each** $v_j \in \mathcal{W}_{v_i}$ **do**
2: $\quad$ **for each** $u_k \in \mathcal{W}_{v_i}[j - w : j + w]$ **do**
3: $\quad\quad J(\Phi) = -\log \Pr(u_k \mid \Phi(v_j))$
4: $\quad\quad \Phi = \Phi - \alpha * \frac{\partial J}{\partial \Phi}$
5: $\quad$ **end for**
6: **end for**

2.3 using an independence assumption as the following

$$\Pr\left(\{v_{i-w}, \cdots, v_{i+w}\} \setminus v_i \mid \Phi(v_i)\right) = \prod_{\substack{j=i-w \\ j \neq i}}^{i+w} \Pr(v_j | \Phi(v_i)) \tag{2.4}$$

Algorithm 2 iterates over all possible collocations in random walk that appear within the window w (lines 1-2). For each, we map each vertex v_j to its current representation vector $\Phi(v_j) \in \mathbb{R}^d$ (See Figure 2.3b). Given the representation of v_j, we would like to maximize the probability of its neighbors in the walk (line 3). We can learn such a posterior distribution using several choices of classifiers. For example, modeling the previous problem using logistic regression would result in a huge number of labels (that is equal to $|V|$) which could be in millions or billions. Such models require vast computational resources which could span a whole cluster of computers [15]. To avoid this necessity and speed up the training time, we instead use the Hierarchical Softmax [95, 97] to approximate the probability distribution.

Hierarchical Softmax

Given that $u_k \in V$, calculating $\Pr(u_k \mid \Phi(v_j))$ in line 3 is not feasible. Computing the partition function (normalization factor) is expensive, so instead we will factorize the conditional probability using Hierarchical softmax. We assign the vertices to the leaves of a binary tree, turning the prediction problem into maximizing the probability of a specific path in the hierarchy (See Figure 2.3c). If the path to vertex u_k is identified by a sequence of tree nodes $(b_0, b_1, \ldots, b_{\lceil \log |V| \rceil})$, ($b_0$ = root, $b_{\lceil \log |V| \rceil} = u_k$) then

$$\Pr(u_k \mid \Phi(v_j)) = \prod_{l=1}^{\lceil \log |V| \rceil} \Pr(b_l \mid \Phi(v_j)) \tag{2.5}$$

Now, $\Pr(b_l \mid \Phi(v_j))$ could be modeled by a binary classifier that is assigned to the parent of the node b_l as Equation 2.6 shows,

$$\Pr(b_l \mid \Phi(v_j) = 1/(1 + e^{-\Phi(v_j) \cdot \Psi(b_l)}) \tag{2.6}$$

where $\Psi(b_l) \in \mathbb{R}^d$ is the representation assigned to tree node b_l's parent. This reduces the computational complexity of calculating $\Pr(u_k \mid \Phi(v_j))$ from $O(|V|)$ to $O(\log |V|)$.

We can speed up the training process further, by assigning shorter paths to the frequent vertices in the random walks. Huffman coding is used to reduce the access time of frequent elements in the tree.

Optimization

The model parameter set is $\theta = \{\Phi, \Psi\}$ where the size of each is $O(d|V|)$. Stochastic gradient descent (SGD) [22] is used to optimize these parameters (Line 4, Algorithm 2). The derivatives are estimated using the back-propagation algorithm. The learning rate α for SGD is initially set to 2.5% at the beginning of the training and then decreased linearly with the number of vertices that are seen so far.

2.4.3 Parallelizability

As shown in Figure 2.2 the frequency distribution of vertices in random walks of social network and words in a language both follow a power law. This results in a long tail of infrequent vertices, therefore, the updates that affect Φ will be sparse in nature. This allows us to use asynchronous version of stochastic gradient descent (ASGD), in the multi-worker case. Given that our updates are sparse and we do not acquire a lock to access the model shared parameters, ASGD will achieve an optimal rate of convergence [121]. While we run experiments on one machine using multiple threads, it has been demonstrated that this technique is highly scalable, and can be used in very large scale machine learning [37]. Figure 2.4 presents the effects of parallelizing DEEPWALK. It shows the speed up in processing BLOGCATALOG and FLICKR networks is consistent as we increase the number of workers to 8 (Figure 2.4a). It also shows that there is no loss of predictive performance relative to the running DEEPWALK serially (Figure 2.4b).

Name	BLOGCATALOG	FLICKR	YOUTUBE
$\|V\|$	10,312	80,513	1,138,499
$\|E\|$	333,983	5,899,882	2,990,443
$\|\mathcal{Y}\|$	39	195	47
Labels	Interests	Groups	Groups

Table 2.1: Graphs used in our experiments.

2.4.4 Algorithm Variants

Here we discuss some variants of our proposed method, which we believe may be of interest.

Streaming

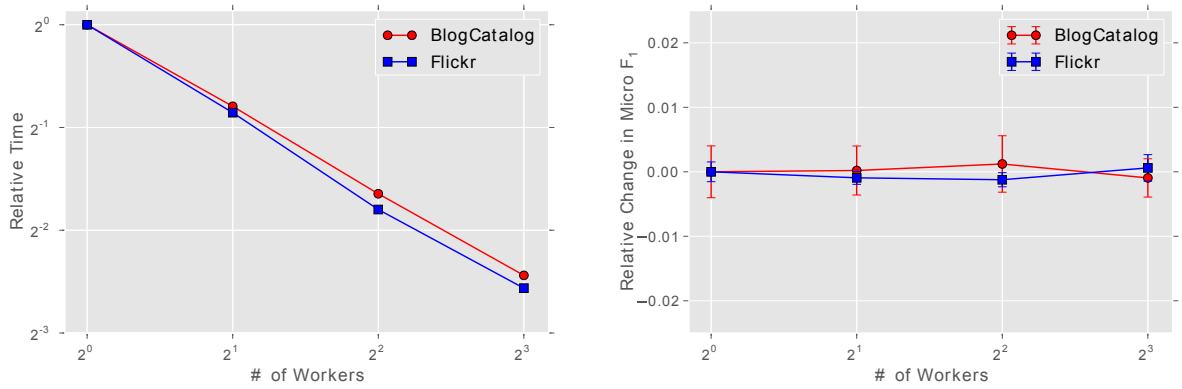
One interesting variant of this method is a *streaming* approach, which could be implemented without knowledge of the entire graph. In this variant small walks from the graph are passed directly to the representation learning code, and the model is updated directly. Some modifications to the learning process will also be necessary. First, using a decaying learning rate may no longer be desirable as it assumes the knowledge of the total corpus size. Instead, we can initialize the learning rate α to a small constant value. This will take longer to learn, but may be worth it in some applications. Second, we cannot necessarily build a tree of parameters any more. If the cardinality of V is known (or can be bounded), we can build the Hierarchical Softmax tree for that maximum value. Vertices can be assigned to one of the remaining leaves when they are first seen. If we have the ability to estimate the vertex frequency a priori, we can also still use Huffman coding to decrease frequent element access times.

Non-random walks

Some graphs are created as a by-product of agents interacting with a sequence of elements (e.g. users' navigation of pages on a website). When a graph is created by such a stream of *non-random* walks, we can use this process to feed the modeling phase directly. Graphs sampled in this way will not only capture information related to network structure, but also to the frequency at which paths are traversed.

In our view, this variant also encompasses language modeling. Sentences can be viewed as purposed walks through an appropriately designed language network, and language models like SkipGram are designed to capture this behavior.

This approach can be combined with the streaming variant (Section 2.4.4) to train features on a continually evolving network without ever explicitly constructing the entire graph. Maintaining representations with this technique could enable web-scale classification without the hassles of dealing with a web-scale graph.



(a) Running Time **(b) Performance**

Figure 2.4: Effects of parallelizing DEEPWALK

2.5 Experimental Design

In this section we provide an overview of the datasets and methods which we will use in our experiments. Code and data to reproduce our results will be available at the first author's website.[1]

2.5.1 Datasets

An overview of the graphs we consider in our experiments is given in Figure 2.1.

- BLOGCATALOG[132] is a network of social relationships provided by blogger authors. The labels represent the topic categories provided by the authors.
- FLICKR[132] is a network of the contacts between users of the photo sharing website. The labels represent the interest groups of the users such as '*black and white photos*'.
- YOUTUBE[133] is a social network between users of the popular video sharing website. The labels here represent groups of viewers that enjoy common video genres (e.g. *anime* and *wrestling*).

2.5.2 Baseline Methods

To validate the performance of our approach we compare it against a number of baselines:

- SpectralClustering[134]: This method generates a representation in $\mathbb{R}^d$ from the d-smallest eigenvectors of $\widetilde{\mathcal{L}}$, the normalized graph Laplacian of G. Utilizing the eigenvectors of $\widetilde{\mathcal{L}}$ implicitly assumes that graph cuts will be useful for classification.
- Modularity[132]: This method generates a representation in $\mathbb{R}^d$ from the top-d eigenvectors of B, the Modularity matrix of G. The eigenvectors of B encode information about modular graph partitions of G[106]. Using them as features assumes that modular graph partitions will be useful for classification.

[1] `http://bit.ly/deepwalk`

	% Labeled Nodes	10%	20%	30%	40%	50%	60%	70%	80%	90%
	DEEPWALK	**36.00**	**38.20**	**39.60**	**40.30**	**41.00**	**41.30**	41.50	41.50	42.00
	SpectralClustering	31.06	34.95	37.27	38.93	39.97	40.99	**41.66**	**42.42**	**42.62**
	EdgeCluster	27.94	30.76	31.85	32.99	34.12	35.00	34.63	35.99	36.29
Micro-F1(%)	Modularity	27.35	30.74	31.77	32.97	34.09	36.13	36.08	37.23	38.18
	wvRN	19.51	24.34	25.62	28.82	30.37	31.81	32.19	33.33	34.28
	Majority	16.51	16.66	16.61	16.70	16.91	16.99	16.92	16.49	17.26
	DEEPWALK	**21.30**	**23.80**	25.30	26.30	27.30	27.60	27.90	28.20	28.90
	SpectralClustering	19.14	23.57	**25.97**	**27.46**	**28.31**	**29.46**	**30.13**	**31.38**	**31.78**
	EdgeCluster	16.16	19.16	20.48	22.00	23.00	23.64	23.82	24.61	24.92
Macro-F1(%)	Modularity	17.36	20.00	20.80	21.85	22.65	23.41	23.89	24.20	24.97
	wvRN	6.25	10.13	11.64	14.24	15.86	17.18	17.98	18.86	19.57
	Majority	2.52	2.55	2.52	2.58	2.58	2.63	2.61	2.48	2.62

Table 2.2: Multi-label classification results in BLOGCATALOG

	% Labeled Nodes	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
	DEEPWALK	**32.4**	**34.6**	**35.9**	**36.7**	**37.2**	**37.7**	**38.1**	**38.3**	**38.5**	**38.7**
	SpectralClustering	27.43	30.11	31.63	32.69	33.31	33.95	34.46	34.81	35.14	35.41
Micro-F1(%)	EdgeCluster	25.75	28.53	29.14	30.31	30.85	31.53	31.75	31.76	32.19	32.84
	Modularity	22.75	25.29	27.3	27.6	28.05	29.33	29.43	28.89	29.17	29.2
	wvRN	17.7	14.43	15.72	20.97	19.83	19.42	19.22	21.25	22.51	22.73
	Majority	16.34	16.31	16.34	16.46	16.65	16.44	16.38	16.62	16.67	16.71
	DEEPWALK	**14.0**	17.3	**19.6**	**21.1**	**22.1**	**22.9**	**23.6**	**24.1**	**24.6**	**25.0**
	SpectralClustering	13.84	**17.49**	19.44	20.75	21.60	22.36	23.01	23.36	23.82	24.05
Macro-F1(%)	EdgeCluster	10.52	14.10	15.91	16.72	18.01	18.54	19.54	20.18	20.78	20.85
	Modularity	10.21	13.37	15.24	15.11	16.14	16.64	17.02	17.1	17.14	17.12
	wvRN	1.53	2.46	2.91	3.47	4.95	5.56	5.82	6.59	8.00	7.26
	Majority	0.45	0.44	0.45	0.46	0.47	0.44	0.45	0.47	0.47	0.47

Table 2.3: Multi-label classification results in FLICKR

- EdgeCluster[133]: This method uses k-means clustering to cluster the adjacency matrix of G. Its has been shown to perform comparably to the Modularity method, with the added advantage of scaling to graphs which are too large for spectral decomposition.
- wvRN[83]: The weighted-vote Relational Neighbor is a relational classifier. Given the neighborhood $\mathcal{N}_i$ of vertex v_i, wvRN estimates $\Pr(y_i|\mathcal{N}_i)$ with the (appropriately normalized) weighted mean of its neighbors (i.e $\Pr(y_i|\mathcal{N}_i) = \frac{1}{Z}\sum_{v_j \in \mathcal{N}_i} w_{ij} \Pr(y_j \mid \mathcal{N}_j)$). It has shown surprisingly good performance in real networks, and has been advocated as a sensible relational classification baseline [84].
- Majority: This naïve method simply chooses the most frequent labels in the training set.

2.6 Experiments

	% Labeled Nodes	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
	DEEPWALK	**37.95**	**39.28**	**40.08**	**40.78**	**41.32**	**41.72**	**42.12**	**42.48**	**42.78**	**43.05**
	SpectralClustering	—	—	—	—	—	—	—	—	—	—
Micro-F1(%)	EdgeCluster	23.90	31.68	35.53	36.76	37.81	38.63	38.94	39.46	39.92	40.07
	Modularity	—	—	—	—	—	—	—	—	—	—
	wvRN	26.79	29.18	33.1	32.88	35.76	37.38	38.21	37.75	38.68	39.42
	Majority	24.90	24.84	25.25	25.23	25.22	25.33	25.31	25.34	25.38	25.38
	DEEPWALK	**29.22**	**31.83**	**33.06**	**33.90**	**34.35**	**34.66**	**34.96**	**35.22**	**35.42**	**35.67**
	SpectralClustering	—	—	—	—	—	—	—	—	—	—
Macro-F1(%)	EdgeCluster	19.48	25.01	28.15	29.17	29.82	30.65	30.75	31.23	31.45	31.54
	Modularity	—	—	—	—	—	—	—	—	—	—
	wvRN	13.15	15.78	19.66	20.9	23.31	25.43	27.08	26.48	28.33	28.89
	Majority	6.12	5.86	6.21	6.1	6.07	6.19	6.17	6.16	6.18	6.19

Table 2.4: Multi-label classification results in YOUTUBE

In this section we present an experimental analysis of our method. We thoroughly evaluate it on a number of multi-label classification tasks, and analyze its sensitivity across several parameters.

2.6.1 Multi-Label Classification

To facilitate the comparison between our method and the relevant baselines, we use the exact same datasets and experimental procedure as in [132, 133]. Specifically, we randomly sample a portion (T_R) of the labeled nodes, and use them as training data. The rest of the nodes are used as test. We repeat this process 10 times, and report the average performance in terms of both Macro-F_1 and Micro-F_1. When possible we report the original results [132, 133] here directly.

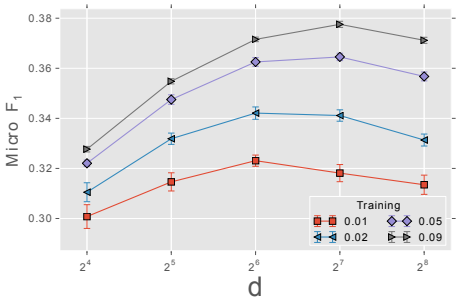
For all models we use a one-vs-rest logistic regression implemented by LibLinear [43] extended to return the most probable labels as in [132]. We present results for DEEPWALK with ($\gamma = 80$, $w = 10$, $d = 128$). The results for (SpectralClustering, Modularity, EdgeCluster) use Tang and Liu's preferred dimensionality, $d = 500$.

BlogCatalog

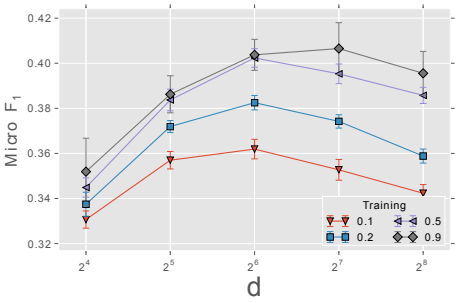
In this experiment we increase the training ratio (T_R) on the BLOGCATALOG network from 10% to 90%. Our results are presented in Table 2.2. Numbers in bold represent the highest performance in each column.

DEEPWALK performs consistently better than EdgeCluster, Modularity, and wvRN. In fact, when trained with only 20% of the nodes labeled, DEEPWALK performs better than these approaches when they are given 90% of the data. The performance of SpectralClustering proves much more competitive, but DEEPWALK still outperforms when labeled data is sparse on both Macro-F_1 ($T_R \leq 20\%$) and Micro-F_1 ($T_R \leq 60\%$).

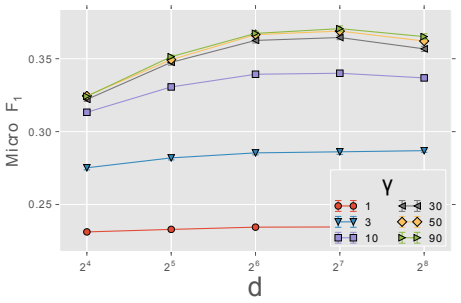
This strong performance when only small fractions of the graph are labeled is a core strength of our approach. In the following experiments, we investigate the performance of our representations on even more sparsely labeled graphs.



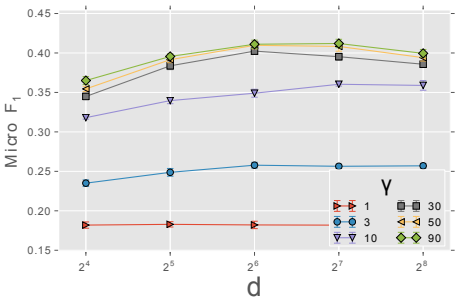
(a1) FLICKR, $\gamma = 30$



(a3) BLOGCATALOG, $\gamma = 30$

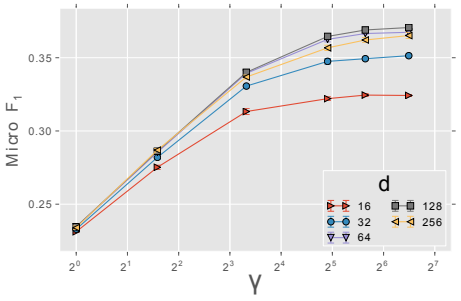


(a2) FLICKR, $T_R = 0.05$

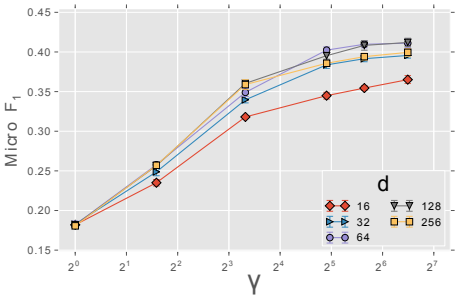


(a4) BLOGCATALOG, $T_R = 0.5$

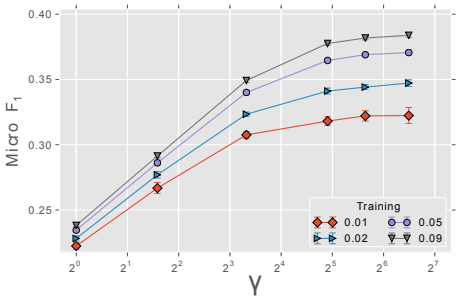
(a) Stability over dimensions, d



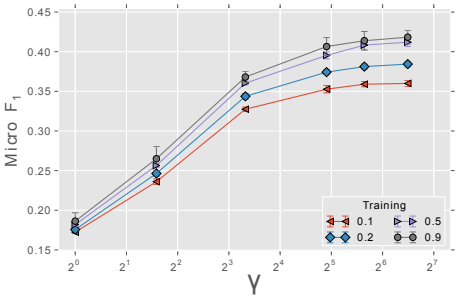
(b1) FLICKR, $T_R = 0.05$



(b3) BLOGCATALOG, $T_R = 0.5$



(b2) FLICKR, $d = 128$



(b4) BLOGCATALOG, $d = 128$

(b) Stability over number of walks, γ

Figure 2.5: Parameter Sensitivity Study

Flickr

In this experiment we vary the training ratio (T_R) on the FLICKR network from 1% to 10%. This corresponds to having approximately 800 to 8,000 nodes labeled for classification in the entire network. Table 2.3 presents our results, which are consistent with the previous experiment. DEEPWALK outperforms all baselines by at least 3% with respect to Micro-F_1. Additionally, its Micro-F_1 performance when only 3% of the graph is labeled beats all other methods even when they have been given 10% of the data. In other words, DEEPWALK can outperform the baselines with 60% less training data. It also performs quite well in Macro-F_1, initially performing close to SpectralClustering, but distancing itself to a 1% improvement.

YouTube

The YOUTUBE network is considerably larger than the previous ones we have experimented on, and its size prevents two of our baseline methods (SpectralClustering and Modularity) from running on it. It is much closer to a real world graph than those we have previously considered.

The results of varying the training ratio (T_R) from 1% to 10% are presented in Table 2.4. They show that DEEPWALK significantly outperforms the scalable baseline for creating graph representations, EdgeCluster. When 1% of the labeled nodes are used for test, the Micro-F_1 improves by 14%. The Macro-F_1 shows a corresponding 10% increase. This lead narrows as the training data increases, but DEEPWALK ends with a 3% lead in Micro-F_1, and an impressive 5% improvement in Macro-F_1.

This experiment showcases the performance benefits that can occur from using social representation learning for multi-label classification. DEEPWALK, can scale to large graphs, and performs exceedingly well in such a sparsely labeled environment.

2.6.2 Parameter Sensitivity

In order to evaluate how changes to the parameterization of DEEPWALK effect its performance on classification tasks, we conducted experiments on two multi-label classifications tasks (FLICKR, and BLOGCATALOG). In the interest of brevity, we have fixed the window size and the walk length to emphasize local structure ($w = 10$, $t = 40$). We then vary the number of latent dimensions (d), the number of walks started per vertex (γ), and the amount of training data available (T_R) to determine their impact on the network classification performance.

Effect of Dimensionality

Figure 2.5a shows the effects of increasing the number of latent dimensions available to our model.

Figures 2.5a1 and 2.5a3 examine the effects of varying the dimensionality and training ratio. The performance is quite consistent between both FLICKR and BLOGCATALOG and show that the optimal dimensionality for a model is dependent on the number of training examples. (Note that 1% of FLICKR has approximately as many labeled examples as 10% of BLOGCATALOG).

Figures 2.5a2 and 2.5a4 examine the effects of varying the dimensionality and number of walks per vertex. The relative performance between dimensions is relatively stable across different values

of γ. These charts have two interesting observations. The first is that there is most of the benefit is accomplished by starting $\gamma = 30$ walks per node in both graphs. The second is that the relative difference between different values of γ is quite consistent between the two graphs. FLICKR has an order of magnitude more edges than BLOGCATALOG, and we find this behavior interesting.

These experiments show that our method can make useful models of various sizes. They also show that the performance of the model depends on the number of random walks it has seen, and the appropriate dimensionality of the model depends on the training examples available.

Effect of sampling frequency

Figure 2.5b shows the effects of increasing γ, the number of random walks that we start from each vertex.

The results are very consistent for different dimensions (Fig. 2.5b1, Fig. 2.5b3) and the amount of training data (Fig. 2.5b2, Fig. 2.5b4). Initially, increasing γ has a big effect in the results, but this effect quickly slows ($\gamma > 10$). These results demonstrate that we are able to learn meaningful latent representations for vertices after only a small number of random walks.

2.7 Related Work

The main differences between our proposed method and previous work can be summarized as follows:

1. We *learn* our latent social representations, instead of computing statistics related to centrality [48] or partitioning [134].
2. We do not attempt to extend the classification procedure itself (through collective inference [125] or graph kernels [69]).
3. We propose a scalable online method which uses only local information. Most methods require global information and are offline [61, 132–134].
4. We apply unsupervised representation learning to graphs.

In this section we discuss related work in network classification and unsupervised feature learning.

2.7.1 Relational Learning

Relational classification (or *collective classification*) methods [52, 83, 102] use links between data items as part of the classification process. Exact inference in the collective classification problem is NP-hard, and solutions have focused on the use of approximate inference algorithm which may not be guaranteed to converge [125].

The most relevant relational classification algorithms to our work incorporate community information by learning clusters [103], by adding edges between nearby nodes [49], by using PageRank [81], or by extending relational classification to take additional features into account [144]. Our work takes a substantially different approach. Instead of a new approximation inference

algorithm, we propose a procedure which learns representations of network structure which can then be used by existing inference procedure (including iterative ones).

A number of techniques for generating features from graphs have also been proposed [48, 61, 132–134]. In contrast to these methods, we frame the feature creation procedure as a representation learning problem.

Graph Kernels [142] have been proposed as a way to use relational data as part of the classification process, but are quite slow unless approximated [67]. Our approach is complementary; instead of encoding the structure as part of a kernel function, we learn a representation which allows them to be used directly as features for any classification method.

2.7.2 Unsupervised Feature Learning

Distributed representations have been proposed to model structural relationship between concepts [62]. These representations are trained by the back-propagation and gradient descent. Computational costs and numerical instability led to these techniques to be abandoned for almost a decade. Recently, distributed computing allowed for larger models to be trained [15], and the growth of data for unsupervised learning algorithms to emerge [42]. Distributed representations usually are trained through neural networks, these networks have made advancements in diverse fields such as computer vision [70], speech recognition [36], and natural language processing [5, 33].

2.8 Conclusions

We propose DEEPWALK, a novel approach for learning latent social representations of vertices. Using local information from truncated random walks as input, our method learns a representation which encodes structural regularities. Experiments on a variety of different graphs illustrate the effectiveness of our approach on challenging multi-label classification tasks.

As an online algorithm, DEEPWALK is also scalable. Our results show that we can create meaningful representations for graphs which are too large for standard spectral methods. On such large graphs, our method significantly outperforms other methods designed to operate for sparsity. We also show that our approach is parallelizable, allowing workers to update different parts of the model concurrently.

In addition to being effective and scalable, our approach is also an appealing generalization of language modeling. This connection is mutually beneficial. Advances in language modeling may continue to generate improved latent representations for networks. In our view, language modeling is actually sampling from an unobservable language graph. We believe that insights obtained from modeling observable graphs may in turn yield improvements to modeling unobservable ones.

Our future work in the area will focus on investigating this duality further, using our results to improve language modeling, and strengthening the theoretical justifications of the method.

Chapter 3

Walklets: Multiscale Graph Embeddings for Interpretable Network Classification

3.1 Introduction

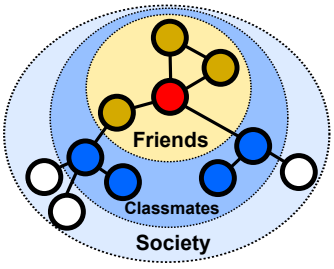
Human society is inherently hierarchical. Each individual is a member of several communities, which range from small (families, friends), to medium (organizations such as schools and businesses)), to large (nation states, set of all people who speak a common language, etc.).

As the scale of these relationships change, so too can their topology. For example, consider a university student on a social network (as illustrated in Figure 3.1a). They may be very tightly tied to their friends and immediate family, and form dense graph structures with these individuals (e.g. near cliques). However they will be more loosely tied to the average student at their university – in fact, they will have no direct social connection to the majority of their fellow students. Finally, they will have relatively few ties to all individuals in their nation, but still will share many attributes due to a common culture.

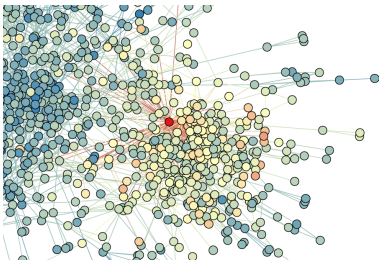
The prediction tasks on social networks also vary from the specific (e.g. a user's movie interests) to attributes associated with the more general communities a user is a member of (e.g. past employers or schools). Previous work on social representation learning has treated the process as a 'one-size fits all' problem, where a single social representation is used for all predictions made for a single user. In our view, it is desirable to have a family of representations which capture the full range of an individual's community membership.

In this paper, we study the problem of embedding a large graph into a finite number of dimensions $d \in \mathcal{R}^d$ that capture the latent hierarchy of communities which an individual participates in. These latent *multiscale representations* of the graph are useful for machine learning tasks in social networks. At prediction time, the representations can be leveraged (both individually, or combined) to provide a more comprehensive model of the user's affiliations. The difference between similarity (distance in the latent space) over varying representation scales is illustrated in Figures 3.1b and 3.1c.

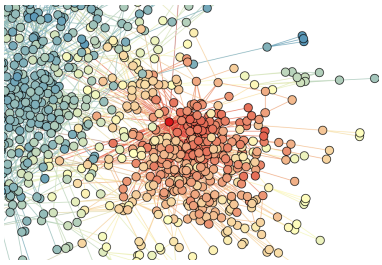
This work originally appeared as "Bryan Perozzi, Vivek Kulkarni, and Steven Skiena. Walklets: Multiscale graph embeddings for interpretable network classification. In *(submitted)*, 2016."



(a) A student (in red) is a member of several increasing larger social communities.



(b) WALKLETS Fine Representation



(c) WALKLETS Coarse Representation

Figure 3.1: Our method captures multiple scales of social relationships like those shown in Figure 3.1a. This is illustrated as a heatmap on the original graph in Figures 3.1b,3.1c. Color depicts cosine distance to a single vertex, with red indicating close vertices (low distance), and blue far vertices (high distance). Figure 3.1b: Only immediate are near the input vertex in a fine grained representation. Figure 3.1c: In a coarse representation, all vertices in the local cluster of the graph are close to the input vertex. Subgraph from the `Cora` citation network.

Recently, there has been a surge of interest in representation learning for social networks, primarily based around neural matrix factorization [25, 117, 131]. These methods have extended models developed for representation learning (or *deep learning*) in natural language processing for analysis of social and information networks, and demonstrated strong task performance on semi-supervised network classification problems.

Despite their strong task performance, we find that these methods leave much to be desired. First, initial work only addressed the problem of learning representations at *multiple scales* indirectly, either as an artifact of the learning process [117] or via an unintuitive combination of different objective functions [131]. More recently, an approach for learning a multiscale representation [25] has been proposed, but its computational complexity renders it intractable for most real world graphs. Second, all these methods have centered on a 'one size fits all' approach to network representation learning, which mask the nuanced information which can be present at each individual scale of graph representation.

In this work, we propose Multiscale Network Embedding (WALKLETS), an algorithm for learning social representations which capture multiple scales of relationships between vertices in a graph. Unlike existing work, WALKLETS's dimensions have meaning, allows for informed network classification, and visualization of relationships captured. The method itself is scalable, and can be run on graphs with millions of vertices.

Specifically, our contributions are the following:

1. **Multiscale Representation Learning**: We propose a graph embedding algorithm which explicitly captures multiple scales of relationships in networked data.

2. **Evaluation**: We extensively evaluate our representations on multilabel classification tasks on several social networks (like LastFM, Twitter, and YouTube). WALKLETS outperforms several challenging baselines like DeepWalk [117], by up to 5 points of Micro-F1.

3. **Visualization and Analysis**: WALKLETS preserves multiple scales of latent representations, which we use to analyzing the multiscale effects present in large graphs.

The rest of the paper is arranged as follows. In Section 3.2 we present a brief overview of representation learning for social networks, and expound on their use for graph classification problems. We follow this by analytically deriving the basis WALKLETS, our approach for Multiscale Social Representation Learning in Section 3.3. We outline ours experiments in Section 3.4, and present their results in Section 3.5. We close with a discussion of related work in Section 3.7, and our conclusions.

3.2 Preliminaries

In this section, we briefly discuss necessary preliminaries of neural representation learning as it relates to our work.

3.2.1 Problem Definition

Let $G = (V, E)$, where V represent the members of the network, E are their connections, $E \subseteq (V \times V)$, and $G_L = (V, E, X, Y)$ is a partially labeled social network, with attributes $X \in \mathbb{R}^{|V| \times S}$ where S is the size of the feature space for each attribute vector, and $Y \in \mathbb{R}^{|V| \times |\mathcal{Y}|}$, $\mathcal{Y}$ is the set of labels. We will refer to G's adjacency matrix A, $A \in \mathbb{R}^{|V| \times |V|}$. The entry A_{ij} is non-zero if and only if there is an edge $(i, j) \in E$.

Our goal is to learn a family of k successively coarser social representations, $X_{E_1}, X_{E_2}, \ldots, X_{E_k}$, $X_{E_k} \in \mathbb{R}^{|V| \times d}$. In these representations, each vertex $v_i \in V$ is represented by d latent dimensions which encodes social similarity in continuous space. Each level of the family encodes a different view of social similarity, corresponding to shared membership in latent communities of differing scales.

Using these structural features, we will learn a hypothesis H that maps the representations of X_E to the labels set $\mathcal{Y}$. This allows generalization (i.e. to infer labels for vertices in G which do not have labels).

3.2.2 Representation Learning

The goal of representation learning is to infer a mapping function $\Phi \colon v \in V \mapsto \mathbb{R}^{|V| \times d}$. This mapping Φ represents the latent social representation associated with each vertex v in the graph. (In practice, we represent Φ by a $|V| \times d$ matrix of free parameters, which will serve later on as our X_E).

In our previous work, [117] we introduce the concept of modeling a vertex as a function of its node co-occurrences in a truncated random walk. These sequences capture the diffusion process around each vertex in the graph, and encode local community structure. Consequently the goal, is to estimate the likelihood of a vertex v_i co-occurring with its local neighborhood:

$$\Pr\Big(v_i \mid \big(\Phi(v_1), \Phi(v_2), \cdots, \Phi(v_{i-1})\big)\Big) \tag{3.1}$$

However, as the walk length grows, computing this conditional probability becomes unfeasible.

To address this computational complexity, we proposed to relax the problem by following [92], by ignoring the order of neighboring vertices, and instead of using the context to predict a missing vertex, it uses one vertex to predict its local structure. In terms of vertex representation modeling, this yields the optimization problem:

$$\underset{\Phi}{\text{minimize}} \quad -\log \Pr\left(\{v_{i-w}, \cdots, v_{i+w}\} \setminus v_i \mid \Phi(v_i)\right) \tag{3.2}$$

Interestingly, this optimization is directly equivalent to learning a low rank approximation of the adjacency matrix, A.

3.2.3 Neural Matrix Factorization

A commonly proposed approach to model the probability of a node v_i co-occurring with v_j uses a softmax to map the pairwise similarity to a probability space,

$$Pr(v_i|v_j) = \frac{exp(\Phi(v_i) \cdot \Phi(v_j))}{\sum_{j \in \mathcal{V}} exp(\Phi(v_i) \cdot \Phi(v_j)}. \tag{3.3}$$

Unfortunately, calculating the denominator in 3.3 is computational expensive.

Alternatively, noise-contrastive estimation [59] has been proposed as a relaxation of (3.3) [25, 92]. It models the probability of a vertex co-occurrence pair (v_i, v_j) appearing as:

$$P((v_i, v_j)) = \sigma(\Phi(v_i) \cdot \Phi(v_j)) = \frac{1}{1 + e^{-\Phi(v_i) \cdot \Phi(v_j)}} \tag{3.4}$$

It can be shown that both of these probabilistic models correspond to implicitly factoring a transformation of the adjacency matrix [77, 147, 150].

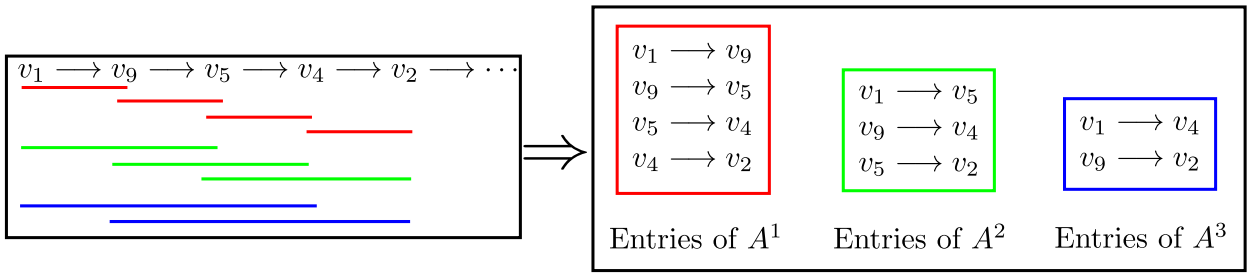
Social Representations as Matrix Factorization

To motivate our method, we briefly discuss the matrices which prior work on learning social representations are implicitly factoring. Specifically, we discuss our closest related work - DeepWalk [117], and LINE [131].[1]

DeepWalk: A closed form solution which describes the implicit matrix factorization performed by DeepWalk is possible, as noted by [25, 150]. They derive a formulation for DeepWalk with Hierarchical Softmax (DWHS) showing that it is factoring a matrix M containing the random walk transition matrix raised to the t power. In other words, the entry M_{ij} is the expectation of a path of length t started at node i, ending at node j.

$$M_{ij}^{DWHS} = \log \frac{\#(v_i, v_j)}{\#(v_i)} = \log \frac{[e_i(A + A^2 + \cdots + A^t)]_j}{t} \tag{3.5}$$

[1] We recently became aware of GraRep [25] - independent work, which draws on similar insights to propose multiscale models. We discuss differences between it and our approach further in Sections 3.5, 3.6.3, and 3.7.



Random walk, rooted at v_1 — WALKLETS corpus of skipped random walks

Figure 3.2: Overview of WALKLETS. Our method samples edges from higher powers of the adjacency matrix using a rooted random walk and skips over vertices. An edge sampled from A^k represents a paths of length k in the original graph.

where $\#(v_i, v_j)$ is counting function, that counts the occurrences of pair (v_i, v_j), and e_i is a $|V|$-dimensional vector which serves as an indicator function (having a 1 in the i-th row, and 0s elsewhere).

LINE: Similarly, the second order dependencies modeled by LINE can be shown to be equivalent to factorizing a matrix M whose entries are given by:

$$M_{ij}^{SGNS} = \text{PMI}(v_i, v_j) - \log k = \log \frac{(\sum_{A_i \cap A_j} 1)|E|}{(\sum_{A_i} 1)(\sum_{A_j} 1)} - \log k \tag{3.6}$$

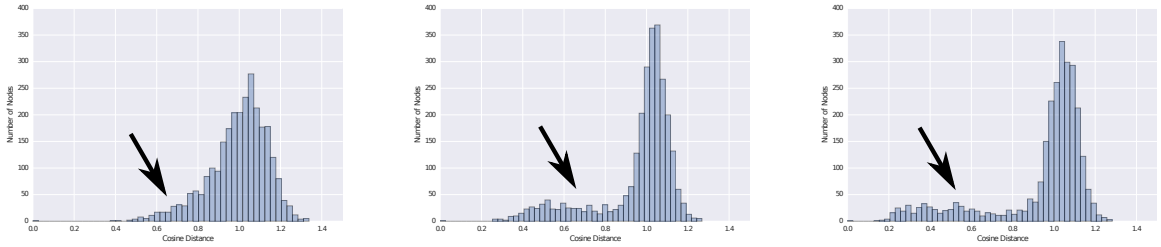
where $\text{PMI}(x, y) = \log \frac{P(x,y)}{P(x)P(y)}$, the Pointwise Mutual Information between discrete observations of x and y (in this case, the edges present in A), and k is a constant.

3.3 Multi-scale Neural Matrix Decomposition

In this section we introduce our algorithm for creating multiscale network representations. We begin by describing our model for capturing different representations scales. We follow this with a discussion of concerns, like search strategy and optimization. Finally, we perform a case study on a small real world citation network, which illustrates the different scales of network representations captured through our method.

3.3.1 Model Description

Here, we build on the intuition developed in section 3.2.3 and formally extend previous methods in social representation learning to explicitly model the multi-scale effects exhibited in real world networks.



(a) Finest Scale: WALKLETS(A^1) **(b) WALKLETS(A^3)** **(c) Coarsest Scale: WALKLETS(A^5)**

Figure 3.3: The distribution of distances to other vertices from v_{35} in the `Cora` network at different scales of network representation. Coarser representations (such as WALKLETS(A^5) 'flatten' the distribution, making larger communities close to the source vertex. Graph heatmap of corresponding distances shown in Figure 3.4.

Multi-scale Properties of DeepWalk

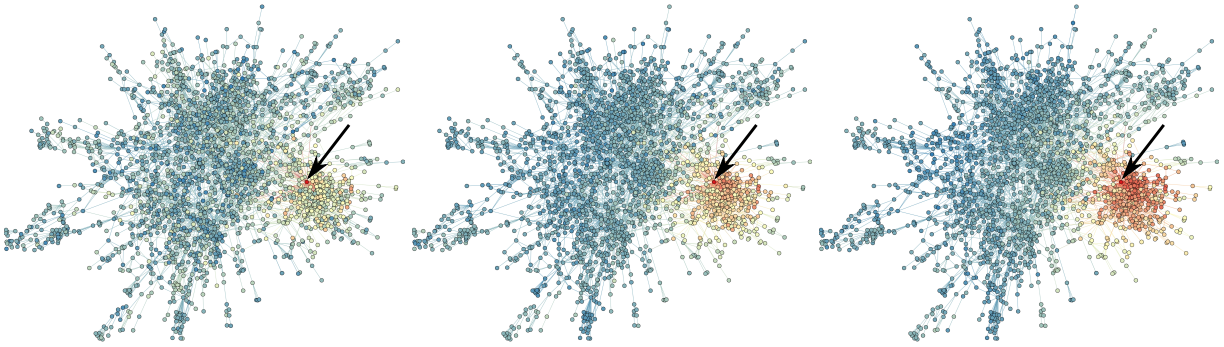
As shown in Eq.(3.5) DeepWalk is implicitly factoring a matrix containing entries of $A, A^2 \ldots, A^k$, where k is the window size over the random walk. Each power of A^k represents a different scale of network structure (recall, the entries of A^k_{ij} is the number of paths between nodes i and j of length k).

It is interesting to see then, that DeepWalk, is already implicitly modeling dependencies of multiple scales. This shows that the representations learned are capable of capturing long-distance dependencies between nodes in a social network. Although DeepWalk is expressive enough to capture these representations, it has limitations:

Not Guaranteed: Multi-scale representations are not necessarily captured, since they are not explicitly preserved by the objective function. In fact, since DeepWalk will always have more entries from A than from A^k, $(k > 1)$, it is *biased* towards representations that preserve the lowest power of A. To see this, observe that given any random walk of length L, the number of entries from A is at most $L - 1$. In general, the number of entries from A^k in a random walk is at most $\frac{L-1}{k}$.

This bias towards lower powers of the adjacency matrix is a fundamental weakness, when higher order powers are the appropriate representations for machine learning tasks on the network. For example, when classifying a feature correlated with large scale graph structure – such as language detection on a social network, a coarse-grained representation may offer performance benefits over a finer representation that preserves individual edges.

Global Representation: Different scales of representation are not independently accessible. That is, DeepWalk learns one global representation that conflates all possible scales of network relationships. This is undesirable, as each individual learning task needs to decode the relevant similarity information from the global representation. In actuality, an ideal representation for social relationships would span multiple scales, each one capturing successively broader levels of latent community memberships. Each learning task then, is able to utilize the best level of social relationships for its task. As we will show in Section 3.5, performance on each learning task can be maximized through a different scale of social representation.



(a) WALKLETS(A^1) **(b) WALKLETS(A^3)** **(c) WALKLETS(A^5)**

Figure 3.4: Heatmap of cosine distance from vertex v_{35} (shown by arrow) in the `Cora` network through a series of successively coarser representations. Close vertices are colored red, distant vertices are colored blue. Corresponding distributions of vertex distances are shown above in Figure 3.3.

WALKLETS: Multiple Scales of Random Walks

Building on observations discussed so far, we propose extending the model introduced by [117] to explictly model multiscale dependencies. Our method operates by factoring powers of the adjacency matrix A, using recently proposed algorithms for learning representations.

Similar to DeepWalk, we model the network through a series of truncated random walks started at each node. As discussed in [117], the co-occurrence of two vertices in these truncated random walks can model the rate of diffusion in the network. However, we make a key change to the sampling procedure. Specifically we choose to *skip* some of the the nodes in the random walk. In this way, we form a set of relationships which are sampled from successively higher powers of A. We denote WALKLETS$(A^1, \ldots, A^k)$ to represent WALKLETS derived from the $A^1, \ldots, A^k$ powers of the adjacency matrix respectively.

Each distinct power forms a corpus in a series of corpora which models a specific distance dependency in the network. This sampling procedure is illustrated in Figure 3.2.

After partitioning the relationships sampled by scale, we model the probability of observing a sample vertex v_i with a vertex v_j as in Eq.(3.3). This implies the following objective function in order to learn representations for each node $v_i \in \mathcal{V}$:

$$J = - \sum_{v_i, v_j \in C_k} \log Pr(v_i | v_j), \tag{3.7}$$

where C_k is the corpus of random walk pairs generated for representation scale k. J seeks to maximize the log likelihood of v_i co-occurring with context node v_j. This objective, commonly referred to as *Skipgram*, was first proposed for language modeling in [92], and first extended to network representation learning in [117].

Loss function Optimization

We optimize the loss function stated in 3.7 using standard back propagation with stochastic gradient descent. We use the default learning rate as 0.025 and set the size of the embedding d to 128 unless stated otherwise. This can be easily extended to weighted graphs (by adapting the gradient proportional to the weights). Furthermore techniques like edge sampling [131] can easily be adapted to our method.

Implicit matrix factorization view

By sampling in this way, we can show that we learn representations of different scales.

Lemma 1. *Using DeepWalk on skipped random walks, where the skip factor is set to k (we sample nodes that are a distance k from each other) implicitly factors a matrix derived from A^k.*

Proof. Observe that in a skipped random walk with a skip factor of k, each consecutive node pair v_i and v_{i+1} are reachable by path of length exactly k and therefore represent edges sampled from A^k.

When we provide DeepWalk with random walks where v_i co-occurs with v_j only if it is reachable by path of length k, in Equation 3.5, only the term corresponding to A^k is present. Therefore we implicitly factor a matrix derived from A^k. This completes the proof. □

Search Strategy

Related work [117, 131] have respectively advocated differing strategies for generating social relationships from edges in a graph. The *breath-first* strategy successively expands from a single node of interest and examines all its neighbors. This works well for local neighbors, but faces a state space explosion as higher levels of expansion (i.e. neighbors of neighbors, etc). The *depth-first* strategy uses a random walk, which encodes longer distance information, and may be better for learning higher order network representations.

We have presented our method by observing relationships of multiple scales through random walk sampling, which we believe would be more scalable. An alternative search strategy (possible for smaller graphs), is to directly compute A^k (i.e. all nodes with a path of length k to another) and use this to sample pairs of vertices.

3.3.2 Case study: `Cora`

In order to illustrate the effects of network representations at multiple scales, we visualize a small citation graph `Cora`, with 2,708 nodes and 5,429 edges.

Figure 3.3 shows a histogram of the distance from a particular node (v_{35}) to every other node's social representation. As we examine successively deeper social representations, we see that a group of *proximal* nodes develops. These nodes are part of the larger community of papers which v_{35} is a member of – specifically, the area of `Genetic Algorithms`. Should we perform classification in `Cora`, this clear separation of network structure enables easier generalization.

	% Labeled Nodes	10%	50%	90%
Micro-F1(%)	WALKLETS(A^1)	23.51	32.19	34.66
	WALKLETS(A^2)	**37.46****	**41.19***	**42.59**
	WALKLETS(A^3)	31.23	33.48	34.96
	WALKLETS(A^1, A^2)	24.88	33.23	35.81
	WALKLETS(A^1, A^3)	24.12	32.48	34.96
	WALKLETS(A^2, A^3)	**37.49****	**41.09**	**42.54**
	WALKLETS(A^1, A^2, A^3)	25.12	33.48	36.02
	DeepWalk	34.55	40.77	42.34
	LINE	23.65	34.67	37.44
	GraRep	37.05	40.95	42.31
Macro-F1(%)	WALKLETS(A^1)	12.98	17.81	18.74
	WALKLETS(A^2)	**21.65****	25.89	27.26
	WALKLETS(A^3)	13.32	16.08	17.08
	WALKLETS(A^1, A^2)	13.88	18.70	19.83
	WALKLETS(A^1, A^3)	13.30	18.03	19.03
	WALKLETS(A^2, A^3)	**21.72****	25.77	27.28
	WALKLETS(A^1, A^2, A^3)	14.07	18.93	19.96
	DeepWalk	20.44	26.31	27.71
	LINE	13.99	20.66	22.36
	GraRep	19.93	24.24	25.20

(a) `BlogCatalog`

	% Labeled Nodes	1%	5%	9%
Micro-F1(%)	WALKLETS(A^1)	47.20	52.29	53.41
	WALKLETS(A^2)	**55.67**	61.18	62.28
	WALKLETS(A^3)	**60.49****	**65.36****	**66.24****
	WALKLETS(A^1, A^2)	49.79	59.17	61.19
	WALKLETS(A^1, A^3)	53.61	62.72	64.48
	WALKLETS(A^2, A^3)	54.61	**63.60**	**65.33**
	WALKLETS(A^1, A^2, A^3)	53.98	63.37	**65.18**
	DeepWalk	54.93	63.45	65.16
	LINE	45.03	51.69	53.32
	GraRep	63.62	67.47	68.34
Macro-F1(%)	WALKLETS(A^1)	39.03	45.02	46.56
	WALKLETS(A^2)	48.20	56.02	57.64
	WALKLETS(A^3)	**53.49**	**60.92***	**62.28**
	WALKLETS(A^1, A^2)	47.35	55.51	57.35
	WALKLETS(A^1, A^3)	50.98	59.27	61.03
	WALKLETS(A^2, A^3)	52.17	**60.38**	61.99
	WALKLETS(A^1, A^2, A^3)	51.55	60.05	61.75
	DeepWalk	52.37	60.37	62.08
	LINE	42.73	48.49	49.91
	GraRep	59.10	63.83	64.97

(b) `DBLP`

Table 3.1: Multilabel classification results on `BlogCatalog` and `DBLP`. In general, specific higher order representations improve task performance. On `DBLP`, (a small, well behaved graph), the exact multiscale computation performed by GraRep outperforms WALKLETS's sampling approach. Numbers greater than DeepWalk are bolded. (*,) indicates statistically superior performance to DeepWalk at level of (0.05,0.001) using a standard paired t-test. The best performing scale of representation is highlighted in blue.**

This phenomenon is also illustrated by Figure 3.4, which shows a heatmap of distance to node v_{35} overlaid on the original network structure. Note how the distance at different scales of network representation encodes membership in successively larger communities.

3.4 Experimental Design

In this section, we analyze our method experimentally by applying our method to several online social networks. In particular, our experiments are motivated by two main goals: First, we seek to characterize the various multi-scale effects manifested in different real world social networks. Second, we evaluate the efficacy of our method in capturing their underlying network structure.

We briefly provide an overview of the different graphs we use in our experiments in Table 3.5 and below:

- `BlogCatalog` is a network consisting of relationships between bloggers. The labels indicate the topic categories associated with authors.
- `DBLP` is a co-author graph between researchers in computer science. The labels indicate the research areas that the researchers publish in (e.g. Artificial Intelligence, or Data Mining).
- `Flickr` is a network consisting of users on the photograph sharing website `Flickr`. An edge in the network indicates a contact relationship between the user pair. The labels indicate the interest groups of the users (e.g. *noir photography*)
- `YouTube` is a social graph between video enthusiasts. The labels indicate shared group memberships which users have in common (e.g. Anime videos).

Since our method learns latent representations of nodes in a social network in an unsupervised manner, these representations should generally serve as useful features for a variety of learning tasks. Therefore, we evaluate our method on one such important task – that of multi-label classification in social networks. This task is motivated by observing that nodes in a network exhibit memberships in many of the same groups as their friends. For example, people in a social network are members of several circles (family, alma-mater, employer, shared hobbies, etc). Modeling and predicting these varied group memberships is not only essential to understanding real world networks, but also has several important commercial applications (e.g. more effective ad targeting).

3.4.1 Baseline Methods

To compare with our approach, we consider three recently proposed models for social representation learning that represent the state of the art.

- **DeepWalk** [117]: This method learns representations of nodes using a sequence of truncated random walks. The learned representations capture a linear combination of community membership at multiple scales.
- **LINE** [131]: Similar to DeepWalk, this method learns node representations with the Skipgram objective function. For this baseline we use the LINE 2^{nd} method, which only considers immediate connections (those in A^1).
- **GraRep** [25]: This multi-scale method generates vertex representations by explicitly computing successive powers of the random walk transition matrix, and uses the SVD to reduce their dimensionality.

Name	$\lvert\mathcal{V}\rvert$	$\lvert E\rvert$	$\lvert\mathcal{Y}\rvert$	Labels
BlogCatalog	10312	333983	39	Topics
DBLP	29199	133664	4	Research Areas
Flickr	80513	5899882	195	Interests
Youtube	1138499	2990443	47	Groups

Figure 3.5: Summary of our datasets.

3.4.2 Multilabel classification

We evaluate our method using the same experimental procedure outlined in [117] We randomly sample T_f fraction of the labeled nodes and use them as training data with the rest being used as a test data set. This process is repeated 10 times, after which we report the mean MACRO-F1 and MICRO-F1 scores. This enables us to compare our method with other relevant baselines easily.

In all cases, we learn a logistic regression model (based on the *one vs rest strategy*) classification. We use a default $L2$ regularization penalty of $C = 1$ and use the optimization algorithm implemented by Liblinear [43].

For WALKLETS, we use only the representations generated from walks in $\pi \in \{1, 2, 3\}$. The number of walks N from each node in all cases was set to 1000 while the length of each such walk L is set to 11. The dimension of embeddings d was 128 in all cases (these settings are also the same for all baselines). In cases, where we use more than one representation from π, we concatenate all such features and use PCA to project them down to 128 dimensions.

With this methodology, we control for differences in sizes of training data, and hyper-parameters that determine the capacity of the representations. This allows for an interpretable comparison with other methods and baselines.

3.5 Experimental Results

In this section, we present our results of the multi-label classification task on the various datasets described in Section 3.4. Tables 3.1 and 3.2 show both the Micro-F1 and Macro-F1 performance of our algorithm and compares with baselines DeepWalk, LINE and GraRep.

BlogCatalog: Table 3.1a shows the multilabel classification results for BlogCatalog. We observe that WALKLETS using features from A^2 outperforms all baselines with respect to Micro-F1. When labeled data is sparse, (only 10% of the nodes labeled), the difference is statistically significant at the 0.001 level for both Micro-F1 and Macro-F1. Statistical significance was established using a paired t-test over the 10 different runs. This translates to an 8% in Micro-F1 and a 5% improvement in Macro-F1. On this dataset the representations capturing dependencies of length 2 (from A^2) perform better than those of length 1 or 3, which we show by highlighting them in blue.

DBLP: The results from experiments on DBLP are shown in Table 3.1b. We notice that the representations of A^3 provide a statistically significant improvement in Micro-F1 over DeepWalk and LINE. These coarser representations offer better encoding of the subject areas which an author publishes in.

We note that on this task however, WALKLETS fails to outperform the multiscale representation learned by GraRep. We attribute this to the fact that the DBLP is quite well behaved. First, it exhibits a highly homophilous behavior, as co-authorship guarantees similar attributes (a shared publication between two authors must be in the same research area). Second, co-authorship edges in the graph indicate a high degree of similarity (it is harder to create spurious edges). When such conditions hold, GraRep's direct computation of the random walk transition matrix can yield high performance gains. However, we note that GraRep's technique requires materializing a dense matrix, which is inherently unscalable – as illustrated by our remaining datasets.

Flickr: Table 3.2a shows the evaluation on the `Flickr` dataset. On this dataset, we see that features derived from A^2 offer the best performance on the task for Micro-F1 significantly outperforming all baselines. The combination of features from A^2 and A^3 offer the best performance for Macro-F1, also outperforming all baselines. Specifically we observe that when only 1% of the data is labeled for training WALKLETS outperforms DeepWalk by 4% in Micro-F1 scores and by 6% Macro-F1.

We note that the most competitive baseline, GraRep, fails to run on this dataset (which has only 80,513 vertices) as it runs out of memory.[2] Our method, instead, handles such large networks gracefully while yielding competitive results. We discuss this further in Section 3.6

Youtube: Our results for `YouTube` are presented in Table 3.2b. Once again, our method outperforms all baseline methods. Specifically, the representations learned from A^2 and A^3 each significantly outperform DeepWalk at the p=0.001 level in Micro-F1. Interestingly, the combined joint representation WALKLETS(A^2,A^3) offers the best performance with respect to Macro-F1, significantly outperforming DeepWalk at the 0.05 level. The best performing individual representations on this task came from A^3, (highlighted in blue). YOUTUBE contains many more vertices than any of the other datasets we have considered, and it is not surprising that GraRep again runs out of memory. We note that the online setting of WALKLETS allows learning of representations for graphs with millions of vertices.

3.6 Discussion

In this section we briefly discuss our results further. We start by addressing the effect of different representation scales on classification tasks, move to discussing the scalability of our approach, and finish with an analysis of our sampling procedure.

3.6.1 Multi-scale Effects in Classification

The experimental results presented in Section 3.5 show that no single representation scale provides the best performance across tasks. Providing explicit representations addresses a limitation shared by other baseline methods which do not explicitly encode information at multiple scales.

For both Micro-F1 and Macro-F1, features from A^2 performed best on two graphs (`BlogCatalog`, `Flickr`) at all variations of training data. This indicates that paths of length 2 are the appropriate social dependencies to model shared user interest (topic categories and photography interests,

[2]Our machine used for experiments was generously apportioned with 384GB RAM.

respectively). The neighbor-of-neighbor information captured in A^2 is especially useful to handle a network which is missing information. One important case where this occurs is the *cold start* problem, where a node has just joined the network and has not had the opportunity to make all of its connections yet.

Interestingly, representations derived from A^3 offered the best task performance on DBLP and YouTube. On these graphs, the classification performance monotonically increases with the scale of representation captured. We can surmise that the classification tasks in these graphs do exhibit hierarchical structure, and that the distinct higher-order representations created through our method allow exploitation of the correlation between a graph's hierarchy and a task's labels.

We emphasize here that our method WALKLETS explicitly models the multi-scale effects present in social networks, enabling a comprehensive analysis of scales that are most informative for a given learning task – something which is not possible using methods which learn global representations [117, 131], or which blend representations of different scales together [25].

3.6.2 Scalability

WALKLETS is an online algorithm which operates on pairs of vertices sampled at different dependency levels from the graph. This online approach approximates the higher-order transition matrices using sampling, which allows scaling to large network graphs with millions of vertices. This is in stark contrast to the closest related method, GraRep, which require the materialization of a dense matrix (A^k and similar matrices rapidly loose their sparsity as k grows, if the graph is connected and the edge distribution follows a power law). The explicit need to compute the successive powers of the transition matrix is further complicated by GraRep's dependence on a SVD – a computation that does not necessarily scale well to large networks.

3.6.3 Sampling Analysis

Here we analyze the effectiveness of our proposed random walk sampling procedure by comparing it to the explicit and exact computation made by GraRep. Given a graph G, we use GraRep to explicitly compute the matrix M_{GR} it factorizes. We then use the random walks used to learn WALKLETS embeddings to estimate the matrix WALKLETS factorizes M_W. We estimate how close our approximation is to the exact computation M_{GR} by $Err = abs(M_{GR} - M_W)$. Parameter settings correspond to those described in Section 3.4.2.

In Table 3.6, we report the mean error for an edge (i, j) we observe in Err for multiple scales on the two graphs small enough to explicitly compute the random walk transition matrix: DBLP and BlogCatalog. We observe that the mean error and the standard deviation of our approximation at various scales is low, indicating that our method captures a reasonable approximation of the random walk transition matrix. Increasing the sample size (by adding random walks) will result in increasingly better approximations.

	BlogCatalog Err_{ij}		DBLP Err_{ij}	
	Mean	Std.Dev	Mean	Std.Dev
A^1	0.000093	0.003697	0.000014	0.001685
A^2	0.000131	0.000764	0.000018	0.001202
A^3	0.000148	0.000448	0.000020	0.001064

Figure 3.6: Mean and Standard Deviation of errors observed from sampling the transition matrices using WALKLETS as compared to exact computation.

3.7 Related Work

The related work falls into three categories: unsupervised feature learning, multiscale graph clustering, and graph kernels.

Unsupervised Feature Learning Recently proposed methods for social representation learning [117, 131] use neural network losses proposed by [92] to learn representations which encode vertex similarity. Unlike our work, these methods do not explicitly preserve the multiscale effects which occur in networks. The closest related work to ours [25], explicitly models multiscale effects in networks through an SVD on the random walk transition matrix. Our independent work uses sampling and online learning to operate on much larger corpora. In addition our work preserves multiple scales of representation, which can provide both better task performance and modeling insight.

Distributed representations have also been proposed to model structural relationships in diverse fields such as computer vision [70], speech recognition [36], and natural language processing [5, 33].

Multiscale Community Detection Many techniques for detecting multiple scales of discrete communities in graphs have been proposed [1, 73, 122]. In general, unlike our method, these approaches seek to return a hard clustering which describes the hierarchy of social relationships in a graph.

Graph Kernels Graph Kernels [142] have been proposed as a way to use relational data as part of the classification process, but are quite slow unless approximated [67]. Recent work has applied neural networks to learn subgraph similarity for graph kernels [149].

3.8 Conclusion

In this work, we introduce multi-scale network representations to specifically capture network structure at multiple resolutions. The online algorithm we propose, WALKLETS, uses the offsets between vertices observed in a random walk to learn a series of latent representations, each of which captures successively larger relationships. WALKLETS utilizes connections between neural representation learning algorithms and matrix factorization to theoretically underpin the matrices implicitly factored at each scale. We demonstrate empirically that WALKLETS's latent representations encode various scales of social hierarchy, and improves multi-label classification results over previous methods on a variety of real world graphs. In addition to strong performance, WALKLETS scales gracefully to arbitrarily large networks.

We believe that WALKLETSś explicit handling of multiple scales of representations allows better comprehension of each network's nuances, and that it lays a strong foundation for developing future multi-scale network approaches. Our further investigations in this area will seek to develop additional theoretical backing for our methods.

	% Labeled Nodes	1%	5%	9%
Micro-F1(%)	WALKLETS(A^1)	24.37	29.99	31.64
	WALKLETS(A^2)	**32.47****	**37.41****	**38.70****
	WALKLETS(A^3)	**31.44**	34.87	35.83
	WALKLETS(A^1, A^2)	25.48	31.35	33.03
	WALKLETS(A^1, A^3)	25.11	30.86	32.51
	WALKLETS(A^2, A^3)	**32.52****	**37.39****	**38.66****
	WALKLETS(A^1, A^2, A^3)	25.93	31.80	33.47
	DeepWalk	31.18	36.64	38.18
	LINE	25.06	30.55	32.85
	GraRep	-	-	-
Macro-F1(%)	WALKLETS(A^1)	8.92	14.29	16.30
	WALKLETS(A^2)	**14.23**	**22.57***	**24.99**
	WALKLETS(A^3)	10.82	17.15	19.22
	WALKLETS(A^1, A^2)	9.70	15.50	17.57
	WALKLETS(A^1, A^3)	9.38	14.96	17.00
	WALKLETS(A^2, A^3)	**14.68**	**22.7****	**25.08***
	WALKLETS(A^1, A^2, A^3)	10.03	15.9	17.99
	DeepWalk	13.84	22.26	24.81
	LINE	9.16	16.04	18.77
	GraRep	-	-	-

(a) `Flickr`

	% Labeled Nodes	1%	5%	9%
Micro-F1(%)	WALKLETS(A^1)	30.97	38.09	39.63
	WALKLETS(A^2)	**38.55****	**43.33****	**44.33****
	WALKLETS(A^3)	**38.91****	**43.73****	**44.77****
	WALKLETS(A^1, A^2)	28.46	33.74	36.73
	WALKLETS(A^1, A^3)	28.43	33.72	36.67
	WALKLETS(A^2, A^3)	**37.19***	**40.73****	**42.14****
	WALKLETS(A^1, A^2, A^3)	29.55	34.70	37.75
	DeepWalk	36.17	39.68	41.49
	LINE	33.21	36.94	39.19
	GraRep	-	-	-
Macro-F1(%)	WALKLETS(A^1)	15.47	26.24	28.87
	WALKLETS(A^2)	24.33	32.30	**34.38**
	WALKLETS(A^3)	24.74	**32.90**	**35.02****
	WALKLETS(A^1, A^2)	22.71	27.41	29.30
	WALKLETS(A^1, A^3)	22.66	27.36	29.21
	WALKLETS(A^2, A^3)	**29.50***	**33.61****	**34.62***
	WALKLETS(A^1, A^2, A^3)	23.93	28.48	30.48
	DeepWalk	28.57	32.82	34.30
	LINE	26.11	31.29	33.04
	GraRep	-	-	-

(b) `YouTube`

Table 3.2: Multi-label classification results in `Flickr` and `Youtube`. Higher order WALKLETS representations outperforms all scalable competitors on these graphs. The most competitive baseline (GraRep) is unable to run on graphs with millions of vertices. Numbers greater than DeepWalk are bolded. (*,) indicates statistically superior performance to DeepWalk at level of (0.05,0.001) using a standard paired t-test. The best performing scale of representation is highlighted in blue.**

Chapter 4

Scalable Anomaly Ranking of Attributed Neighborhoods

4.1 Introduction

Graph anomaly detection [4] is a problem of pressing concern, with broad applications including network security, spam/fraud detection (in social networks, financial networks, etc.), database integrity, and more.

The essence of graph anomaly detection lies in determining an appropriate *anomaly score*, which effectively characterizes the quality of each neighborhood. Quantifying the quality of graph neighborhoods has long been a research area of interest on its own, where it finds additional applications in network community detection and graph partitioning. Most of the existing approaches focus on unattributed or plain graphs and hence only utilize structure.

These structural approaches can be divided by whether they focus on the internal characterizations of a neighborhood, its separability from its boundary, or a combination of both. Measures which solely use internal information judge quality on intra-group connectivity statistics (e.g. average degree, edge density) and have been used for dense subgraph mining [28, 139] and anomaly detection [2]. In contrast to internal measures, boundary-centric measures (e.g. expansion, cut-ratio) define a neighborhood's quality only in terms of how separable it is from the rest of the graph. Perhaps the most popular measures are those which characterize a neighborhood by both its internal characteristics and its boundary (e.g. modularity [106] and conductance [8]).

In contrast to numerous measures that focus solely on structural quality, there exist only a few attempts that also utilize attributes. For example, several methods aim to find communities in attributed graphs that not only are dense but also exhibit attribute coherence [3, 47, 56]. Most recently, others also quantify the connectivity at the boundary to find outlier nodes or subgraphs in attributed graphs [58, 114].

In this work, we propose a novel approach to detecting anomalous neighborhoods in attributed graphs. We define a neighborhood to be high quality when its nodes are (1) internally well

This work originally appeared as "Bryan Perozzi and Leman Akoglu. Scalable anomaly ranking of attributed neighborhoods. In *Proceedings of 2016 SIAM International Conference on Data Mining*, SDM '16, 2016."

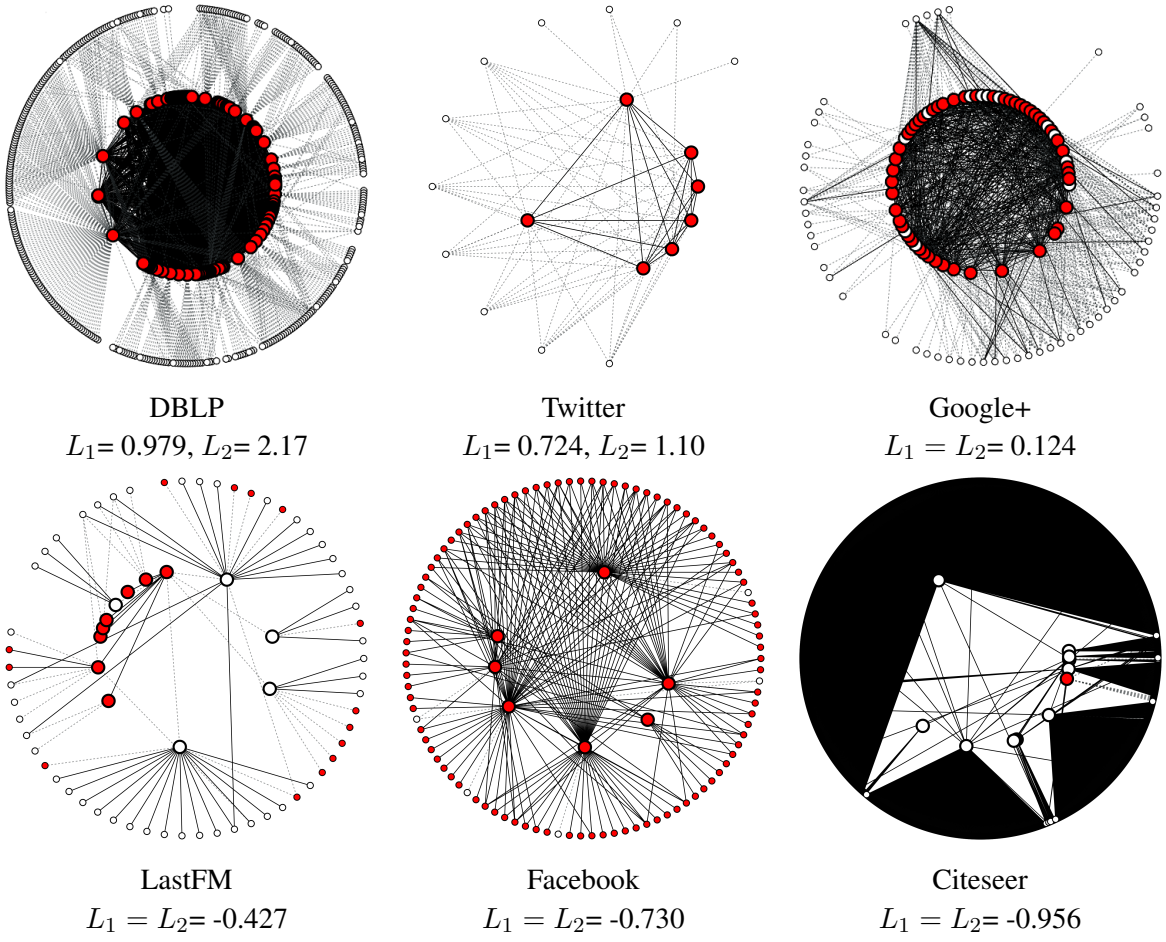


Figure 4.1: Example neighborhoods (inner circles) and their boundaries (outer circles) in our real-world graphs from high (top-left) to low (bottom-right) `normality` **scores. Colors depict presence (red) or absence (white) of the attribute that maximizes** `normality` **for each neighborhood. Dashed edges are "exonerated". Results discussed further in Section 4.5.1.**

connected and similar to each other on a specific attribute subspace (we call these shared attributes the *neighborhood focus*), as well as (2) externally well separated from and/or dissimilar to the nodes at the boundary. Based on this definition, we introduce a new measure called `normality` to quantify the quality of attributed neighborhoods, which carefully utilizes both structure and attributes together to quantify their internal consistency within, as well as external separability at the boundary. (See Figure 4.1 for examples of high-to-low `normality` neighborhoods from various real-world graphs.) We note that the focus attributes of the neighborhoods may be latent and unknown a priori, especially in high dimensions. Our method AMEN (for Anomaly Mining of Entity Neighborhoods) automatically *infers* the focus attributes and their respective weights, so as to maximize the normality score of a neighborhood. Neighborhoods with low normality scores, i.e., for which a focus that yields high `normality` cannot be found, are considered low quality or anomalous. The main contributions of our work are summarized as follows:

- **Neighborhood Quality Score**: We propose `normality`, a new measure to quantify the quality of the structure (topology) as well as the focus (attributes) of neighborhoods in attributed graphs. Intuitively, `normality` quantifies the extent which a neighborhood is "coherent"; i.e., (i) internally consistent and (ii) externally separated from its boundary.
- **Neighborhood Anomaly Mining**: We formulate and solve a novel anomaly mining task for attributed graphs. Our proposed method, AMEN, discovers a given neighborhood's *latent* focus through the unsupervised maximization of its `normality`. Those neighborhoods for which a proper focus cannot be identified receive low score, and are deemed as anomalous.
- **Scalable Optimization**: Our proposed measure lends itself to an efficient convex optimization procedure. This allows us to analyze real-world graphs with millions of attributes.

Experiments on real-world attributed graphs show the effectiveness of our approach. Specifically, we show the utility of our measure in spotting anomalies in attributed graphs, where AMEN outperforms existing approaches including conductance, density, OddBall [2], and SODA [58] by 16%-25% mean precision. Furthermore, we qualitatively analyze the high and low `normality` neighborhoods, and show how our method can effectively contrast differences in correlation between structure and attributes across graphs.

4.2 Problem Statement

We consider the ranking problem of entity neighborhoods in a graph with node attributes by quality. More formally, let $G = (\mathcal{V}, \mathcal{E}, \mathcal{A})$ denote an attributed graph with $|\mathcal{V}| = n$ nodes, $|\mathcal{E}| = m$ edges, and $|\mathcal{A}| = d$ node attributes (or features). A neighborhood of G is defined as a set of nodes $C \subseteq \mathcal{V}$ and the edges among them, $(i, j) \in \mathcal{E},\ \{i, j\} \in C$. We denote by $B \subseteq \mathcal{V}$ the set of boundary nodes that are outside the neighborhood but have at least one edge to some node in the neighborhood, i.e., $(c, b) \in \mathcal{E}, c \in C, b \in B, C \cap B = \emptyset$.

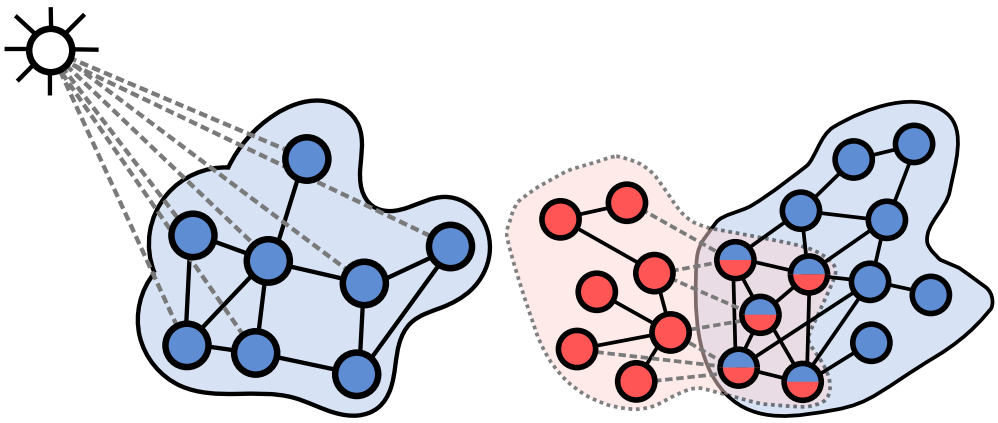
We consider a neighborhood to be of high quality based on two criteria: (i) internal consistency, and (ii) external separability. Intuitively, a good neighborhood has many internal edges among its members where they share a set of attributes with similar values. In other words, a common set of *focus* attributes makes the neighborhood members highly similar. In addition, a good neighborhood has either only a few edges at its boundary or many of the cross-edges can be "exonerated", that is, the focus attributes that make the neighborhood members similar to one another also make them dissimilar to or separated from their boundary nodes.

In summary, the *Anomaly Mining of Entity Neighborhoods* (AMEN) problem is given as follows:

Given a set of neighborhoods $\mathcal{C} = \{C_1, \ldots, C_k\}$ from a graph $G = (\mathcal{V}, \mathcal{E}, \mathcal{A})$ with node attributes;

Define a measure to quantify the quality of C_i's based on internal connectivity, boundary B_i, and attributes $\mathcal{A}$,

Find the neighborhoods of lowest quality.



(a) "Hub node" effect (b) Neighborhood overlap

Figure 4.2: High-quality neighborhoods *can* have bad cuts (i.e., many boundary-edges) due to (a) hub nodes or (b) neighborhood overlap. Our `normality` measure carefully utilizes a null model in (a) and node attributes in (b) to exonerate such edges (dashed lines).

4.3 Neighborhood Quality in Attributed Graphs

In this work we propose `normality`, a new measure of neighborhood quality based on a combination of internal connectivity, external separability, and attributes. Our measure is motivated by two properties of real world networks: (i) that there are no good cuts in real world graphs [76], and (ii) that most user-defined social circles overlap with each other [87]. Together, these imply the existence of (many) cross-edges between neighborhoods and at their boundary. Figure 4.2 illustrates two specific scenarios that could drive the emergence of many cross-edges at the boundary of high-quality neighborhoods in real-world attributed graphs.

The first scenario is where the cross-edges are due to hub nodes in real graphs. Consider Figure 4.2 (a) as an example, which shows a well structured neighborhood and a hub node in the host graph. Notice that the hub node connects to a considerable fraction of the nodes in the neighborhood, creating many cross-edges at its boundary. These edges, however, are *not surprising*. In fact, they are expected—as hub nodes by definition connect to a large body of nodes in a graph. While the quality of such a neighborhood is diminished e.g., based on conductance, our measure exonerates those edges as unsurprising under a null model, and does not penalize the neighborhood's `normality` score.

Another scenario where good neighborhoods have many cross-edges at their boundary is when the neighborhoods overlap. An example is given in Figure 4.2 (b) where two overlapping neighborhoods are shown. Good neighborhoods have many internal edges among their nodes, which implies that overlap nodes have many edges to non-overlap nodes in both neighborhoods. This in turn creates many cross-edges for both neighborhoods at their boundary. These edges, however, are driven by the internal density and should not affect their (external) quality. Provided that overlapping neighborhoods have sufficiently different *focus* that makes them separable (e.g., football vs. chess group), `normality` exonerates such cross-edges based on the attribute dissimilarity of boundary nodes to internal nodes. In contrast, measures that ignore attributes (e.g., cut-ratio, conductance,

etc.) penalize these cross-edges irrespectively. Those measures are expected to perform poorly for graphs with many overlapping communities such as social networks.

These scenarios illustrate that using both structure and attributes is meaningful and necessary to quantify the quality of neighborhoods in real graphs. Our `normality` measure is unique in its notion of "exoneration" of edges at the boundary, which none of the existing measures exhibit. Rather, they either completely ignore the boundary or equally penalize all the boundary edges irrespective of context.

4.3.1 Preliminaries

Our `normality` measure is inspired by modularity and assortativity, which we first briefly describe here.

4.3.2 Modularity

Newman's modularity [106] is a measure of network structure that quantifies the extent which the network divides into modules (a.k.a. clusters, communities, groups, circles, etc.). Networks with high modularity have dense connectivity among the nodes within communities, but sparse connectivity between nodes from different communities. Specifically, modularity is written as

$$M = \frac{1}{2m}\sum_{ij}(A_{ij} - \frac{k_i k_j}{2m})\delta(c_i, c_j) \ , \tag{4.1}$$

where A is the adjacency matrix of graph G, c_i denotes the community assignment of node i, k_i denotes its degree, and $\delta(\cdot)$ is the indicator function, with value 1 if two nodes belong to the same community and 0 otherwise. Modularity then, is the difference between the actual and the expected fraction of edges between nodes in the same community. The larger the difference, the more modular is the network.

4.3.3 Assortativity

While modularity is defined for non-attributed graphs and solely quantifies structure, a similar formula called *assortativity* has been used to measure homophily in attributed networks [105]. Homophily is the extent which the same type of nodes connect to one another, e.g., in social networks [90]. Specifically, for a graph in which every node is associated with a single, *nominal/categorical* attribute (e.g., gender, nationality, etc.), its assortativity is

$$H^{(\text{nom})} = \frac{1}{2m}\sum_{ij}(A_{ij} - \frac{k_i k_j}{2m})\delta(a_i, a_j) \ , \tag{4.2}$$

where this time a_i depicts the attribute value of node i (e.g., what nationality i belongs to). As such, assortativity is the difference between the actual and the expected fraction of edges between nodes of the same type.

In a perfectly mixed network, all edges fall between nodes of the same type, and assortativity is maximum. It takes negative values for disassortative networks, and is 0 for networks in which attributes and structure are uncorrelated.

4.3.4 Scalar Assortativity

Eq. (4.2) is for networks with a nominal/categorical attribute a, such as gender, nationality, etc. For a *numerical/scalar* attribute x, like income, age, etc., one can derive a corresponding formula using the co-variance of the attribute values among connected nodes. Specifically, $cov(x_i, x_j) = \frac{1}{2m}\sum_{ij} A_{ij}(x_i - \mu)(x_j - \mu)$, where $\mu = \frac{1}{2m}\sum_i k_i x_i$ is the mean value of attribute x over the edge-ends, and k_i denotes the degree of node i (note that the average here is over the edges rather than the nodes). From $cov(x_i, x_j)$ one can derive the assortativity for numerically attributed networks (See [107] p.228) as

$$cov(x_i, x_j) = H^{(\text{num})} = \frac{1}{2m}\sum_{ij}(A_{ij} - \frac{k_i k_j}{2m})x_i x_j \ , \tag{4.3}$$

where assortativity is positive when x_i, x_j are both small or both large (w.r.t. the mean), and is negative if they vary in opposite directions. Zero assortativity means the attributes of connected nodes are uncorrelated.

4.3.5 Modularity vs. Assortativity

The specific applications that leverage these two measures have traditionally been different. Modularity is often used as an objective function in community detection and graph partitioning [20, 31, 106, 126]. Assortativity, on the other hand, has often been used in measuring homophily in social science studies, e.g., in analyzing how school children of different races and genders interact [96], and how people from various nationalities are segregated in residential areas [86].

Nevertheless, despite the differences between modularity and assortativity, the two quantities are related. It was observed that assortative networks are likely more modular and tend to break into communities in which "like is connected to like" [108]. In other words, one can think of assortativity as a driving force of modular structure in networks—one that influences the emergence of communities.

4.3.6 Proposed Measure

Next we formally introduce `normality`, our new measure to quantify the quality of attributed neighborhoods. `Normality` is inspired by Newman's modularity and assortativity [105, 106] but exhibits key differences. First, `normality` utilizes *both structure and attributes* together. Second, its formulation generalizes to graphs with *multiple* node attributes (as opposed to assortativity which is defined only for a single node attribute). Our measure is built upon two intuitive criteria that define a high quality attributed neighborhood, (1) internal consistency and (2) external separability.

Internal consistency:

To quantify the internal consistency of a neighborhood, we propose to generalize scalar assortativity (Eq. (4.3)) which is defined over a graph with a single attribute to a set of nodes with multiple attributes. The internal consistency I of a neighborhood C is then written as

$$\begin{aligned} I = cov(\mathbf{x_i}, \mathbf{x_j}) &= \sum_{f=1}^{|\mathcal{A}|} \Big(\sum_{i \in C, j \in C} \big(A_{ij} - \frac{k_i k_j}{2m}\big) \mathbf{x_i}(f) \mathbf{x_j}(f) \Big) \\ &= \sum_{i \in C, j \in C} \big(A_{ij} - \frac{k_i k_j}{2m}\big) \sum_{f=1}^{|\mathcal{A}|} \mathbf{x_i}(f) \mathbf{x_j}(f) \\ &= \sum_{i \in C, j \in C} \big(A_{ij} - \frac{k_i k_j}{2m}\big) \ \ \mathbf{x_i} \cdot \mathbf{x_j} \end{aligned} \tag{4.4}$$

where A is the adjacency matrix of graph G, k_i denotes node i's degree, and $\mathbf{x_i}$ is the *vector* of attributes that node i is associated with.

We remark that $\mathbf{x_i} \cdot \mathbf{x_j}$ translates to the dot-product similarity between the attribute vectors of neighborhood members. However, it treats all of the attributes as equally important in quantifying the similarity among nodes. In general, it is more reasonable to assume that the neighborhood members come together around *a few* common attributes (e.g., same school and same hobby). This is expected especially in very high dimensions. We refer to those attributes upon which neighborhood members agree, i.e. have similar values, as the *focus* (attributes) of a neighborhood.

Therefore, we modify the internal consistency score by introducing a non-negative weight vector $\mathbf{w}$ to compute the weighted node similarities as

$$I = \sum_{i \in C, j \in C} \big(A_{ij} - \frac{k_i k_j}{2m}\big) \ s(\mathbf{x_i}, \mathbf{x_j} | \mathbf{w}) \ , \tag{4.5}$$

for similarity $s(\mathbf{x_i}, \mathbf{x_j}|\mathbf{w}) = \mathbf{w}^T \cdot (\mathbf{x_i} \circ \mathbf{x_j})$, where $\circ$ denotes the Hadamard (element-wise) product, and $\mathbf{w}$ is a vector with the attribute weights. This corresponds to weighted dot-product similarity, $(\mathbf{w}^{1/2} \circ \mathbf{x_i})^T \cdot (\mathbf{w}^{1/2} \circ \mathbf{x_j})$. We expect $\mathbf{w}$ to be *sparse*, in which only a few attributes corresponding to the neighborhood *focus* have large values and zero elsewhere.

Note that each neighborhood C is associated with its own weight vector $\mathbf{w_C}$, i.e. focus, which is potentially different across neighborhoods. Moreover, the attribute weights are often latent. For defining our quality criteria we can assume $\mathbf{w}$ is known. Later in Section 4.4 we will show that thanks to our formulation, we can infer this weight vector so as to make a given neighborhood as internally consistent and externally well-separated as possible. In the following we discuss the properties captured by Eq. (4.5).

First, notice that the internal consistency is decreased by missing edges inside a neighborhood, as $A_{ij} = 0$ for $(i, j) \notin \mathcal{E}$. Second, the existence of an edge is rewarded as much as the "surprise" of the edge. Specifically, $\frac{k_i k_j}{2m}$ denotes the probability that two nodes of degrees k_i and k_j are connected to each other by chance in a random network with the same degree distribution as the original graph [106]. As such, we define the surprise of an edge $(i, j) \in \mathcal{E}$ as $(1 - \frac{k_i k_j}{2m})$. The smaller $\frac{k_i k_j}{2m}$ is for

an existing edge inside a neighborhood, the more surprising it is and the more it contributes to the quality of the neighborhood.

These two properties quantify the *structure* of the neighborhood. On the other hand, the similarity function quantifies the *attribute* coherence. As a result, the more similar the neighborhood nodes can be made by some choice of $\mathbf{w}$, the higher I becomes. If no such weights can be found, internal consistency reduces even if the community is a complete graph with no missing edges.

Overall, a neighborhood with (1) many existing and (2) "surprising" internal edges among its members where (3) (a subset of) attributes make them highly similar receives a high internal consistency score.

External separability:

Besides being internally consistent, we consider a neighborhood to be of high quality if it is also well-separated from its boundary. In particular, a well-separated neighborhood either has (1) few cross-edges at its boundary, or (2) many cross-edges that can be "exonerated". A cross-edge $(i, b) \in \mathcal{E}$ ($i \in C, b \in B$) is exonerated either when it is unsurprising (i.e., expected under the null model) or when internal node i is dissimilar to boundary node b based on the focus attribute weights. The latter criterion ensures that what makes the neighborhood members similar to one another does not also make them similar to the boundary nodes, but rather differentiates them. The external separability E of a neighborhood C is then

$$E = - \sum_{\substack{i \in C, b \in B, \\ (i,b) \in \mathcal{E}}} \big(1 - \min(1, \frac{k_i k_b}{2m})\big)\ s(\mathbf{x_i}, \mathbf{x_b}|\mathbf{w})\ \leq 0\ . \tag{4.6}$$

External separability considers only the boundary edges and quantifies the degree that these cross-edges can be exonerated. As discussed earlier, cross-edges are exonerated in two possible ways. First, a cross-edge may be unsurprising; in which case the term $(1 - \min(1, \frac{k_i k_b}{2m}))$ becomes small or ideally zero (recall Fig. 4.2 (a) scenario). Second, the boundary node of a cross-edge may not share the same focus attributes with the internal node; in which case the term $s(\mathbf{x_i}, \mathbf{x_b}|\mathbf{w})$ becomes small or ideally zero (recall Fig. 4.2 (b) scenario). The higher the number of cross-edges that can be exonerated, the larger E (note the negative sign) and hence the quality of a neighborhood becomes.

Note that good neighborhoods by `normality` differ from quasi-cliques for which only internal quality measures, such as density [139] or average degree [28], are defined. Different from those and besides internal consistency, we also quantify the quality of the boundary of a neighborhood. `Normality` is also different from popular measures that do quantify the boundary, such as cut-ratio [45], modularity [106] or conductance [8], for which good neighborhoods are expected to have only a few cross-edges. In contrast, our formulation allows for *many* cross-edges *as long as* they are either (i) unsurprising (under the null model) or if surprising, (ii) can be exonerated by the neighborhood *focus*. These advantages arise as we utilize both structure and attributes in a systematic and intuitive way to define our measure.

4.3.7 Normality

Having defined the two criteria for the quality of a neighborhood, `normality` (N) is written as the sum of the two quantities I and E, where high quality neighborhoods are expected to have both high internal consistency and high external separability.

$$N = I + E = \sum_{i \in C, j \in C} \big(A_{ij} - \frac{k_i k_j}{2m}\big) s(\mathbf{x_i}, \mathbf{x_j}|\mathbf{w}) - \sum_{\substack{i \in C, b \in B \\ (i,b) \in \mathcal{E}}} \big(1 - \min(1, \frac{k_i k_b}{2m})\big) s(\mathbf{x_i}, \mathbf{x_b}|\mathbf{w}) \tag{4.7}$$

For a neighborhood with the highest `normality`, all the possible internal edges exist and are also surprising for which pairwise similarities are high. These ensure that the first term is maximized. Moreover, the neighborhood either has no cross-edges or the similarity or surprise of existing cross-edges to the boundary nodes are near zero, such that the second term vanishes. Neighborhoods of a graph for which the `normality` takes negative values are of lesser quality and deemed as anomalous.

Choice of similarity function: To this end, we considered the node attributes to be scalar variables where $s(\mathbf{x_i}, \mathbf{x_j}|\mathbf{w})$ is the weighted dot-product similarity. If the attributes are categorical (e.g., location, occupation, etc.), one can instead use the Kronecker delta function $\delta(\cdot)$ that takes the value 1 if two nodes exhibit the same value for a categorical attribute and 0 otherwise.

The choice of the similarity function is especially important for binary attributes (e.g., likes-biking, has-job, etc.). While those can be thought of as categorical variables taking the values $\{0, 1\}$, using Kronecker δ becomes undesirable for nodes inside a neighborhood. The reason is, internal consistency by the δ function is the same both when all the neighborhood nodes exhibit a particular binary attribute (all 1) and when none does (all 0). However, one may not want to characterize a neighborhood based on attributes that its members do not exhibit even if the agreement is large. Therefore, we propose to use dot-product for computing internal consistency and Kronecker δ for computing external separability for binary-attributed graphs.

4.4 Anomaly Mining of Entity Neighborhoods

As presented so far, when given a neighborhood C of an attributed graph and vector $\mathbf{w}$ of attribute weights, we can directly compute its `normality` using Eq. (4.7). However, for the task of anomaly mining, the *focus* of a neighborhood is latent and hard to guess without any prior knowledge. This is especially true in high dimensions where most attributes are irrelevant, making a uniform attribute weight vector impractical. Moreover, even if the neighborhood focus is known a priori, it is hard to assign weights to those attributes beyond that of binary relevance.

In this section, we propose an optimization approach to automatically *infer* the attribute weight vector for a given neighborhood, as the vector that maximizes its normality score. That is, we aim to identify a subspace that would make the neighborhood's `normality` as high as possible. All neighborhoods can then be ranked based on their (best possible) normality scores, and those with

Table 4.1: Real-world graphs used in this work. * depicts datasets with ground truth circles. n: number of nodes, m: number of edges, d: number of attributes, $|\mathcal{C}|$: number of circles, $|S|$: average circle size.

Name	$n = \|\mathcal{V}\|$	$m = \|\mathcal{E}\|$	$d = \|\mathcal{A}\|$	$\|\mathcal{C}\|$	$\|S\|$	nodes	edges	attributes
*Facebook	4,039	88,234	42-576	193	21.93	users	friendships	user profile information
*Twitter	81,306	1,768,149	1-2,271	4,869	12.51	users	follow relations	hashtags and user mentions
*Google+	107,614	13,673,453	1-4,122	479	134.75	users	friendships	user profile information
DBLP	108,030	276,658	23,285	n/a	n/a	authors	co-authorships	title words used in articles
Citeseer	294,104	782,147	206,430	n/a	n/a	articles	citations	abstract words used in articles
LastFM	272,412	350,239	3,929,101	n/a	n/a	users	friendships	music pieces listened to

lowest scores can be deemed anomalous. This allows us to restate our original problem in Section 4.2 as follows:

Given a set of neighborhoods $\mathcal{C}$ and `normality` N;
Find the attribute weight vector $\mathbf{w_{C_i}}$ which maximizes $N(C_i)$ for each neighborhood $C_i \in \mathcal{C}$,
Rank neighborhoods $\mathcal{C}$ by normality score,
Find the neighborhoods of lowest quality.

4.4.1 Neighborhood Focus Extraction

Our goal is to find an attribute weight vector (hereafter called $\mathbf{w_C}$) for a neighborhood C, which makes its `normality` as high as possible, such that connected nodes in the neighborhood are very similar and the nodes at the boundary are dissimilar. To this end, we leverage our `normality` to formulate an objective function parameterized by the attribute weights. This objective also has the nice property of quantifying structure, by penalizing non-existing in-edges and surprising cross-edges. Our formulation for focus extraction is then $\mathbf{max_{w_C}}\ N(C)$, which by reorganizing the terms that do not depend on $\mathbf{w_C}$, can be rewritten (based on Eq. (4.7)) as

$$\max_{\mathbf{w_C}} \quad \mathbf{w_C}^T \cdot \left[\sum_{i \in C, j \in C} \left(A_{ij} - \frac{k_i k_j}{2m}\right) s(\mathbf{x_i}, \mathbf{x_j}) - \sum_{\substack{i \in C, b \in B \\ (i,b) \in \mathcal{E}}} \left(1 - \min(1, \frac{k_i k_b}{2m})\right) s(\mathbf{x_i}, \mathbf{x_b}) \right]$$

$$\max_{\mathbf{w_C}} \quad \mathbf{w_C}^T \cdot (\mathbf{x}_I + \mathbf{x}_E) \tag{4.8}$$

where $\mathbf{x}_I$ and $\mathbf{x}_E$ are vectors that respectively denote the first and the second summation terms. Note that these can be directly computed from data. Moreover, the similarity function $s(\mathbf{x_i}, \mathbf{x_j})$ can be replaced by either $(\mathbf{x_i} \circ \mathbf{x_j})$ or $\delta(\mathbf{x_i}, \mathbf{x_j})$ depending on the type of the node attributes.

4.4.2 Size-invariant Scoring

The normality score in Eq. (4.8) grows in magnitude with the size of the neighborhood C being considered. Normalization is desirable then, in order to compare across differences in both neighborhood and boundary size.

We note that I is the maximum in the case of a fully connected neighborhood the members of which all agree upon the focus attributes. Therefore, $I_{\max} = |C|^2$, where $s_{\max}(\mathbf{x_i}, \mathbf{x_j}) = 1$ provided

that the attributes $\mathbf{x_i}(f)$ are normalized to $[0, 1]$ for each node i. On the other hand the minimum is negative, when there exists no internal edges and pairwise similarities are maximum. That is, $I_{\min} = \sum_{i \in C, j \in C} -\frac{k_i k_j}{2m}$. To normalize the internal consistency I, we subtract $I_{\min}$ and divide by $I_{\max} - I_{\min}$, which is equivalent to a weighted version of edge density.

To normalize external separability, we derive a measure similar to conductance [8], i.e., ratio of boundary or cut edges to the total volume (sum of the degrees of the neighborhood nodes). The difference is that each edge is weighted based on its surprise and the similarity of its end nodes. In particular, we define $\mathbf{x}_{\tilde{I}} = \sum_{\substack{i \in C, j \in C \\ (i,j) \in \mathcal{E}}} \big(1 - \min(1, \frac{k_i k_j}{2m})\big) s(\mathbf{x_i}, \mathbf{x_j})$. Note that similar to E, $\tilde{I}$ considers only the existing edges in the graph. Therefore, $\tilde{I} - E$ can be seen as the total weighted volume of the neighborhood.

Overall, we scale our measure as follows, where the division of the vectors in the second term is element-wise. As such, $\hat{\mathbf{x}}_I(f) \in [0, 1]$ and $\hat{\mathbf{x}}_E(f) \in [-1, 0]$.

$$\hat{N} = \mathbf{w_C}^T(\hat{\mathbf{x}}_I + \hat{\mathbf{x}}_E) = \mathbf{w_C}^T\left(\frac{\mathbf{x}_I - I_{\min}}{I_{\max} - I_{\min}} + \frac{\mathbf{x}_E}{\mathbf{x}_{\tilde{I}} - \mathbf{x}_E}\right)$$

4.4.3 Objective Optimization

The normalized objective function can be written as

$$\begin{aligned} \max_{\mathbf{w_C}} \quad & \mathbf{w_C}^T \cdot (\hat{\mathbf{x}}_I + \hat{\mathbf{x}}_E) \\ \text{s.t.} \quad & \|\mathbf{w_C}\|_p = 1, \;\; \mathbf{w_C}(f) \geq 0, \; \forall f = 1 \ldots d \end{aligned} \tag{4.9}$$

Note that we introduce a set of constraints on $\mathbf{w_C}$ to fully formulate the objective. In particular, we require the attribute weights to be non-negative and that $\mathbf{w_C}$ is normalized (or regularized) to its p-norm. These constraints also facilitate the interpretation of the weights. In the following we let $\mathbf{x} = (\hat{\mathbf{x}}_I + \hat{\mathbf{x}}_E)$, where $\mathbf{x}(f) \in [-1, 1]$.

There are various ways to choose p, yielding different interpretations. If one uses $\|\mathbf{w_C}\|_{p=1}$, a.k.a. the L_1 norm, the solution picks as the neighborhood focus the *single* attribute with the largest $\mathbf{x}$ entry. That is, $\mathbf{w_C}(f) = 1$ where $\max(\mathbf{x}) = \mathbf{x}(f)$ and 0 otherwise. One can interpret this as the most important attribute that characterizes the neighborhood. Note that $\mathbf{x}$ may contain only negative entries, in which case the largest negative entry is selected. This implies that there exists no attribute that can make the `normality` positive, and hence such a neighborhood is considered anomalous. Note that when $p = 1$, $\hat{N} \in [-1, 1]$.

If there are *multiple* attributes with positive $\mathbf{x}$ entries, we can also select all of them as the neighborhood focus. The weights of these attributes, however, should be proportional to the magnitude of their $\mathbf{x}$ values. This is exactly what $\|\mathbf{w_C}\|_{p=2}$, or the L_2 norm yields. In particular, we can show that $\mathbf{w_C}(f) = \frac{\mathbf{x}(f)}{\sqrt{\sum_{\mathbf{x}(i)>0} \mathbf{x}(i)^2}}$, for $\mathbf{x}(f) > 0$ and 0 otherwise, where $\mathbf{w_C}$ is unit-normalized. Then, the normality score of the neighborhood becomes

$$\begin{aligned} N &= \mathbf{w_C}^T \cdot \mathbf{x} \\ &= \sum_{\mathbf{x}(f)>0} \frac{\mathbf{x}(f)}{\sqrt{\sum_{\mathbf{x}(i)>0} \mathbf{x}(i)^2}} \mathbf{x}(f) = \sqrt{\sum_{\mathbf{x}(i)>0} \mathbf{x}(i)^2} = \|\mathbf{x}_+\|_2 \end{aligned}$$

i.e., the L_2-norm of $\mathbf{x}$ induced on the positive entries. As such, when there are multiple attributes that can make the `normality` positive, L_2 formulation produces an objective value that is higher than that of the L_1 formulation. This agrees with intuition; the larger the number of attributes with positive $\mathbf{x}$ entries, the more attribute-coherence the neighborhood exhibits, and the higher the `normality` gets incrementally. On the other hand, if there are no positive entries in $\mathbf{x}$, the L_2 optimization selects the single attribute with the largest negative entry, and we consider the neighborhood as anomalous. In all, $\hat{N} \in [-1, \|\mathbf{x}_+\|_2]$ when $p = 2$.

While L_1 and L_2 are the two most commonly used norms, one can also enforce $\mathbf{w_C}(f) \leq \frac{1}{k}$, for each f, to obtain the largest k entries of $\mathbf{x}$ that can be interpreted as the top-k most relevant attributes for the neighborhood (note that those may involve both positive and negative entries). In principle, $\mathbf{x}$ provides a systematic and intuitive way to rank the attributes by their relevance to a neighborhood.

Computational complexity: Notice that the solution to the optimization is quite straightforward where the complexity mainly revolves around computing the $\mathbf{x}$ vector. Specifically, the complexity is $O(|C|^2 d + |\mathcal{E}_B| d)$ for computing $\mathbf{x}$ and $O(d)$ for finding the maximum entry (for L_1 regularization) or positive entries (for L_2 regularization), where $\mathcal{E}_B$ is the number of cross-edges which is upper-bounded by $|C||B|$. Therefore, the complexity is quadratic w.r.t. the neighborhood size $|C| \ll n$, and *linear* in the number of attributes d, while it is independent of the size of the entire graph.

4.5 Experiments

Through experiments, we (1) evaluate AMEN's performance [1] in anomaly detection, (2) perform case studies that analyze the type of anomalies we find, and (3) utilize `normality` as an exploratory tool to study the correlation between structure and attributes across different graphs.

Datasets. A detailed description of real-world graphs used in this work is given in Table 5.2.[2] Facebook, Twitter, and Google+ each consists of a collection of ground-truth social circles. For these graphs, we consider these circles as their entity neighborhoods. For the other graphs, we consider the egonets (subgraphs induced on each node and its neighbors) as their entity neighborhoods.

Baselines. We compare AMEN's performance in anomaly detection against the following existing measures and methods. Notation used in baselines: $\mathcal{E}(C) = \{(i,j) \in \mathcal{E} : i \in C, j \in C\}$ (edges induced by C); $cut(C) = \sum_{i \in C, b \in B, (i,b) \in \mathcal{E}} 1$ (cut size induced by C); $vol(C) = \sum_{i \in C} k_i$ (sum of degrees in C).

- **Average degree** [28], $\frac{2|\mathcal{E}(C)|}{|C|}$ (internal consistency only, non-attributed).
- **Cut ratio** [45], $\frac{cut(C)}{|C|(n-|C|)}$, is the fraction of boundary edges over all possible boundary edges (external separability only, non-attributed).
- **Conductance** [8], $\frac{cut(C)}{\min(vol(C), vol(G \setminus C))}$, normalizes the cut by the total volume of C (internal+external quality, non-attributed).

[1] AMEN available at `http://www.perozzi.net/projects/amen`

[2] Datasets available from `http://snap.stanford.edu/data/` and `https://code.google.com/p/scpm/`.

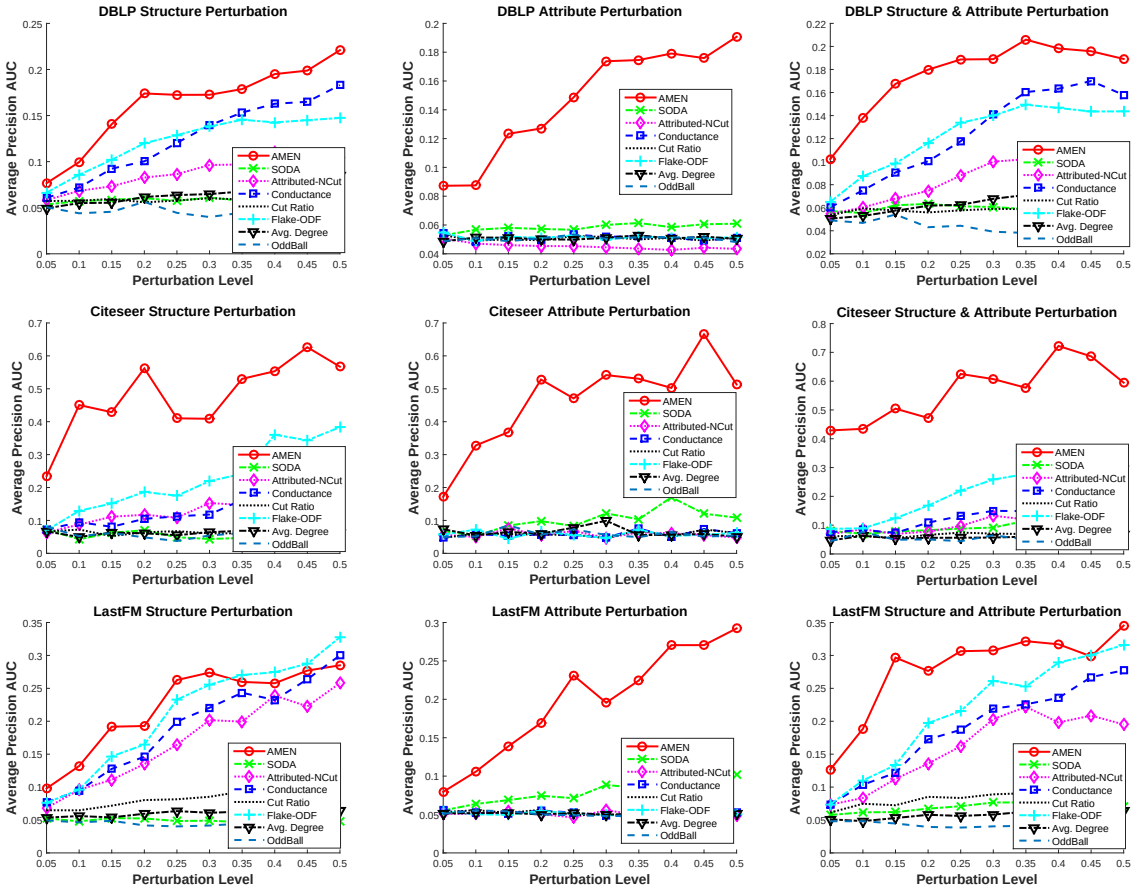


Figure 4.3: Anomaly detection results (mean precision vs. perturbation intensity) for structure only (left), attribute only (center), and structure & attribute (right) anomalies on DBLP (top), Citeseer (middle), and LastFM (bottom). AMEN is superior, especially when attribute perturbations are involved.

- **Flake-ODF** [44], $\frac{\left|\left\{i \in C : |\{(i,j) \in \mathcal{E} : j \in C\}| < k_i/2\right\}\right|}{|C|}$, is the fraction of nodes inside a neighborhood that have less than half of their edges pointing inside (internal+external quality, non-attributed).
- **OddBall** [2] uses a linear model to find neighborhoods that deviate in node density (internal consistency only, non-attributed).
- **SODA** [58] finds a max-margin hyperplane that separates connected and disconnected nodes using both structure and attributes. It ranks neighborhoods by the negative margin of this hyperplane (internal+external quality, attributed).
- **Attribute-Weighted Normalized Cut** is based on a cluster quality measure proposed by [56] for attributed graphs. They identify a subspace of attributes for a cluster, which minimizes its weighted normalized cut, where edges are weighted by the similarity of end-nodes on the selected subspace. Subspace selection is *quadratic* in the number of all attributes. Our real-world datasets DBLP, Citeseer, and LastFM have more than 23 thousand, 206 thousand,

and 3.9 million attributes, respectively, for which [56] is intractable. As such, we consider a simplified version; by using a uniform weight vector over the full attribute space to compute normalized cut (internal+external quality, attributed).

4.5.1 Anomaly Detection

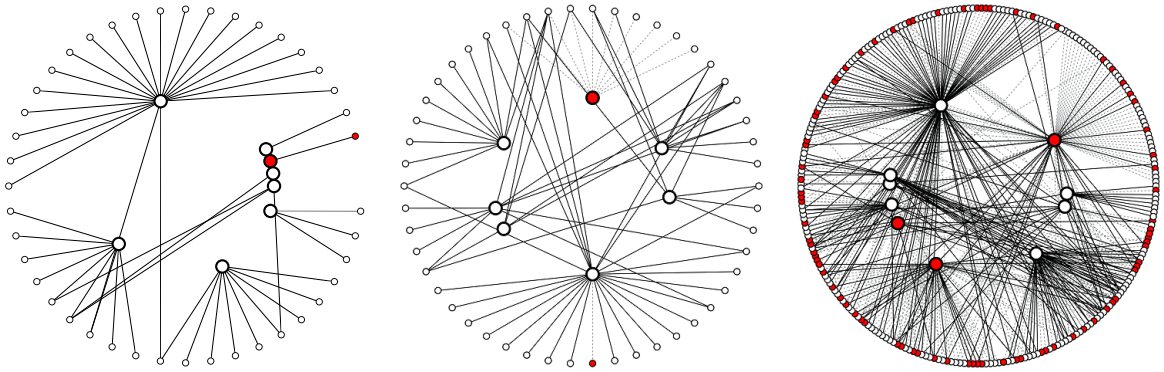
Our evaluation of AMEN's anomaly detection performance is two-fold. First, we perform quantitative evaluation; we inject anomalies into DBLP, Citeseer, and LastFM and compare detection performance of different approaches. Second, we perform a qualitative case study of ground-truth neighborhoods with low `normality` score from Facebook, Twitter, and Google+.

Quantitative Evaluation. To create ground truth anomalies, we use the egonets from DBLP, Citeseer, and LastFM which we perturb to obtain anomalous neighborhoods. Perturbations involve disruptions in (1) structure, (2) attributes, and (3) both. We start by choosing "good" neighborhoods; specifically, small egonets (of size 30-100) that we expect to have low conductance cuts [54]. From these egonets, we choose 5% of them as anomalous, i.e., to be perturbed. To perturb structure, we rewire inside edges to random outside nodes with rewiring probability p. To perturb attributes, we replace the attributes of inside nodes with the corresponding attributes of randomly picked outside nodes with probability q (note that this "inheritance" only affects the egonet, and keeps the outside nodes unchanged). For structure and attribute perturbation, we vary p or q from 0.05 to 0.50. To perturb both, we vary them simultaneously. The larger the perturbation intensities p and q, the more disrupted the egonet. We expect this process to create anomalous (ground truth) egonets, that are structurally poor, for which it is hard to find shared focus attributes.

We evaluate our measure in its ability to rank the disrupted ground truth anomalies high. Specifically, we rank the neighborhoods by their `normality` and report the AUC (i.e., average precision) of the precision-recall plots for each p and/or q perturbation intensity in Figure 4.3. We find that AMEN consistently outperforms all other measures and methods, especially at low perturbation intensities where the task is harder—hence the gradual increase in performance by intensity. When structure perturbation is involved (See figures on (left) and (right)), conductance and Flake-ODF also appear to do well. When attributes are perturbed, on the other hand, AMEN is superior, where SODA, the second best, is significantly worse than AMEN. Across perturbation strategies and datasets, on average AMEN outperforms Flake-ODF by 16%, conductance by 18%, AW-NCut by 20%, SODA by 23%, average degree by 24%, cut ratio by 24%, and OddBall by 25%.

Case Studies. We begin our case studies by returning to Figure 4.1, which illustrated several examples of neighborhoods sorted by their `normality` (from high to low) from various graphs. We see that the highest quality neighborhood shown (from DBLP) has good internal structure, and attributes which allow the complete exoneration of edges at its boundary. As we progress through neighborhoods in order of quality, we see that internal structure weakens, and boundaries become larger. The lowest quality neighborhood shown (from Citeseer) has almost no internal structure, and no attributes which exonerate its large boundary.

Next, we examine some of the most anomalous ground truth circles from Facebook, Twitter, and Google+. Figure 4.4 shows that the circles with the lowest `normality` have both weak internal connectivity and poor boundary separation. None of the available attributes is able to meaningfully



Facebook
$L_1 = L_2$= -0.995

Twitter
$L_1 = L_2$= -0.993

Google+
$L_1 = L_2$= -0.873

Figure 4.4: Low `normality` ground truth neighborhoods from Facebook, Twitter, and Google+.

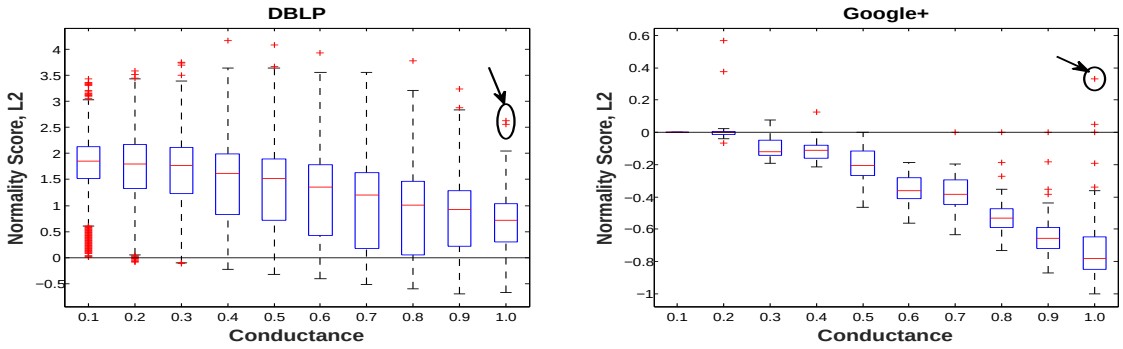
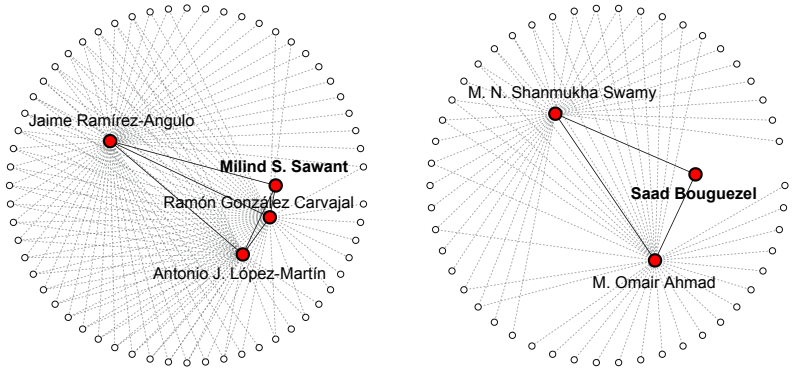


Figure 4.5: Box plots depicting L_2-`normality` score vs. conductance for (a) DBLP and (b) Google+.

improve their score. We find these ground-truth neighborhoods to be particularly interesting, as they are real circles manually defined by social network users (hence ground truth) which do not, however, exhibit characteristics of what 'good' communities in graphs look like.

Finally, we examine the difference between `normality` and conductance, a popular structural measure, in Figure 4.5. As expected, `normality` gets lower for neighborhoods with increasingly high (bad) conductance (results are for DBLP and Google+, others are similar). However, notice the many neighborhoods in DBLP with high `normality` scores even if they have very high (in range $(0.9, 1.0]$) conductance (several others in other graphs, and even one in Google+). These are exactly the type of neighborhoods that are considered low quality by solely structural measures such as conductance, but achieve high score when attributes and surprise are carefully accounted for under the notion of "exoneration".

For example, consider the two neighborhoods with poor conductance but high `normality` from DBLP as shown in Figure 4.6. The left drawing depicts the egonet of 'Milind S. Sawant' (ego)—a co-authorship circle of four researchers from electrical engineering—as well as their boundary



L_2-`normality`: 2.55
Focus: {telescopic, multidecade, cascode}

L_2-`normality`: 2.63
{reciprocal, split, reverse}

Figure 4.6: Example high (poor) conductance but high `normality` neighborhoods from DBLP.

(other collaborators). We notice that the other nodes in the egonet, i.e. the neighbors of the ego, have considerably larger set of collaborators. This creates numerous cross-edges, yielding poor conductance. However, this circle is well-defined and "focused"; these four authors exclusively work together on 'telescopic op-amps'; notice the listed focus attributes (words). Similarly, the ego 'Saad Bouguezel' of the right neighborhood is much less connected than his collaborators, creating excessive cross-edges. Those edges are exonerated by `normality`, due to their discriminating focus on 'reciprocal transforms' and 'split-radix FFTs'.

4.5.2 Graph Analysis with Normality

`Normality` also serves as a powerful tool for analyzing the correlation of structure and attributes in a graph. In this section, we compare the distribution of neighborhood scores across all graphs. Using `normality` as a lens, we see that:

- Different graphs display distinct distributional fingerprints, indicating very diverse correlation patterns between structure and attributes.
- In all graphs, the biggest gains in `normality` come only from a few attributes—suggesting that a neighborhood *focus* is often sparse.

First, to examine the influence of attributes upon each neighborhood, we turn to $\mathbf{x} = (\hat{\mathbf{x}}_I + \hat{\mathbf{x}}_E)$ (Eq. (4.9)). For each neighborhood, we count the number of positive entries of $\mathbf{x}$, $\#(\mathbf{x}(i) > 0)$, and present their distribution in Figure 4.7(a). As expected, only a small number of attributes are relevant for most communities, so a sparse $\mathbf{w}$ is obtained by L_2. We see that both LastFM and DBLP have many positive attributes for each neighborhood, followed by Facebook, Citeseer, and Twitter. The worst neighborhoods are found in Google+, where 99% of the circles do not have *any* attribute that characterizes them (i.e., non-positive $\mathbf{x}$). Figure 4.7(b) shows the fraction of nodes in each neighborhood which exhibit the attribute that would be chosen by L_1 maximization. Again, DBLP and LastFM have the most agreement inside each neighborhood, followed closely by

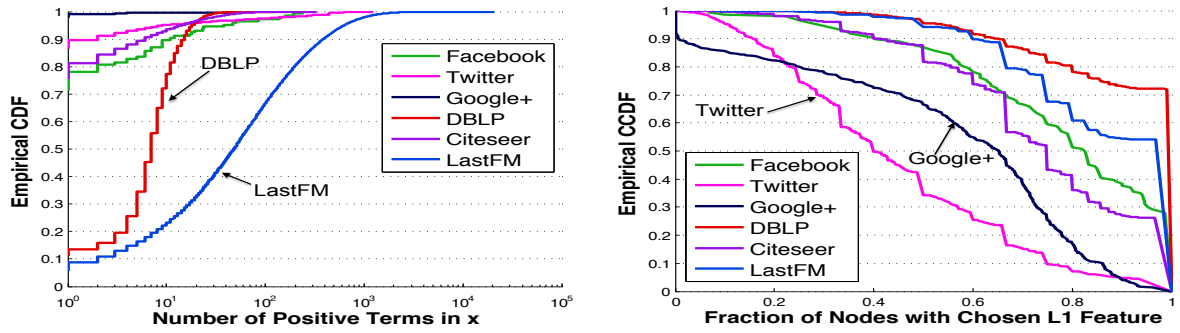


Figure 4.7: Distribution of neighborhoods w.r.t. (a) count of positive terms in x (cdf), (b) fraction of nodes exhibiting the L_1-selected attribute (ccdf).

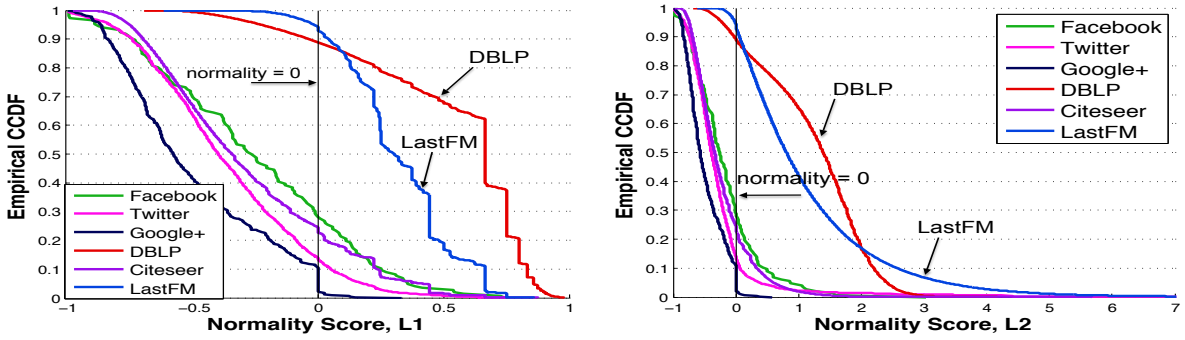


Figure 4.8: Distribution of neighborhoods w.r.t. `normality`**(ccdf); w constraint by (a)** L_1**, (b)** L_2**.**

Facebook and Citeseer, while Twitter and Google+ neighborhoods are the most noisy in exhibiting the highest ranked attribute.

Figure 4.8 shows the distribution of normality scores across neighborhoods for both L_1 (a) and L_2 (b) maximization. We see that 89% of DBLP and 95% of LastFM neighborhoods have positive `normality`, while with the exception of a few very good neighborhoods, the rest are mostly negative. L_2 optimization does not dramatically change where the bulk of the distribution is. Majority of DBLP neighborhoods gain additional score, and 10% of LastFM neighborhoods gain a lot of score. In others we see a small improvement, where Google+ presents neighborhoods with lowest `normality` (its best circle score by L_2 is ≈ 0.6).

The contribution from each positive attribute to the neighborhood score for DBLP and LastFM is shown in Figure 4.9. We see that relevance drops fast, and essentially zeroes out after around 20 attributes. An interesting difference occurs between the two graphs, which have both many good neighborhoods, but achieve them in different ways. In DBLP, the first few positive attributes are all about equally good, but then they degrade quickly. On the other hand, LastFM attributes are usually not as good, but many of them can add up to achieve a neighborhood with a high score (and hence the long tail in Figure 4.8(b)).

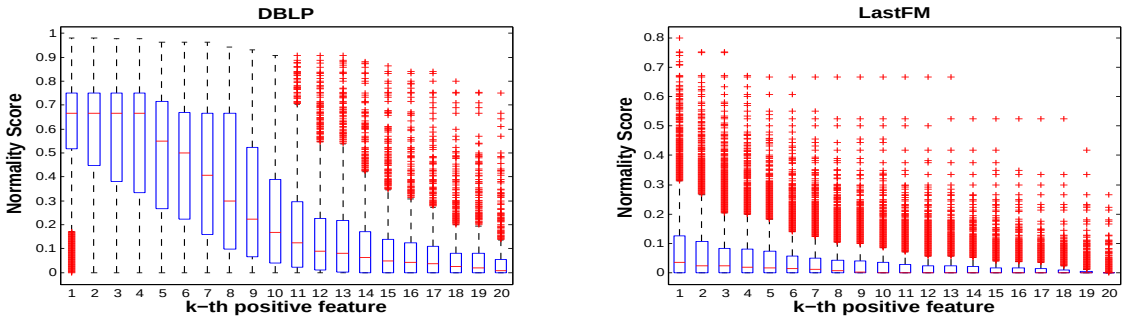


Figure 4.9: `Normality` **of neighborhoods based on k-th most positive attribute with highest x entry.**

4.6 Related Work

We organize related work into two groups: (1) analysis of community structure and community quality, and (2) anomaly detection in non-attributed and attributed graphs.

4.6.1 Analysis of community structure and quality

Leskovec *et al.* studied the statistical properties of communities in real social and information networks and analyzed how they split into communities and how typical community qualities (according to the conductance measure) change over a range of size scales [76]. Their findings suggest the absence of large well-defined communities, which has been corroborated in a later study by Gleich *et al.* [54]. Arnaboldi *et al.* analyzed the structure of ego networks and found that social relationships in online and offline social networks are organized similarly [10]. Akoglu *et al.* found that egonet characteristics display power-law-like patterns in real networks [2]. Other works studied the structure and dynamics of real-world graphs at large, without specific focus on communities [75, 89]. The focus of all these works is on networks with no attributes.

Several other works quantify the quality of communities in graphs. Yang and Leskovec investigated a long list of such measures and compared their performance based on ground-truth communities [151]. Newman studied the mixing properties in attributed graphs to quantify the correlations between the attributes of adjacent nodes; high correlation is referred as assortative mixing, which tends to break the network into communities [105]. Similarly, Silva *et al.* studied the correlation between attribute sets and dense subgraphs, called the structural correlation patterns in attributed graphs [128]. Finally, work in subspace clustering [100] has also been extended as a quality measure to find graph-cuts which correlate with attributes [56]. These methods have utilized their measures to define a global objective for the graph clustering/partitioning problem, and are not directly applicable to anomaly detection of neighborhoods.

4.6.2 Graph anomaly detection

In their seminal work, Noble and Cook [110] used frequent subgraph mining and information theoretic principles to identify anomalous subgraphs in graphs with a *single* attribute. Similarly, Ni *et al.* developed algorithms to find sets of connected nodes in a graph for which a *single* attribute value is significantly higher than in the neighborhood of the set [78, 79]. Akoglu *et al.* proposed OddBall to find *structural* anomalies, where they found and used patterns in egonets to flag the anomalies in non-attributed graphs [2]. Gao *et al.* formulated a new problem to identify community outliers which deviate in the full attribute space from others that belong to the same community [51]. Perozzi *et al.*, on the other hand, focused on community outliers that deviate on a pre-defined subset of attributes that is inferred from user preference [114]. Both of those works find node outliers within communities rather than anomalous communities.

Most recently, Gunnemann *et al.* generalized the normalized cut measure for clustering attributed graphs [56]. Each cluster k is associated with a respective subspace s_k of relevant attributes, similar to our focus attributes. As such, a score called normalized subspace cut (NSC) can be computed for each cluster. While NSC score can be used as a quality measure for a given neighborhood, there are three main shortcomings to computing it: (1) The formulated objective is *not convex* in subspace s_k. (2) Proposed method is a *heuristic*, as the problem is not tractable for large graphs. Moreover, to make the search space more tractable, the authors limit solution vectors to binary: irrelevant/relevant, and by doing so compromise from inferring attribute weights. Finally, (3) despite the way the search space is limited to binary weight vectors, their proposed heuristic is *quadratic* in the number of all attributes $|S|$ (and not the subspace size $|s_k|$). This disables us to use their method for graphs with millions, or even tens of thousands of attributes (e.g., our LastFM and DBLP graphs in Table 5.2).

Most similar to ours is the work by Gupta *et al.* on outlier subgraph discovery [58]. Their formulation involves inferring an attribute subspace in which the margin between minimum dissimilarity among disconnected nodes and maximum dissimilarity among connected nodes is maximized. For normal subgraphs, this margin is expected to be large—as the minimum dissimilarity among disconnected nodes is still large and the maximum dissimilarity among connected nodes is small. Our quality formulation is considerably different and yields a much faster optimization. Moreover, none of the existing works including [58] has a notion of edge "exoneration" as we introduce in our work.

4.7 Conclusion

In this work, we considered the problem of discovering anomalous neighborhoods in attributed graphs. We proposed `normality`, a new quality measure that evaluates neighborhoods both Internally and Externally. Intuitively, a high-quality neighborhood has members with ($I1$) many *surprising* edges connecting them that ($I2$) share similar values in a particular attribute subspace, called the neighborhood *focus*. Moreover, it has either ($E1$) a few edges at the boundary, or ($E2$) many cross-edges which can be *exonerated* as unsurprising under the null graph model and/or dissimilar with respect to the focus attributes.

We utilized `normality` within an objective formulation for anomaly mining and provided solutions to maximize it, which automatically identify the latent focus attributes as well as their respective weights. Our formulation yields a scalable optimization that is only quadratic w.r.t. (often small) neighborhood size and linear in attribute size. Overall, `normality` is unique in its (i) exoneration of cross-edges, and (ii) automatic inference of focus attributes under a scalable convex optimization.

Experiments on real-world graphs demonstrate the utility of our measure in ranking neighborhoods by quality, where we outperform well-established measures and methods. `Normality` also provides a powerful tool for studying the correlation between structure and attributes for graphs. Our work enables a number of future directions, including utilizing `normality` for community detection at large and for user profiling, i.e., recovering missing node attributes.

Chapter 5

Focused Clustering and Outlier Detection

5.1 Introduction

Many real-world graphs have attributes associated with the nodes, in addition to their connectivity information. For example, social networks contain both the friendship relations as well as user attributes such as interests and demographics. A protein-protein interaction network may not only have the interaction relations but the gene expressions associated with the proteins. Both types of information can be described by a graph in which nodes represent the objects, edges represent the relations between them, and feature vectors associated with the nodes represent the attributes. Such graph data is often referred to as an *attributed graph*.

For attributed graphs, we see major challenges that remain unsolved by traditional graph mining techniques [11, 38, 68, 109, 146], which consider plain graphs (without attributes). Recent methods have been proposed for attributed graphs, however, they either use all the given attributes [3, 51, 82, 154] or they perform an unsupervised feature selection [57, 66, 98, 130]. In contrast to all of these graph mining paradigms (cf. Table 5.1), we consider a user-oriented setting where the users can control the relevance of attributes and as a consequence, the graph mining results.

In particular, we consider cluster and outlier detection based on user preference. This *focused clustering* is of particular interest in attributed graphs, where users might not be concerned with all but a few available attributes. As different attributes induce different clusterings of the graph, the user should be able to steer the clustering accordingly. As such, the user controls the clustering by providing a set of exemplar nodes (perceived similar by the user) from which we infer *attribute weights* of relevance that capture the user-perceived similarity. The essence of user preference is captured by those attributes with large weights. We call these the *focus attributes*, which form the basis of our approach for discovering focused clusters and outliers.

To elaborate on this new terminology, we give a toy example in Figure 5.1. The graph represents the friendship relations and the node attributes denote `degree, location, mother`

This work originally appeared as "Bryan Perozzi, Leman Akoglu, Patricia Iglesias Sánchez, and Emmanuel Müller. Focused clustering and outlier detection in large attributed graphs. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 1346–1355, New York, NY, USA, 2014. ACM."

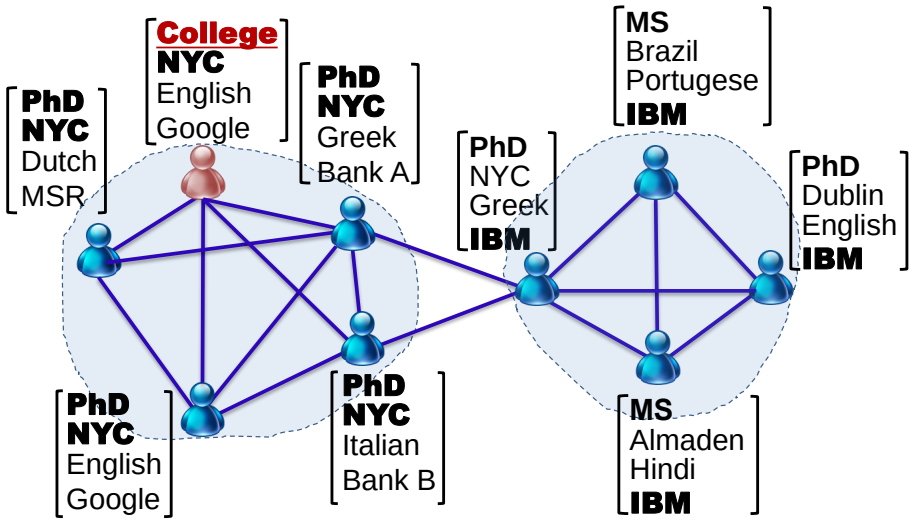


Figure 5.1: Example graph with two focused clusters and one focused outlier.

Table 5.1: Comparison of related work.

Property	Graph clustering	On attributed graphs	Attribute subspace	User-preferred clusters	Overlapping clusters	Outlier detection	Scalability
METIS [68], Spectral [109], Co-clustering [38]	✓						✓
Autopart, Cross-associations [26]	✓					✓	✓
PageRank-Nibble [8], [145], BigClam [152]	✓				✓		✓
Spectral Relational [82], SA-Cluster [154]	✓	✓					
CoPaM [98], Gamer [57]	✓	✓	✓				✓
PICS [3]	✓	✓					✓
CODA [51]	✓	✓				✓	
GOutRank [101], ConSub [66]		✓	✓			✓	✓
FocusCO [this paper]	✓	✓	✓	✓	✓	✓	✓

`tongue`, and `work`. There are two focused clusters: On the left, people know each other due to their `degrees` and `locations`. On the right, a similarity in `work` induces the second cluster. As such, different user interest in subsets of the attributes may induce different clusters. In case a user is interested in `degree` and `location`, focused clustering should only find the left cluster and not the right one. Analogously, the example outlier is deviating with a college degree among all others having PhDs, where `degree` is a focus attribute.

While our example is on a toy graph, our problem setting has several practical applications in the real-world. For instance, a marketing manager interested in selling cosmetics could aim to find communities in a large social network with its members being of a certain age, gender, and income-level. S/he could then offer deals to a few members from each community, and expect the news to propagate by the word-of-mouth. A scientist could aim to identify clusters of sky-objects that are all in close-distance to one another (assuming a graph is constructed among sky-objects by

distance in space) and share certain characteristics of interest (e.g., helium level, temperature, light, etc.).

Such user-preferred clusters are likely a handful in the graph, and thus an algorithm should be able to (i) effectively identify user preference, (ii) efficiently "chop out" relevant clusters locally without necessarily partitioning the whole graph, and additionally (iii) spot outliers if any. In this paper, we offer the following contributions:

- **Focused Graph Mining**: We propose a new user-centric problem setting that exploits the user interest for focused mining in attributed graphs.
- **Steered Search by User Preference**: We infer user preference and steer the graph clustering and outlier detection accordingly.
- **Scaling to Large Graphs:** Our proposed method has near-linear time complexity, and with appropriate initialization can run in time sub-linear w.r.t. the size of the input graph.

In our evaluation we demonstrate the effectiveness and scalability of our method on synthetic and real-world graphs, compared to existing graph clustering and outlier detection methods. Our experiments show that existing approaches are not suitable for the new focused graph mining setting.

5.2 Related Work

We show the highlights of related work in Table 5.1. The two key differences of our work are summarized as follows: (1) we introduce a new user-oriented problem setting for attributed graphs, in which we aim to find *focused* clusters and outliers based on *user preference*, and (2) we propose an algorithm that *simultaneously* extracts relevant clusters and outliers from large graphs. In the following, we discuss related work in three areas; traditional plain graph mining, attributed graph mining, and semi-supervised data mining.

Graph mining on plain graphs Graph partitioning has been well studied in the literature. Widely used methods include METIS [68] and spectral clustering [39, 109], which aim to find a k-way partitioning of the graph. Different from partitioning, community detection methods [44] cluster the graph into variable size communities. Autopart, cross-associations [26], and information-theoretic co-clustering [38] are parameter-free examples to graph clustering methods. Several methods [8, 145, 152] also allow clusters to overlap as observed in real-world social and communication networks. Works that aim to spot structural outliers in plain graphs include [2, 138]. However, all of these methods are limited to plain graphs (without attributes).

Graph mining on attributed graphs Compared to the wide range of work on plain graph mining, there has been much less work on attributed graphs. The representative methods [3, 51, 60, 82, 137, 154] aim to partition the given graph into structurally dense and attribute-wise homogeneous clusters, detect deviations from frequent subgraphs [110], or search for community outliers in attributed graphs [51]. These methods, however, enforce attribute homogeneity in all attributes. Recently some methods loosen this constraint by unsupervised feature selection [130], subspace clustering [57, 98] and subspace outlier detection [66, 101] and extract cohesive subgraphs with homogeneity

in a subset of attributes. However, all of these methods either do not perform a selection of attributes or do not allow for user preference to steer the algorithm.

Semi-supervised methods A broad variety of methods for semi-supervised clustering consider user-given constraints like 'must-link' and 'cannot-link' referred to as constraint-based clustering [13]. There also exist methods for semi-supervised outlier mining [50]. However, these methods are based on vector data and further, not applicable to graphs with attributes.

Methods on seeded community mining [8, 32, 54, 145] find communities around (user-given) seed nodes. However, those methods find structural communities on plain graphs and neither apply to attributed graphs, nor enable user preference on attributes. Moreover, they do not provide outlier detection. In contrast, we use user-given exemplar nodes to automatically infer user preference on attributes. To the best of our knowledge this problem setting is new and we propose the first focused graph mining approach for simultaneous clustering and outlier detection in attributed graphs.

5.3 Method FOCUSCO

In this section we first introduce the notation and pose the focused clustering and outlier detection problem formally. Next, we discuss the main components of our approach and walk through the details of our algorithm. Lastly, we analyze the computational complexity of FOCUSCO.

5.3.1 Problem Formulation

In this paper we introduce the novel problem of *focused clustering and outlier detection* in attributed graphs, defined as follows: Given a large attributed graph $G(V, E, F)$ with $|V| = n$ nodes and $|E| = m$ edges, where each node is associated with $|F| = d$ attributes (features), extract from G only the (type of) clusters pertaining to a user u's interest (rather than partitioning the whole graph). To do so, the user provides a small set C_{ex} of exemplar nodes that s/he considers to be similar to the type of nodes the clusters of his/her interest should contain. Assuming that the nodes in a cluster "unite" or "knit up" around a few defining attributes, we then aim to infer the implicit weights $\boldsymbol{\beta_u}$ (i.e., relevance) of attributes that "define" the nodes in C_{ex}, i.e., the weights of attributes that make them as similar as possible. Thus, $\boldsymbol{\beta_u}$ is expected to be a sparse vector with large weights for only a few attributes (e.g., `degree` and `location` in Figure 5.1), which we call the *focus attributes*.

Having inferred the attribute weights $\boldsymbol{\beta_u}$ from user u, our first goal is to extract focused clusters $\mathcal{C}$ from G that are (1) structurally dense and well separated from the rest of the graph, as well as (2) consistent on the focus attributes with large weights. The focused clusters can be overlapping, sharing several of their nodes, as observed in real-world social and communication networks. Moreover, the set $\mathcal{C}$ is a subset of all the clusters in G since different sets of clusters are expected to unite around different attributes and we aim to extract only those that are specifically similar to the type of clusters user u is interested in. Besides focused clustering, our second goal is to also perform *outlier detection*. Outliers $\mathcal{O}$ are those nodes that structurally belong to a focused cluster (i.e., have many cluster neighbors), but deviate from its members in some focus attributes. In summary, the focused clustering and outlier detection problem in attributed graphs is given as follows:

Given a large graph $G(V, E, F)$ with node attributes, and a set of exemplar nodes C_{ex} of user u's interest;

Infer attribute *weights* $\boldsymbol{\beta_u}$ of relevance/importance,

Extract *focused* clusters $\mathcal{C}$ that are (1) dense in graph structure, and (2) coherent in heavy focus attributes,

Detect focused outliers $\mathcal{O}$, i.e. nodes that deviate from their cluster members in some focus attributes.

5.3.2 Approach and Algorithm Details

Next we present the details of the three main components of FOCUSCO: (1) inferring attribute weights, (2) extracting focused clusters, and (3) outlier detection.

Inferring Attribute Relevance

Our focused clustering setting is a user-oriented one, where each user is interested in extracting certain kind of clusters from a given graph. The user steers the clustering by providing a small set of exemplar nodes that are similar to one another as well as similar to the type of nodes the clusters of his/her interest should contain. Our first goal then is to identify the relevance weights of node attributes that make the exemplar nodes similar to each other. This kind of weighted similarity is often captured by the (inverse) Mahalanobis distance: the distance between two nodes with feature vectors $\boldsymbol{f_i}$ and $\boldsymbol{f_j}$ is $(\boldsymbol{f_i} - \boldsymbol{f_j})^T \boldsymbol{A} (\boldsymbol{f_i} - \boldsymbol{f_j})$. Setting $\boldsymbol{A}$ as the identity matrix yields Euclidean distance, otherwise the features/dimensions are weighted accordingly.

Given the exemplar nodes, how can we learn an $\boldsymbol{A}$ such that they end up having small distance to each other? This is known as the distance metric learning problem [143]. We adopt the optimization objective by [148]:

$$\min_{\boldsymbol{A}} \sum_{(i,j)\in P_S} (\boldsymbol{f_i} - \boldsymbol{f_j})^T \boldsymbol{A} (\boldsymbol{f_i} - \boldsymbol{f_j}) - \gamma \log \left(\sum_{(i,j)\in P_D} \sqrt{(\boldsymbol{f_i} - \boldsymbol{f_j})^T \boldsymbol{A} (\boldsymbol{f_i} - \boldsymbol{f_j})} \right) \tag{5.1}$$

which is convex and enables efficient, local-minima-free algorithms to solve it, especially for a diagonal solution.

We give the details of inferring attribute weights in Procedure P1. P_S and P_D are two sets of similar and dissimilar pairs of nodes, respectively (P1 Line 1). In our setting, all pairs of exemplar nodes constitute P_S (P1 Line 2). We create P_D by randomly drawing pairs of nodes that do not belong to the exemplar set (P1 Lines 3-7).

We remark that in creating P_D, we may also obtain samples similar to those in P_S since these draws are random. To alleviate the affect of such draws, we keep the size of P_D sufficiently large. This assumes that the number of dissimilar pairs is substantially larger than the number of similar pairs in the original distribution. This a suitable assumption, given that the number of "focused" clusters for any particular user-preference is likely small. Thus, we make the size of P_D be $|F|$ times larger than that of P_S (P1 Line 7). This also ensures that the data size exceeds dimension size, and that the learning task is feasible.

Moreover, in inferring attribute weights we learn a diagonal $\boldsymbol{A}$ matrix (P1 Line 9). The reason for this choice is two-fold. First, individual weights for attributes provide ease of interpretation. Second, learning a diagonal $\boldsymbol{A}$ is computationally much more tractable (especially in high dimensions) than learning a full one, since the latter requires solving a program with a semi-definite constraint. Of course if desired, one can instead learn a full matrix (in low dimensions).

Procedure P3 INFERATTRIBUTEWEIGHTS

Input: exemplar set of nodes C_{ex}
Output: attribute weights vector $\boldsymbol{\beta}$

// generate similar and dissimilar node pairs
1: Similar pairs $P_S = \emptyset$, Dissimilar pairs $P_D = \emptyset$
2: **for** $u \in C_{ex}, v \in C_{ex}$ **do** $P_S = P_S \cup (u, v)$ **end for**
3: **repeat**
4: Random sample u from set $V \backslash C_{ex}$
5: Random sample v from set $V \backslash C_{ex}$
6: $P_D = P_D \cup (u, v)$
7: **until** $d|P_S|$ dissimilar pairs are generated, $d = |F|$
8: Oversample from P_S such that $P_S = P_D$
9: Solve objective function in Equ. (5.1) for diagonal $\boldsymbol{A}$
10: **return** $\boldsymbol{\beta} = diag(\boldsymbol{A})$

Focused Cluster Extraction

Having determined attribute weights $\boldsymbol{\beta}$, we extract the focused clusters of interest. The main idea in "chopping out" focused clusters from G is to first identify good candidate nodes that potentially belong to such clusters, and then to expand around those candidate nodes to find the clusters. Details are given in Algorithm A1, which we describe below.

The process of finding good candidate sets to expand is detailed in Procedure P2. Intuitively, nodes in focused clusters have high weighted similarity to their neighbors. Therefore, we first re-weigh the edges E by the weighted similarity of their end nodes (P2 Lines 2-4), induce G on the edges with notably large weights[1] (P2 Line 5), and consider the nodes in the resulting connected components as our candidate nodes (P2 Lines 6-7). We call each such component a *core* set.

Next, we expand around each core by carefully choosing new nodes to include in the cluster and continue expanding until there exist no more nodes that increase the quality of the cluster. There exist several measures of cluster quality including modularity and cut size [109]. In this work, we use conductance [8] as it accounts for both the cut size as well as the total volume/density retained within the cluster. The weighted conductance $\phi^{(w)}(C, G)$ of a set of nodes $C \subset V$ in graph $G(V, E, F)$ is defined as

[1]To identify such edges, we use hypothesis testing. We first find the top few most weighted edges, and bootstrap a Normal distribution. We then progressively subject the remaining edges to a membership-test and consider only those that pass the test. Every time an edge passes the test, model parameters are updated.

Algorithm A1 FOCUSCO: FOCUSED CLUSTERS&OUTLIERS

Input: attributed graph $G(V, E, F)$, exemplar nodes C_{ex}
Output: focused clusters $\mathcal{C}$ and outliers $\mathcal{O}$

1: $Cores \leftarrow$ FINDCORESETS$(G(V, E, F), C_{ex})$
2: $\mathcal{C} = \emptyset$, $\mathcal{O} = \emptyset$
3: **for each** core $i \in Cores$ **do**
4: $C \leftarrow Seeds(i).getNodes()$
5: $BSN = \emptyset$ *// holds all Best Structural Nodes to add*
6: $\phi^{(w)}_{curr} \leftarrow \phi^{(w)}(C, G)$
7: **repeat**
8: $\phi^{(w)}_{init} \leftarrow \phi^{(w)}_{curr}$
9: $(C, BSN, \phi^{(w)}_{curr}) \leftarrow$ EXPAND$(G, C, BSN, \phi^{(w)}_{curr})$
10: $(C, \phi^{(w)}_{curr}) \leftarrow$ CONTRACT$(G, C, \phi^{(w)}_{curr})$
11: $BSN \leftarrow BSN \backslash C$
12: **until** $\phi^{(w)}_{init} = \phi^{(w)}_{curr}$
13: $\mathcal{C} \leftarrow \mathcal{C} \cup C, \quad \mathcal{O} \leftarrow \mathcal{O} \cup BSN$
14: **end for**

$$\phi^{(w)}(C, G) = \frac{W_{cut}(C)}{WVol(C)} = \frac{\sum\limits_{(i,j)\in E, i\in C, j\in V\backslash C} w(i,j)}{\sum\limits_{i\in C}\sum\limits_{j,(i,j)\in E} w(i,j)}$$

where $WVol(C)$ is the total weighted degree of nodes in C. The lower the conductance of a cluster, the better its quality is with few cross-cut edges and large within-density.

The expansion operation is presented in Procedure P3. First, we enlist all their non-member neighbors as the candidate set (P3 Line 4). For each candidate node n, we compute the difference $\Delta\phi^{(w)}_n$ in cluster conductance if n was to be added to C (P3 Lines 6-16). If there exist any node with negative Δ (i.e., node improves conductance), we pick the best n with the minimum (i.e., largest absolute drop in conductance) (P3 Lines 17-23). We continue iterating until no candidate node yields negative $\Delta\phi^{(w)}$.

The node additions are based on a best-improving search strategy. While being cautious in which node to add (the best one at every step), our decisions are greedy. Thus, in the following step of our algorithm we adopt a retrospective strategy and check if there exist any nodes in C whose removal would drop conductance, presented in Procedure P4. We repeat the node addition and removal iterations until convergence, that is, when the conductance stops changing (A1 Lines 7-12).

We remark that our algorithm is guaranteed to converge; as the (weighted) conductance of a cluster is lower-bounded by 0 and we improve (i.e., decrease) the weighted conductance in every iteration (P3 Line 8, P4 Line 5).[2]

[2]P4 Line 5 removes a node even if conductance remains the same, however, the number of such steps is also bounded by cluster size.

Procedure P2 FINDCORESETS

Input: attributed graph $G(V, E, F)$, exemplar nodes C_{ex}
Output: seed sets to expand as focused clusters

1: $\boldsymbol{\beta} \leftarrow$ INFERATTRIBUTEWEIGHTS(C_{ex})
// (re)-weigh edges by feature similarity of end-nodes
2: **for each** $(i, j) \in E$ **do**
3: $w(i, j) = 1/(1 + \sqrt{(\boldsymbol{f_i} - \boldsymbol{f_j})^T diag(\boldsymbol{\beta})(\boldsymbol{f_i} - \boldsymbol{f_j})})$
4: **end for**
5: $w' \leftarrow \max_{w'} w' \notin$ distribution$f(\{w|\ w \geq w'\})$
6: Build induced subgraph $g(V', E', F)$ s.t.
$\forall u, v \in V', (u, v) \in E, w(u, v) \geq w'$ iff $(u, v) \in E'$
7: **return** ConnectedComponents$(g(V', E', F))$

We omit the details of the Δ conductance computation for brevity, but remark that it is an efficient operation. Specifically, the operation of a node u to be added to or to be removed from a cluster S has complexity proportional to the degree of u, i.e. $O(d(u))$. In addition, the total volume of S is simply increased/decreased by the weighted degree $w(u)$ of u, when it is added/removed, which takes $O(1)$.

Focused Outlier Detection

Our algorithm also identifies outlier nodes in each focused cluster along with finding the clusters in a unified fashion. Our definition of a focused cluster outlier is quite intuitive: a node that belongs to a focused cluster structurally (having many edges to its members), but that deviates from its members in some focus attributes significantly is an outlier. To quantify this definition, the main idea is to identify the best *structural* nodes BSNs (best in terms of *un*weighted conductance) during the course of expansion (P3 Lines 3, 14, 22) and later check if there exist any BSNs which were not included in the resulting focused cluster (A1 Line 11).

In order to identify the best structural node for a cluster in each iteration, we need to also track its unweighted conductance. An advantage of our proposed approach is that the overhead of computing the unweighted $\Delta\phi$ of a node, in addition to its weighted $\Delta\phi^{(w)}$, is negligible. The reason is that, to compute $\Delta\phi$, we simply count the total number, instead of the total weight, of those same edges that are involved in the computation of $\Delta\phi^{(w)}$. As such, both conductances can be computed efficiently at the same time and the best structural node and the best focused cluster node can be identified simultaneously.

Complexity analysis

Given an attributed graph $G(V, E, F)$ and the exemplar nodes C_{ex}, we first create similar and dissimilar node pairs which we use to infer the attribute weights. As the optimization objective we adopt is convex and as we aim for a diagonal solution, local-optima-free gradient descent techniques will take $O(\frac{d}{\epsilon^2})$ for an ϵ-approximate answer [23].

Procedure 13 EXPAND

Input: attributed graph $G(V, E, F)$, focused cluster C, set BSN, current conductance $\phi_{curr}^{(w)}$
Output: a focused cluster C, its best structural nodes BSN, and its conductance $\phi_{curr}^{(w)}$

repeat
  $bestNode = NULL$,
  $bestStructureNode = NULL$
  $candidateNodes \leftarrow neighbors(C)$
  $\Delta\phi_{best}^{(w)} = 0$, $\Delta\phi_{best} = 0$
  for each node n in $candidateNodes$ **do**
    $\Delta\phi_n^{(w)}, \Delta\phi_n \leftarrow$ GETΔCONDUCTANCE$(G, C, n,$ADD$)$
    if $\Delta\phi_n^{(w)} < \Delta\phi_{best}^{(w)}$ **then**
      $\Delta\phi_{best}^{(w)} = \Delta\phi_n^{(w)}$
      $bestNode \leftarrow n$
    end if
    if $\Delta\phi_n < \Delta\phi_{best}$ **then**
      $\Delta\phi_{best} = \Delta\phi_n$
      $bestStructureNode \leftarrow n$
    end if
  end for
  if $bestNode \neq NULL$ **then**
    $C \leftarrow C \cup bestNode$
    $\phi_{curr}^{(w)} = \phi_{curr}^{(w)} + \Delta\phi_{best}^{(w)}$
  end if
  if $bestStructureNode \neq NULL$ **then**
    $BSN \leftarrow BSN \cup bestStructureNode$
  end if
until $bestNode = NULL$
return $C, BSN, \phi_{curr}^{(w)}$

To determine good core sets to expand clusters around, we re-weigh the graph edges by the weight vector $\boldsymbol{\beta}$ with complexity $O(dm)$. Assuming $\boldsymbol{\beta}$ is sparse with only a few non-zero entries for focus attributes, the multiplicative factor becomes effectively constant yielding a complexity of $O(m)$. Next, we identify the top-k edges with largest weights on which we induce G to find the core sets ($k \ll m$). To do so, we use a min-heap to maintain this top set while making a single pass over the edges. This requires $O(m \log k)$ in the worst case, assuming each edge triggers an insertion into the heap. Using these top-k edges we estimate the parameters of a Normal distribution, which takes $O(k)$. Next we make another pass over the edge set and subject each to a membership test against the Normal model in $O(m)$. We induce the graph on all the edges that pass the test, the connected components of which yield the core sets. Overall complexity for finding the core sets is thus $O(m \log k)$.

For expanding a focused cluster, we enlist all the non-member neighbors as the candidate set C and evaluate their weighted Δ conductance. As discussed in §5.3.2, the complexity is $\sum_{n \in C} d(n)$.

Procedure P4 CONTRACT

Input: attributed graph $G(V, E, F)$, focused cluster C, current conductance $\phi_{curr}^{(w)}$
Output: a focused cluster C and its conductance $\phi_{curr}^{(w)}$

1: **repeat**
2: $removed \leftarrow false$
3: **for each** node n in C **do**
4: $\Delta\phi_n^{(w)} \leftarrow$ GETΔCONDUCTANCE(G, C, n,REMOVE)
5: **if** $\Delta\phi_n^{(w)} \leq 0$ **then**
6: $C \leftarrow C \backslash n$
7: $\phi_{curr}^{(w)} = \phi_{curr}^{(w)} + \Delta\phi_n^{(w)}$
8: $removed \leftarrow true$
9: **end if**
10: **end for**
11: **until** $removed = false$
12: **return** $C, \phi_{curr}^{(w)}$

Since $C \subseteq V$, it is equivalently $O(m)$. As we add one node at each iteration, the total complexity becomes $O(|S|m)$ where $|S|$ is the size of the focused cluster, and $|S| \ll n$. Also note that focused clusters can be extracted around each core set in parallel.

We remark that scanning candidate set C for picking the best node takes $O(m)$ in the worst case. Assuming small rounded focused clusters, one can expect that not all edges of G are "touched" by C's neighbors. This implies sub-linear performance for expanding a single cluster in practice.[3]

Variants of FOCUSCO

We conclude this section by briefly discussing a couple of variants of our problem setting and how we can adapt our proposed algorithm to handle these variants.

In one variant of the problem, the user might explicitly ask for the exemplar nodes s/he provided to be included in the focused clusters. While it is highly likely that most of the exemplar nodes will indeed be part of the focused clusters found in Algorithm 1, inclusion of all is not guaranteed. To handle this setting, we can include the connected components induced on the exemplar nodes as additional core sets (in P2 Line 7) and later never allow the removal of any exemplar node in extracting the clusters (in P4 Lines 5-9).

Another variant involves the user asking for a sparser representation of the focus attributes. In other words, it may be practical to define the similarity among the exemplar nodes using as few attributes as possible, especially in high dimensions. In such a case, we can tune the regularization constant γ that we introduced in Equation (5.1) for learning the weight vector $\boldsymbol{\beta}$. Specifically, a large γ drives the second term in the objective to become large. To make the pairs in P_D as

[3]Different ways of choosing the core sets (e.g., only using user-provided nodes) can remove the dependence on processing the entire graph, and allow FOCUSCO to run in sublinear time.

dissimilar as possible, a dense $\boldsymbol{\beta}$ is learned. The smaller the γ gets, the less the emphasis on the dissimilar pairs becomes, and a sparser weight vector is learned.[4]

In other settings, the user may choose to explicitly provide the set of dissimilar nodes or the attribute relevances (i.e., the $\boldsymbol{\beta}$ vector) directly, which can be incorporated trivially.

5.4 Evaluation

In this section we thoroughly evaluate our method[5] on clustering quality, outlier detection performance, and runtime on synthetic and real-world networks. None of the existing methods address the focused clustering and outlier detection problem we pose in this paper. Nevertheless, we compare to two representative techniques, CODA [51] and METIS [68]. CODA is a graph clustering and outlier detection algorithm on attributed graphs, and treats all attributes equally. The clustering is not steered by user-preference, as such, it clusters the whole graph. METIS is a graph partitioning algorithm and does not provide outlier detection. Both methods expect the number of clusters as input.

To evaluate *focused clustering* quality, we use the Normalized Mutual Information (NMI), a widely used metric for computing clustering accuracy of a method against the desired ground truth [85]. The ground truth in our case is the true focused clusters that are relevant to a particular user's interest. The best NMI score is 1. We evaluate *outlier detection* performance by the F1-score; the harmonic mean of precision and recall for a known set of outliers.

5.4.1 Results on Synthetic Graphs

Data generation

To study the behavior of our algorithm compared to other approaches on graphs with ground truth (focused) clusters and outliers, we generated synthetic graphs with various number of clusters focusing on different subsets of the attribute space, containing various number of clusters, and with varying size ranges. Our generative algorithm is based on the planted partitions model [35].

Simply put, given the desired number of nodes in each cluster we split the adjacency matrix into blocks defined by the partitioning. For each block B_{ij}, we choose a probability p_{ij}. Using a random draw process we assign a 1, i.e. an edge, for each possible entry in the block, and 0 otherwise. In other words, p_{ij} specifies the density of each block. The diagonal blocks constitute the actual clusters and off-diagonal entries yield the cross edges. Unless otherwise noted, we set $p_{ii} = 0.35$ and $0.10 \leq p_{ij} \leq 0.25$, $i \neq j$.

We assign the graph clusters generated, either to one of two focus attribute sets (i.e. focus-1 or focus-2) or as unfocused. Please note that in real-world graphs, we expect to see more than two focuses on a variety of attribute subspaces. For each focused cluster, one of the two subsets (focus-1 or focus-2) is chosen as focus attributes. For each attribute i in this subset the attribute values are drawn from a Normal distribution $N(\mu_i, \sigma)$ with uniform random mean $\mu_i \in [0, 1]$ and a variance

[4]For $\gamma = 0$, constraint on P_D is completely waived and $\boldsymbol{\beta}$ is zero.

[5]A FOCUSCO implementation is available at `http://bit.ly/focusedclustering`

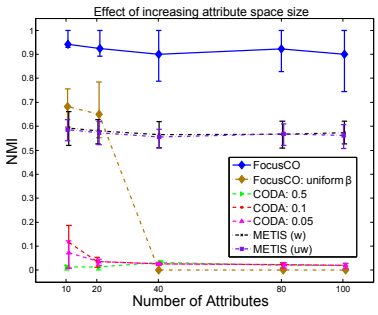


Figure 5.2: NMI vs. attribute size $|F|$. Results averaged over 100 runs, bars depict 25-75%.

$\sigma = 0.001$. The variance is specifically chosen to be small such that the clustered nodes "agree" on their focus attributes. The rest of the attributes, on the other hand, are drawn from a Normal distribution with much larger variance; $N(0, 1)$. In contrast, all of the attribute values of nodes in unfocused clusters are drawn from large-variance Normals.

Focused outliers are generated by randomly choosing members from each focused cluster and "deflating" (depending on the setting) one or more of their focus attributes i; by replacing them by a value drawn from $N(\mu_i, \sigma = 1)$.

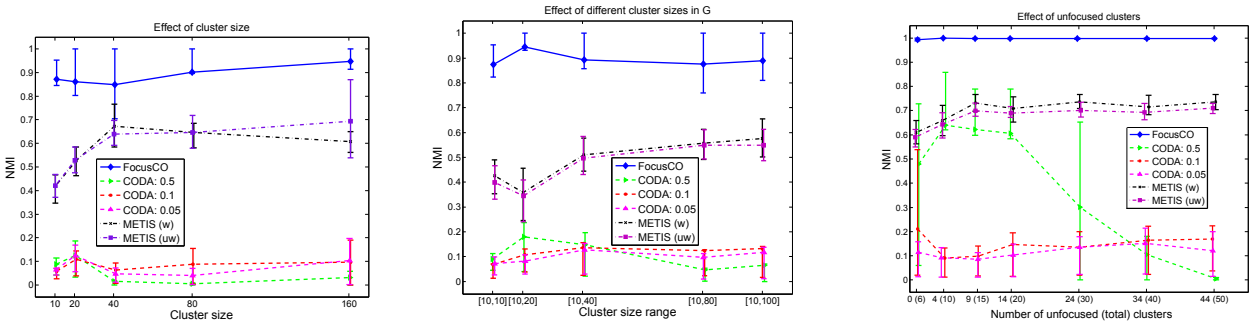
Clustering quality

To study the clustering performance, we generated graphs with 3 focused clusters that are coherent in focus-1 attributes, 3 focused clusters that are coherent in focus-2 attributes, and 3 unfocused clusters, for a total of 9 clusters. The task is to extract the focus-1 clusters with the respective user preference.

For comparison we use CODA and METIS, which do not perform focused cluster extraction. They both partition the entire graph, and thus, we explicitly need to select the 3 best clusters that these methods returned, by measuring the overlap among the clusters they produced to the ground-truth clusters. Since both methods require the number of clusters to be provided as input, we asked for the correct number of (9) clusters, i.e. best performance, although in practice this number is hard to choose as the number of hidden clusters is unknown, especially for large graphs.

METIS is a graph partitioning algorithm that does not handle attributes. In one version of METIS, we ignore the attributes and use only the graph structure. In a second version, we incorporate attribute information by weighing the edges of the graph (using $\boldsymbol{\beta}$) by the attribute similarity of their end nodes. We call these two versions as weighted METIS (w) and unweighted METIS (uw). While CODA can perform clustering for attributed graphs, a main challenge with it is to carefully choose a λ parameter that controls the trade-off between structural and attribute similarity of the nodes. In our experiments we report results using 3 different λ settings, 0.05, 0.1, and 0.5, for CODA.

Figure 5.2 shows the clustering performance (mean NMI over 100 independent runs) of the methods when we increase the number of attributes while retaining the same number of (5) focus attributes for the focused clusters. We observe that FOCUSCO remains superior to all the other approaches in the face of irrelevant attributes for the clustering task.



(a) NMI vs. cluster size (b) NMI vs. variable cluster size (c) NMI vs. # unfocused clusters

Figure 5.3: NMI clustering quality results on synthetic graphs, for FOCUSCO, CODA with three different λ parameter settings, and METIS on un/weighted graphs; (a) for changing cluster sizes (all clusters have the same size), (b) for changing cluster size variance (graph has variable size clusters), and (c) for increasing number of unfocused clusters. FOCUSCO performs the best in all scenarios across a wide range of settings. Symbols depict the mean over 100 runs, bars depict 25-75%.

To illustrate the importance of weight learning, we also study the performance of a variant of our FOCUSCO, in which we use a uniform attribute weight vector $\boldsymbol{\beta}$, i.e., we bypass weight learning and directly perform cluster extraction. We observe that the performance of this version of our algorithm drops quickly with increasing attribute size. Our analysis suggests that this occurs due to the edge weights having a more and more uniform distribution when weighted by using a uniform $\boldsymbol{\beta}$, which yields an inferior collection of core sets around which we find clusters. Thus, we proceed with studying the performance of our original FOCUSCO.

Next in Figure 5.3 we show the clustering performance of the methods under various other settings. In (a), we increase the cluster size where we create clusters of the same size in the graph. In (b), we allow the graph to contain variable size clusters and increase the variance of the cluster sizes, by randomly drawing them from increasing ranges. Finally in (c), we increase the number of unfocused clusters in the graph, while keeping the number of focused clusters fixed.[6] Notice that recovering a few focused clusters of interest in the existence of more unfocused clusters is an increasingly challenging problem.

From all these setups, we observe that FOCUSCO outperforms the competing methods and their variants in all scenarios. Weighted METIS seems to achieve slightly better performance than the unweighted version, although the differences are not significant. CODA's accuracy is the lowest, as the homophily assumption it is making does not hold in the full attribute space. We note that in (c), one parameterization of CODA ($\lambda = 0.5$) achieves as high accuracy as METIS for small number of clusters. Its accuracy, however, quickly drops when many more unfocused clusters than focused ones are introduced. Other two parameterizations give low accuracy, pointing out the sensitivity of CODA to the choice of its parameter.

Finally, we study the clustering performance when focus-1 and focus-2 clusters share common focus attributes. We create 20 node attributes out of which the first 10 are assigned as focus-1

[6]To ensure sparsity of the growing graphs, we set $p_{ij} = 0.02$.

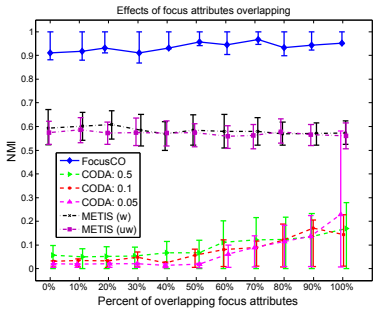


Figure 5.4: Clustering performance by increasing overlap on focus attributes of different focused clusters. (mean NMI over 100 runs, bars: 25-75%).

attributes and the next 10 are assigned as focus-2 attributes to the respective clusters. Then, we gradually overlap the focus attributes until they are the same set of 10 for all the focused clusters. Figure 5.4 shows that the performance of FOCUSCO remains stable and high across all overlaps. The accuracy of METIS (w) is also quite stable and stays around 0.6 (METIS (uw) is not expected to be affected in this setup). We also notice that CODA's performance starts increasing after more than 50% of the focus attributes overlap. At 100%, there is essentially only a single focus in the graph, where CODA's performance peaks. This suggests that CODA is more suitable for attributed graphs with uniform graph clusters and would suffer when the graph contains many heterogeneous focused clusters (with multiple focuses) as we would expect to see in real networks.

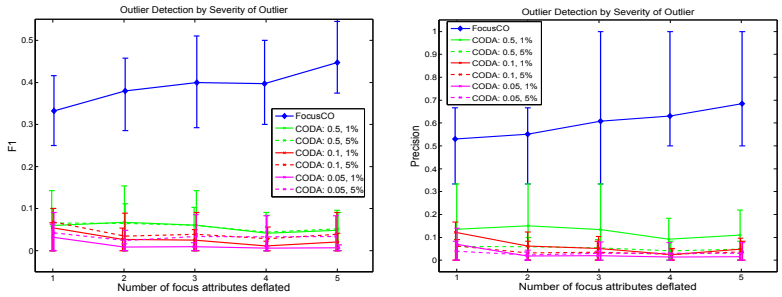
These results show the robustness of FOCUSCO, where its performance remains quite stable across different settings. They also illustrate that the general graph clustering methods are not suitable for our focused clustering problem, as their performance is (i) sensitive to the (parameter) setting, and (ii) lower than that of FOCUSCO at all settings.

Outlier detection

Next we evaluate outlier detection performance. Since METIS does not detect outliers, we compare to CODA. In addition to its λ parameter, CODA expects a parameter r that controls the top percentage of nodes to be returned as outliers. We report experiments for the cross-product of $\lambda = \{0.05, 0.1, 0.5\}$ and $r = \{1\%, 5\%\}$.

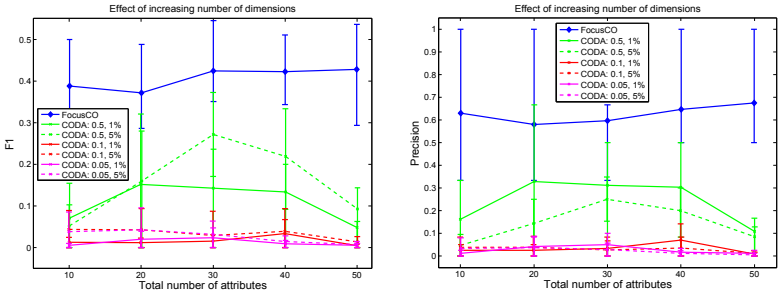
In the first setup, we study the performance with respect to the severity of outliers. If an outlier deviates in a larger number of focus attributes from its cluster members, it becomes more severe in outlierness, but easier to detect. To create outlier nodes with higher outlierness, we gradually increase their number of focus attributes that we deflate. Figure 5.5 shows the F1-score and precision (averaged over 100 runs). We observe that FOCUSCO achieves superior performance to CODA in both metrics. We can also notice the increase in detection accuracy of FOCUSCO with increasing number of deflated focus attributes (i.e., increasing ease in spotting outliers), as we expected, while the same trend is not apparent for CODA potentially because it does not calibrate to attribute subspaces.

We further analyze the outlier detection accuracy with respect to the attribute space size. In Figure 5.6 we observe that CODA's performance starts dropping after a certain number of attributes,



(a) F1-score (b) precision

Figure 5.5: Outlier detection performance by increasing number of deflated focus attributes.



(a) F1-score (b) precision

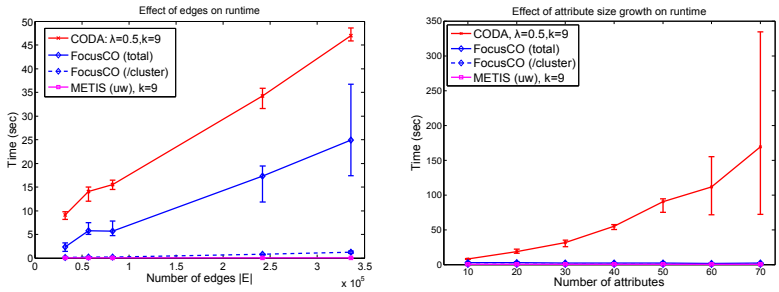
Figure 5.6: Outlier detection performance by increasing number of attributes $|F|$.

where identifying descriptive focus attributes becomes more and more crucial to accurately spot the outliers that are hidden in different community structures. The performance of FOCUSCO on the other hand remains quite stable and superior to CODA.

Finally, we note that the precision of FOCUSCO is often higher than its recall, while both being superior to CODA's.

Scalability

Finally we study the scalability of the methods. Figure 5.7 shows the running times with increasing number of edges and attributes. CODA's inference techniques are computationally demanding, and thus, its running time is much higher than other methods. METIS on the other hand uses an extremely efficient heuristic and achieves low running times. We report the average cluster extraction time for FOCUSCO in addition to total running time, as each cluster extraction can be performed in parallel. In fact, we notice that while its total running time increases by graph size, average time to extract a single cluster remains stable and low. In such a case, FOCUSCO is also comparable to METIS.



(a) time vs. #edges (b) time vs. #attributes

Figure 5.7: Running time scalability w.r.t. (a) number of edges $|E|$ and (b) number of attributes $|F|$.

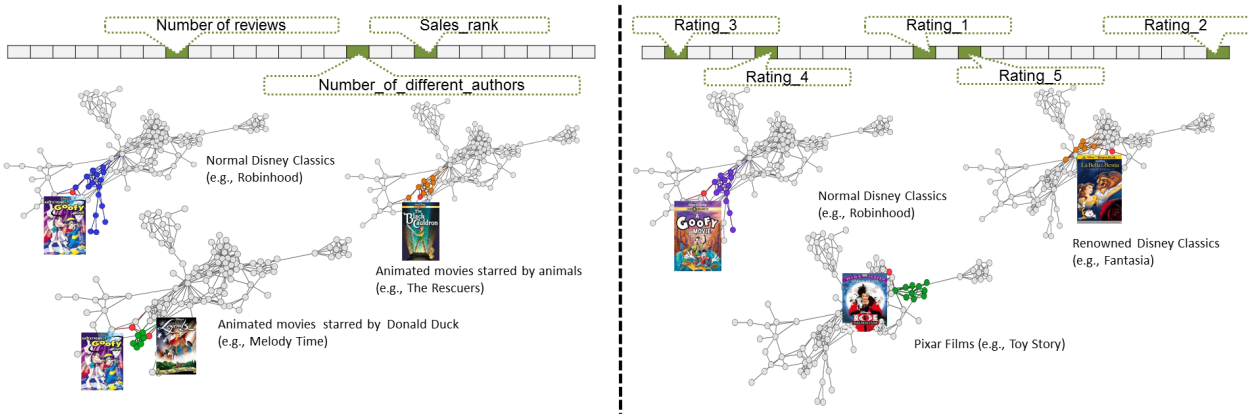


Figure 5.8: Two sets of focused clusters in DISNEY. Left clusters focus on attributes related to popularity (sales rank, number of reviews, etc.) Nodes in right clusters share similar ratings. Outliers are marked with red and illustrated. See text for discussion. (best viewed in color)

5.4.2 Results on Real-world Graphs

Dataset description

We use several attributed networks obtained from real-world data to evaluate our approach. DISNEY is an *Amazon* co-purchase graph of Disney movies.[7] Each movie has 28 attributes such as price, rating, number of reviews, etc. POLBLOGS is the citation network among a collection of online blogs that discuss political issues. Attributes are the keywords in their text. DBLP and 4AREA are two different co-authorship networks of computer science authors. The attributes reflect the conferences which an author has published in broadly (DBLP), or just in databases, data mining, information retrieval, and machine learning (4AREA). Finally, we have the friendship relations of YOUTUBE users and the attributes depict their group memberships. Dataset statistics are given in Table 5.2.

[7] http://www.ipd.kit.edu/~muellere/consub/

Dataset	$\|V\|$	$\|E\|$	$\|F\|$	Running time (sec)
DISNEY	124	333	28	0.0017 ± 0.0022 (8.3)
POLBLOGS	362	1288	44839	0.0040 ± 0.0052 (2.8)
4AREA	27199	66832	4	0.0052 ± 0.0018 (3390.9)
DBLP	30599	146647	18	0.2868 ± 0.0630 (761.4)
YOUTUBE	77381	367151	30087	2.9643 ± 0.7201 (257.4)

Table 5.2: Real-world datasets used in this work. Average running time in seconds per cluster $\pm$ std (avg. number of clusters extracted).

Running time

Our real datasets come from various domains and have different node, edge, and attribute counts. Here we report running time experiments to demonstrate the efficiency of our method on these real graphs.

We setup 10 runs of our method on each graph, each with a randomly sampled 1% of the attributes as the focus attributes. Each run returns a different number of clusters, thus we report the running time per cluster averaged over the 10 runs and their standard deviations in Table 5.2. Notice that similar to Figure 5.7, the running times are quite low. In particular, the average time to extract a cluster in our largest graph YOUTUBE takes around 3 seconds.

Case Studies

The first case study we consider is finding two types of focused clusters in DISNEY. In one instance, a user wants to understand how the popularity of a movie influences its community in a co-purchase network. The user decides that the product's popularity is related to the features `Number_of_reviews` and `Sales_rank`, and so chooses a few products which have similar values in those attributes. FOCUSCO then uses this exemplar set C_{ex} to learn an attribute weighting $\boldsymbol{\beta_u}$. This $\boldsymbol{\beta_u}$ reflects the user's intent, and has also captured another dimension correlated with those attributes; `Number_of_different_authors`.

Several extracted focused clusters for this task are shown on the left in Figure 5.8. In general, the discovered clusters consist of movies of similar age and acclaim. The first focused cluster (blue) reflects traditional Disney classics such as `Robinhood`. Its outlier is a sequel (`An Extremely Goofy Movie`) that is much less popular than the other classics in the cluster. The second community (green) focuses on popular older Disney movies, and has outliers such as `American Legends` and again the Goofy sequel, that are much less popular. The third cluster (orange) *overlaps* with the first focused cluster. It is a subset of the classic Disney movies of the larger cluster that were predominantly starred by animals (e.g., `The Rescuers`). Its cluster outlier is `The Black Caldron`, which although of similar vintage, starred a human protagonist and was much less popular.

In the next instance, a user wants to examine how the differences in the distribution of consumer ratings affect the DISNEY clustering. In this case, the focused clusters represent collections of movies that are similarly rated (e.g., Pixar films or animated Disney classics). The outliers represent

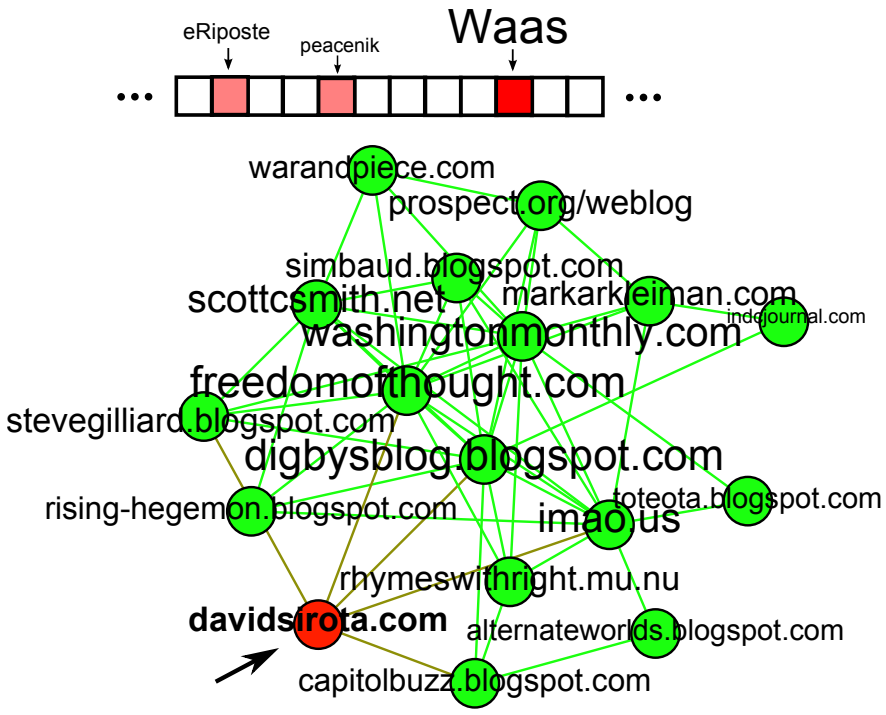


Figure 5.9: A focused cluster of liberal blogs in POLBLOGS with a focus on Iraq war debate. Outlier `David Sirota` does not mention `Waas` in his posts.

movies which are rated differently from the movies they are purchased with, and reflect consumer opinion. The results are shown in the right of Figure 5.8. The first focused cluster (purple) represents traditional Disney classics. Its outlier (which was included in the previous focused popularity cluster) is the movie `A Goofy Movie` which although reasonably popular, is not as high rated. The second cluster (green) is the Pixar community, featuring high-rated movies like `Toy Story`. Its focused outlier is the live action version of `101 Dalmatians`, a movie which is rated quite differently than most Pixar films. The third cluster consists of renowned Disney films such as `Fantasia`. It also contains an outlier, the Spanish version of `Beauty and the Beast`.

We consider a second case study on POLBLOGS, where a user seeks to understand the difference between blog content written by different liberal bloggers during the height of the Iraq war controversy in 2005. Using several liberal blogs as an example set, the user learns a β_u which represents the focus of the bloggers (as shown in the top of Figure 5.9, text size of features proportional to weight). It contains words such as `eriposte`, an active liberal blogger, `peacenik`, a term for an anti-war activist, and most strongly `Waas`. Murray Waas was an independent journalist praised for his investigative journalism of the Bush administration.[8] The focused community outlier for this group is David Sirota, a well-connected liberal blogger who did not explicitly mention Waas in the dataset.

The third case study we consider is on the 4AREA dataset. Here a user wants to understand who the outliers are in the data mining co-authorship network. S/he learns

[8] `http://www.huffingtonpost.com/jay-rosen/murray-waas-is-the-woodwa_b_18875.html`

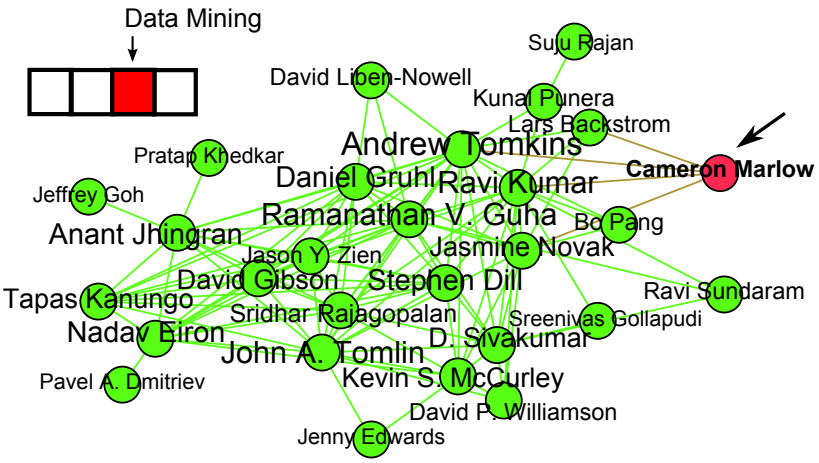


Figure 5.10: A focused cluster of `data mining` researchers in 4AREA. Outlier `Cameron Marlow` closely publishes with them, but on `information retrieval`.

a β_u using {`Christos Faloutsos, Jiawei Han, Jon M. Kleinberg, Jure Leskovec, Andrew Tomkins`} as input, who focus primarily on data mining. One of the focused clusters found is around data mining researchers mostly in industry, as shown in Figure 5.10. The outlier is `Cameron Marlow`, the former head of Facebook's data science team, who collaborated with the researchers in the data mining community but published on information retrieval.

5.5 Conclusion

In this work we introduce a new problem of finding *focused clusters and outliers* in large attributed graphs.

Given a set of exemplar nodes that capture the user interest, our goal is two-fold: 1) "chop out" clusters of similar nodes that are densely connected and exhibit coherence in a subset of their attributes, called the focus attributes, and 2) identify focused outliers, i.e. nodes that belong to a focused cluster in network structure but show deviance in some focus attribute(s). We propose an efficient algorithm that infers the focus attributes of interest to the user, and that both extracts focused clusters and spots outliers simultaneously. Experiments on synthetic and real-world graphs show the effectiveness and scalability of our approach and that the existing graph clustering and outlier detection techniques are not suitable to handle the newly posed problem.

Chapter 6

Inducing Language Networks from Continuous Space Word Representations

6.1 Introduction

Unsupervised feature learning (*deep learning*) utilizes huge amounts of raw data to learn representations that model knowledge structure and disentangle the explanatory factors behind observed events. Under this framework, symbolic sparse data is represented by lower-dimensional continuous spaces. Integrating knowledge in this format is the secret behind many recent breakthroughs in machine learning based applications such as speech recognition, computer vision, and natural language processing (NLP).

We focus here on word representations (*word embeddings*) where each word representation consists of a dense, real-valued vector. During the pre-training stage, the representations acquire the desirable property that similar words have lower distance to each other than to unrelated words [64]. This allows the representations to utilize the abundance of raw text available to learn features and knowledge that is essential for supervised learning applications such as part-of-speech tagging, named entity recognition, machine translation, language modeling, sentiment analysis etc [34, 55, 91, 124].

Several methods and algorithms have been proposed to learn word representations along different benchmarks for evaluation [29]. However, these evaluations are hard to comprehend as they squash the analysis of the representation's quality into abstract numbers. To enable better understanding of the actual structure of word relationships which have been captured, we have to address the problems that come with analyzing high-dimensional spaces (typically between 50-1000 dimensions). We believe that network induction and graph analysis are appropriate tools to give us new insights.

In this work, we seek to induce meaningful graphs from these continuous space language models. Specifically, our contributions include:

This work originally appeared as "Bryan Perozzi, Rami Al-Rfou, Vivek Kulkarni, and Steven Skiena. Inducing language networks from continuous space word representations. In *Complex Networks V*, volume 549 of *Studies in Computational Intelligence*, pages 261–273. 2014."

- **Analysis of Language Network Induction** - We propose two criteria to induce networks out of continuous embeddings. For both methods, we study and analyze the characteristics of the induced networks. Moreover, the networks generated lead to easy to understand visualizations.

- **Comparison Between Word Representation Methods** - We evaluate the quality of two well known words embeddings. We contrast between their characteristics using the analysis developed earlier.

The remainder of this paper is set up as follows. First, in Section 6.2, we describe continuous space language models that we consider. In Section 6.3, we discuss the choices involved with inducing a network from these embeddings and examine the resulting networks. Finally, we finish with a discussion of future work and our conclusions.

6.2 Continuous Space Language Models

The goal of a language model is to assign a probability for any given sequence of words estimating the likelihood of observing such a sequence. The training objective usually maximizes the joint probability of the training corpus. A continuous space probabilistic language model aims to estimate such probability distribution by, first, learning continuous representations for the words and phrases observed in the language. Such mapping is useful to cope with the curse of dimensionality in cases where data distribution is sparse as natural language. Moreover, these representations could be used as features for natural language processing applications, domain adaptation and learning transfer scenarios that involve text or speech.

More precisely, given a sequence of words $S = [w_1 \dots w_k]$, we want to maximize $P(w_1, \dots, w_k)$ and learn representations for words. During the training process the continuous space language model learns a mapping of words to points in $\mathbb{R}^d$, where d usually ranges between $20 - 200$. Prior to training we build a vocabulary V that consists of the most frequent $|V|$ words, we map each word to a unique identifier that indexes an embeddings matrix C that has a size of $|V| \times d$. The sequence S is now represented by a matrix $\left[C[w_1]^T \quad \dots \quad C[w_k]^T\right]^T$, enabling us to compose a new representation of the sequence using one of several compositional functions. The simplest is to concatenate all the rows in a bigger vector with size kd. Another option is to sum the matrix row-wise to produce a smaller representation of size d. While the first respects the order of the words, it is more expensive to compute.

Given a specific sequence representation as an input, we will define a task that the model should solve, given the sequence representation as the only input. Our choice of the task ranges from predicting the next/previous word(s) to distinguishing between observed phrases and other corrupted copies of them. The chosen task and/or the compositional function influence the learned representations greatly as we will discuss later.

We will focus our investigations, here, on two embeddings which are trained with different tasks and compositional functions; the Polyglot and SkipGram embeddings.

6.2.1 Polyglot

The Polyglot project offers word representations for each language in Wikipedia [5]. For large enough Wikipedias, the vocabulary consists of the most frequent 100,000 words. The representations are learned through a procedure similar to the one proposed by [34]. For a given sequence of words $S_t = [w_{t-k} \dots w_t \dots w_{t+k}]$ observed in the corpus T, a corrupted sequence S'_t will be constructed by replacing the word in the middle w_t with a word w_j chosen randomly from the vocabulary V. Once the vectors are retrieved, we compose the sequence representation by concatenating the vectors into one vector called the projection layer S_t. The model is penalized through the hinge loss function,

$$\frac{1}{T}\sum_{t=1}^{t=T} |1 - score(S'_t) + score(S_t)|_+$$

where $score$ is calculated through a hidden layer neural network

$$score(S_t) = W_2(tanh(W_1 S_t + b_1)) + b_2.$$

For this work, we use the Polyglot English embeddings[1] which consist of the 100,000 most frequent words in the English Wikipedia, each represented by a vector in $\mathbb{R}^{64}$.

6.2.2 SkipGram

While the Polyglot embeddings consider the order of words to build the representation of any sequence of words, the SkipGram model proposed by [92] maximizes the average log probability of the context words independent of their order

$$\frac{1}{T}\sum_{t=1}^{T}\Big[\sum_{j=-k}^{k} \log p(w_{t+j}|w_t)\Big]$$

where k is the size of the training window. This allows the model to scale to larger context windows. In our case, we train a SkipGram model[2] on the English Wikipedia corpus offered by the Polyglot project for the most frequent 350,000 words with context size k set to 5 and the embeddings vector size set to 64.

6.2.3 Random

In order to have a baseline, we also generate random embeddings for the most frequent 100,000 words. The initial position of words in the Polyglot embeddings were sampled from a uniform distribution, therefore, we generate the random embedding vectors by sampling from $\mathcal{U}(\bar{m} - \sigma, \bar{m} + \sigma)$, where $\bar{m}$ and σ are the mean and standard deviation of the trained Polyglot embeddings' values respectively. This baseline allows us to see how the language networks we construct differ from networks induced from randomly initialized points.

[1] Polyglot embeddings and corpus available at `http://bit.ly/embeddings`

[2] SkipGram training tool available at `https://code.google.com/p/word2vec/`

6.3 Word Embedding Networks

We now consider the problem of constructing a meaningful network given a continuous space language model. As there are a variety of ways in which such a network could be induced, we start by developing a list of desirable properties for a language network. Specifically, we are seeking to build a network which:

1. **Is Connected** - In a connected graph, all the words can be related to each other. This allows for a consistent approach when trying to use the network to solve real-world problems.

2. **Has Low Noise** - Minimizing the spurious correlations captured by our discrete representation will make it more useful for application tasks.

3. **Has Understandable Clusters** - We desire that the community structure in the network reflects the syntactic and semantic information encoded in the word embeddings.

We also require a method to compute the distance in the embedding space. While there are a variety of metrics that could be used, we found that Euclidean distance worked well. So we use:

$$dist(x, y) = ||x - y||_2^2 = (\sum_{i=1}^{m}(x_i - y_i)^2)^{(1/2)} \tag{6.1}$$

where x and y are words in an d-dimensional embedding space ($x, y \in \mathbb{R}^d$). With these criteria and a distance function in hand, we are ready to proceed. We examine two approaches for constructing graphs from word embeddings, both of which seek to link words together which are close in the embedding space. For each method, we induce networks for the $20,000$ most frequent words for each embedding type, and compare their properties.

6.3.1 k-Nearest Neighbors

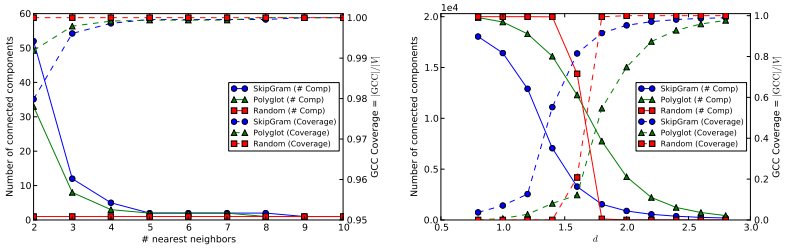
The first approach we will consider is to link each word to the k closest points in the embedding space. More formally, we induce a set of directed edges through this method:

$$E_{knn} = \{(u, v) : \min_x dist(u, v)\} \quad \forall u, v \in V, x \leq k \tag{6.2}$$

where $\min_x$ denotes the rank of the x-th number in ascending sorted order (e.g. $\min_0$ is the minimum element, $\min_1$ the next smallest number). After obtaining a directed graph in this fashion, we convert it to an undirected one.

The resulting undirected graph does not have a constant degree distribution. This is due to the fact that the nearest-neighbor relation may not be symmetric. Although all vertices in the original directed graph have an out-degree of k, their orientation in the embedding space means that some vertices will have higher in-degrees than others.

Results from our investigation of basic network properties of the k-NN embedding graphs are shown in Figures 6.1 and 6.2. In (6.1a) we find that the embedding graphs have few disconnected components, even for small values of k. In addition, there is an obvious GCC which quickly



(a) k-NN Coverage **(b) d-Threshold Coverage**

Figure 6.1: Graph Coverage. The connected components and relative size of the Giant Connected Component (GCC) in graphs created by both methods. We see that very low values of k quickly connect the entire network (6.1a), while values of d appear to have a transition point before a GCC emerges (6.1b).

emerges. In this way, the embeddings are similar to the network induced on random points (which is fully connected at $k = 2$). We performed an investigation of the smaller connected components when k was small, and found them to contain dense groupings of words with very similar usage characteristics (including ordinal values, such as Roman numerals (`II, III, IV`)).

In (6.2a) we see that the clustering coefficient initially grows quickly as we add edges to our network ($k \leq 6$), but has leveled off by ($k = 20$). This tendency to bridge new clusters together, rather than just expand existing ones, may be related to the *instability* of the nearest neighbor [18] in high dimensional spaces. In (6.2b), we see that the networks induced by the k-NN are not only connected, but have a highly modular community structure.

6.3.2 d-Proximity

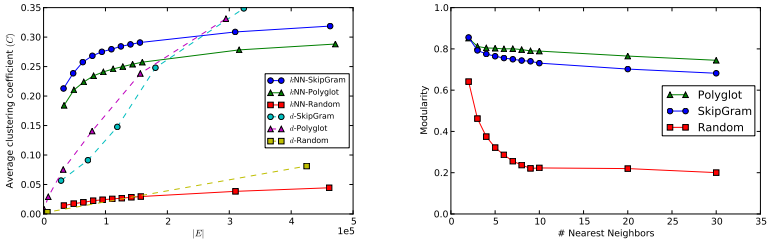
The second approach we will consider is to link each word to all those within a fixed distance d of it:

$$E_{proximity} = \{(u, v) : dist(u, v) < d\} \quad \forall u, v \in V \tag{6.3}$$

We perform a similar investigation of the network properties of embedding graphs constructed with the d-Proximity method. The results are shown in Figures 6.1 and 6.2. We find that networks induced through this method quickly connect words that are near each other in the embedding space, but do not bridge distant groups together. They have a large number of connected components, and connecting 90% of the vertices requires using a relatively large value of d (6.1b).

The number of connected components is closely related to the average distance between points in the embedding space (around d =(3.25, 3.80, 2.28) for (SkipGram, Polyglot, Random)). As the value of d grows closer to this average distance, the graph quickly approaches the complete graph.

Figure 6.2a shows that as we add more edges to the network, we add triangles at a fast rate than using the k-NN method.



(a) Clustering Coeff., C **(b) k-NN Modularity, Q_{knn}**

Figure 6.2: Community Metrics. In (6.2a), C shown for k = [2,30] and d = [0.8,1.6] against number of edges in the induced graph. When the total number of edges is low ($|E| < 150,000$), networks induced through the k-NN method have more closed triangles than those created through d-Proximity. In (6.2b), Q_{knn} starts high, but slowly drops as larger values of k include more spurious edges.

6.3.3 Discussion

Here we discuss the differences exposed between the methods for inducing word embeddings, and the differences exposed between the embeddings themselves.

Comparison of Network Induction Methods.

Which method then, provides the better networks from word embeddings? To answer this question, we will use the properties raised at the beginning of this section:

1. **Connectedness** - Networks induced through the k-NN method connect much faster (as a function of edges) than those induced through d-Proximity (Fig. 6.1). Specifically, the network induced for $k = 6$ has nearly full coverage (6.1a) with only 100K edges (6.2a).

2. **Spurious Edges** - We desire that our resulting networks should be modular. As such we would prefer to add edges between members of a community, instead of bridging communities together. For low values of $|E|$, the k-NN approach creates networks which have more closed triangles (6.2a). However this does not hold in networks with more edges.

3. **Understandable Clusters** - In order to qualitatively examine the quality of such a language network, we induced a subgraph with the k-NN of the most frequent 5,000 words in the Polyglot embeddings for English. Figure 6.3 presents the language network constructed for ($k = 6$).

According to our three criteria, k-NN seems better than d-Proximity. In addition to the reasons we already listed, we prefer k-NN as it seems to require less parameterization (d-Proximity has a different optimal d for each embedding type).

Comparison of Polyglot and SkipGram.

Having chosen to use k-NN as our preferred method for inducing language networks, we now examine the difference between the Polyglot and SkipGram networks.

Clustering Coefficient. We note that in Figure 6.2a, the SkipGram model has a consistently higher clustering coefficient than Polyglot in k-NN networks. A larger clustering coefficient denotes more triangles, and this may indicate that points in the SkipGram space form more cohesive local clusters than those in Polyglot. Tighter local clustering may explain some of the interesting regularities observed in the SkipGram embedding [94].

Modularity. In Figure 6.2b, we see that Polyglot modularity is consistently above the SkipGram modularity. SkipGram's embeddings capture more semantic information about the relations between words, and it may be that causes a less optimal community structure than Polygot whose embeddings are syntactically clustered.

Clustering Visulizations. In order to understand the differences between the language networks better, we conducted an examination of the clusters found using the Louvain method [21] for modularity maximization. Figure 6.4 examines communities from both Polyglot and SkipGram in detail.

6.4 Related Work

Here we discuss the relevant work in language networks, and word embeddings. There is also related work on the theoretical properties of nearest neighbor graphs, consult [41] for some basic investigations.

6.4.1 Language Networks

Word Co-occurrences. One branch of the study of language as networks seeks to build networks directly from a corpus of raw text. [24] examine word co-occurrence graphs as a method to analyze language. In their graph, edges connect words which appear below a fixed threshold ($d \leq 2$) from each other in sentences. They find that networks constructed in this manner show both small world structure, and a power law degree distribution. Language networks based on word co-occurrence have been used in a variety of natural language processing tasks, including motif analysis of semantics [19], text summarization [9] and resolving disambiguation of word usages [141].

Hypernym relations. Another approach to studying language networks relies on studying the relationships between words exposed by a written language reference. [99] use a thesaurus to construct a network of synonyms, which they find to find to exhibit small world structure. In [127], [127] investigate the graph structure of the Wordnet lexicon. They find that the semantic edges in Wordnet follow scale invariant behavior and that the inclusion of polysemous edges drastically raises the clustering coefficient, creating a small world effect in the network.

Relation to our work. Much of the previous work in language networks build networks that are prone to noise from spurious correlations in word co-occurrence or infrequent word senses [24, 127]. Dimensionality reduction techniques have been successful in mitigating the effects of noise in

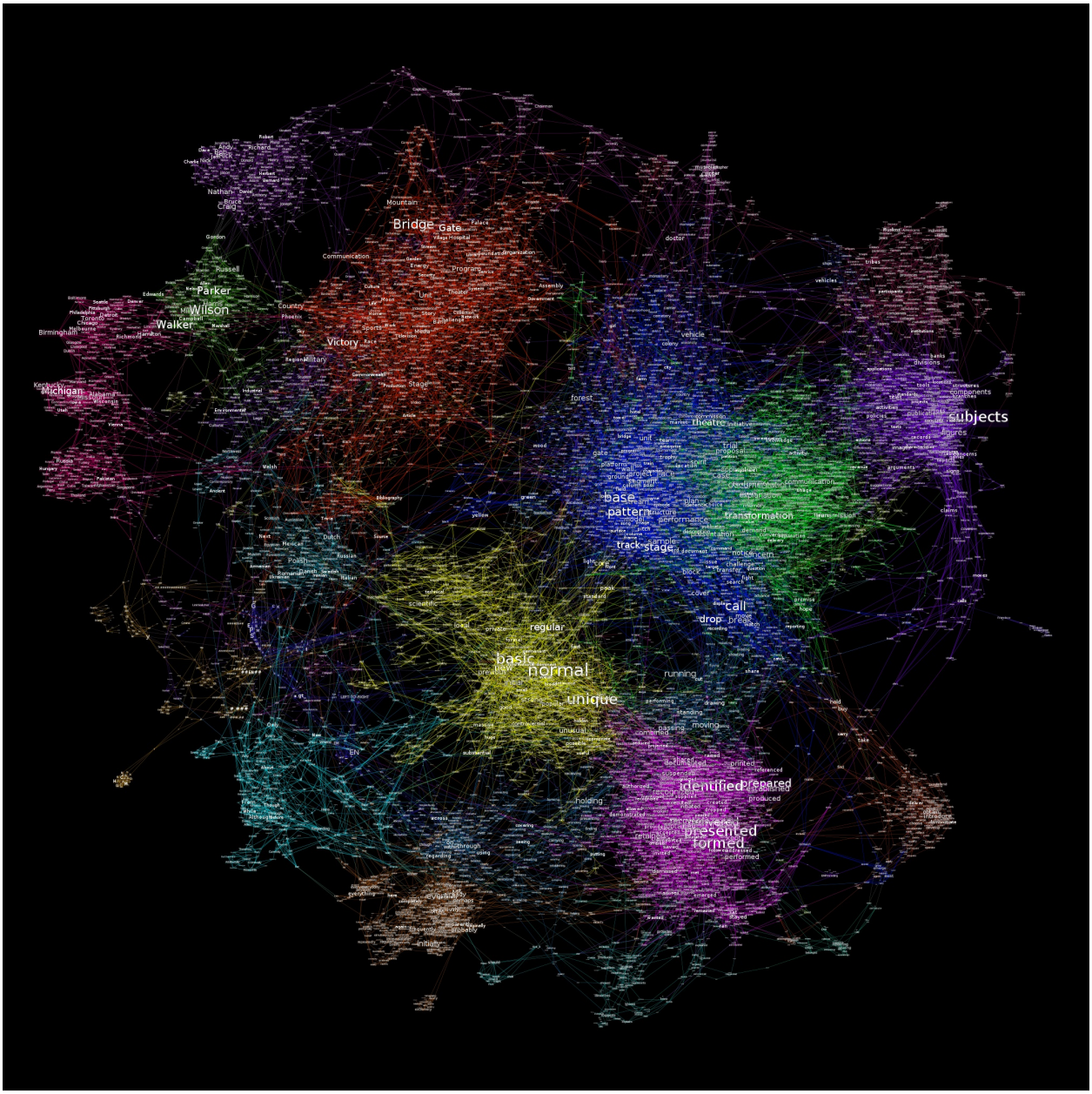
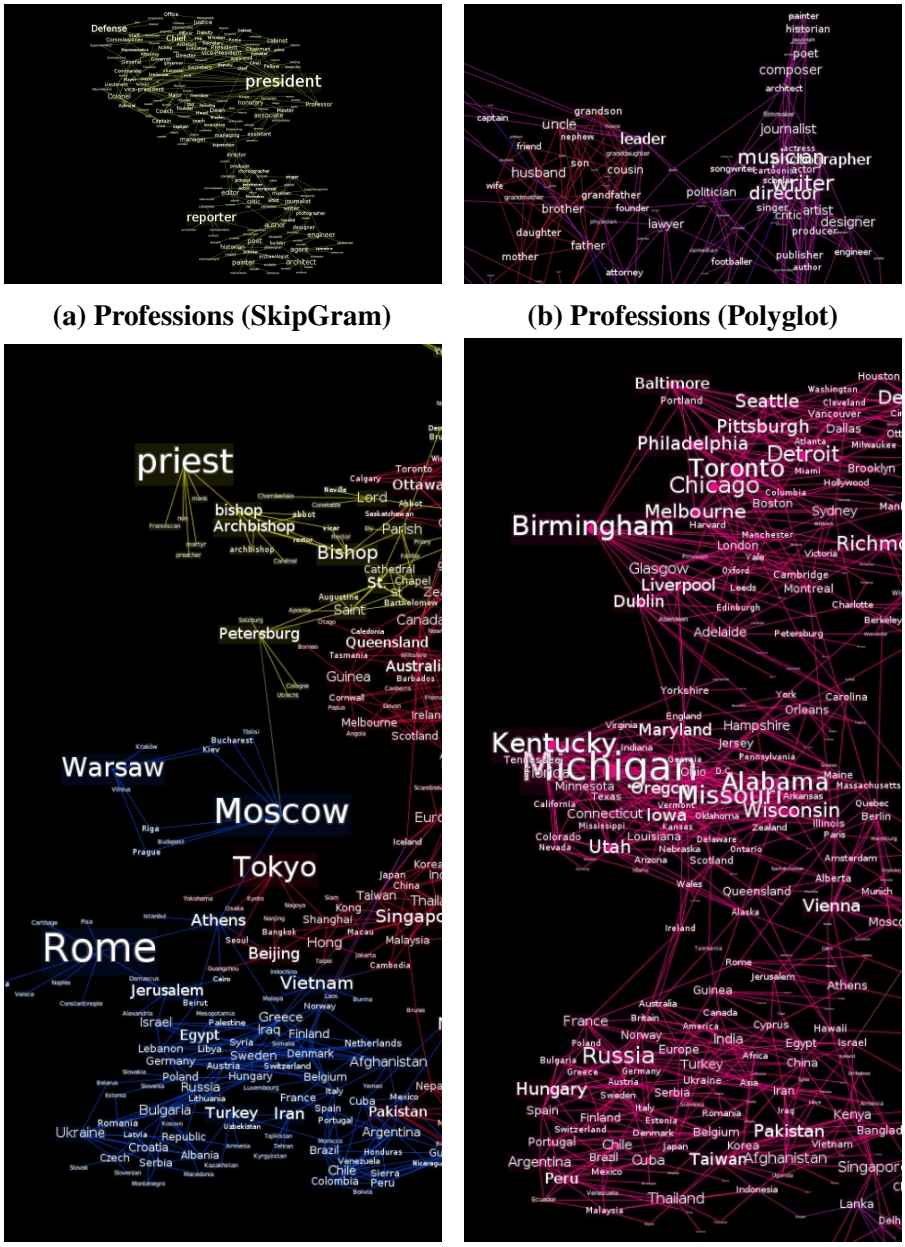


Figure 6.3: Polyglot Nearest Neighbor Graph. Here we connect the nearest neighbors ($k = 6$) of the top 5,000 most frequent words from the Polyglot English embeddings. Shown is the giant connected component of the resulting graph ($|V| = 11,239$; $|E| = 26,166$). Colors represent clusters found through the Louvain method (modularity $Q = 0.849$). Vertex label size is determined by its PageRank. Best viewed in color.

a variety of domains. The word embedding methods we examine are a form of dimensionality reduction that has improved performance on several NLP tasks and benchmarks.



(a) Professions (SkipGram) (b) Professions (Polyglot)

(c) Locations (SkipGram) (d) Locations (Polyglot)

Figure 6.4: Comparison of clusters found in Polyglot and SkipGram language networks. Polyglot clusters viewed in context of the surrounding graph, SkipGram clusters have been isolated to aide in visualization. SkipGram's bag-of-words approach favors a more semantic meaning between words, which can make its clusters less understandable (Note how in Figure 6.4c `Petersburg` is included in a cluster of religious words, because of `Saint`.) Images created with Gephi [12].

The networks produced in our work are considerably different from language networks created by previous work that we are aware of. We find that our degree distribution does appear to follow a power-law (like [24, 99, 127]) and we have some small world properties like those present in those works (such as $C \gg C_{random}$). However, the average path length in our graphs is considerably larger than the average path length in random graphs with the same node and edge cardinalities. Table 6.1 shows a comparison of metrics from different approaches to creating language networks.[3]

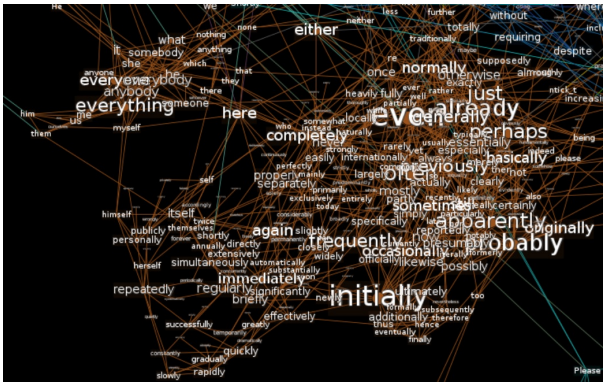
	$\|V\|$	$\|E\|$	C	C_{random}	pl	pl_{random}	γ
[24](UWN)	$478,773$	1.77×10^7	0.687	1.55×10^{-4}	2.63^*	3.03	-1.50,-2.70
[24](RWN)	$460,902$	1.61×10^7	0.437	1.55×10^{-4}	2.67^*	3.06	-1.50,-2.70
[99]	$30,244$	–	0.53	0.002	3.16	–	–
Polyglot, 6-NN	$20,000$	$96,592$	0.241	0.0004	6.78^*	4.62^*	-1.31
SkipGram, 6-NN	$20,000$	$94,172$	0.275	0.0004	6.57^*	4.62^*	-1.32

Table 6.1: A comparison of properties of language networks from the literature against those induced on the 20,000 most frequent words in the Polyglot and SkipGram Embeddings. (C clustering coefficient, pl average path length, γ exponent of power law fits to the degree distribution) '*' denotes values which have been estimated on a random subset of the vertices.

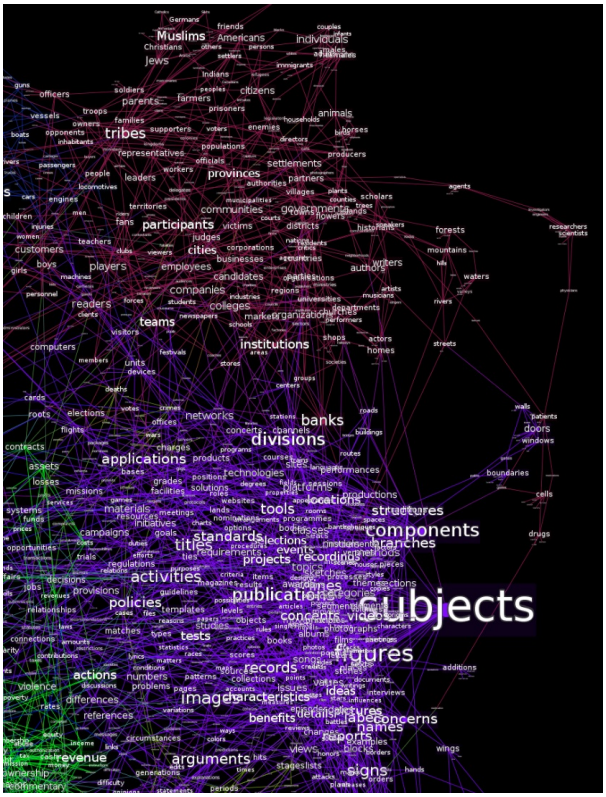
6.4.2 Word Embeddings

Distributed representations were first proposed by [63], to learn a mapping of symbolic data to continuous space. These representations are able to capture fine grain structures and regularities in the data [94]. However, training these models is slow due to their complexity. Usually, these models are trained using back-propagation algorithm [123] which requires large amount of computational resources. With the recent advancement in hardware performance, [16] used the distributed representations to produce a state-of-the-art probabilistic language model. The model maps each word in a predefined vocabulary V to a point in $\mathbb{R}^d$ space (word embeddings). The model was trained on a cluster of machines for days. More applications followed, [34] developed SENNA, a system that offers part of speech tagger, chunker, named entity recognizer, semantic role labeler and discriminative syntactic parser using the distributed word representations. To speed up the training procedure, importance sampling [14] and hierarchical softmax models [95, 97] were proposed to reduce the computational costs. The training of word representations involves minimal amount of language specific knowledge and expertise. [5] trained word embeddings for more than a hundred languages and showed that the representations help building multilingual applications with minimal human effort. Recently, SkipGram and Continuous bag of words models were proposed by [92] as simpler and faster alternatives to neural network based models.

[3] Our induced networks available at `http://bit.ly/inducing_language_networks`



(a) Pronouns & Adverbs



(b) Plurals

Figure 6.5: Additional close-ups of clusters in Polyglot embeddings (from Figure 6.3)

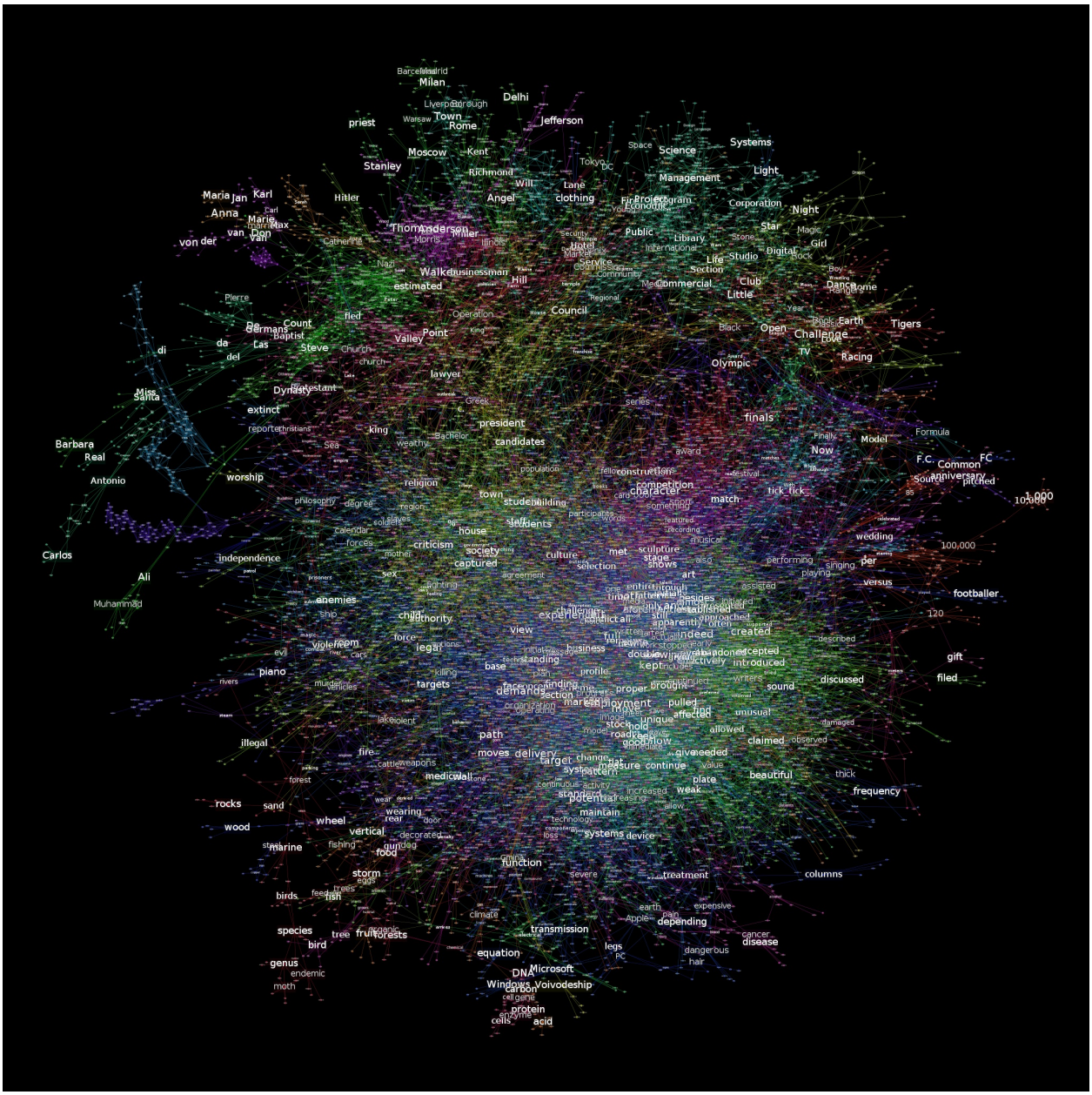


Figure 6.6: Visualization of the 6-NN for the GCC of the top 5,000 most frequent words in the SkipGram embeddings. SkipGram's representations are more semantic, and so language features like polysemous words make global visualization harder.

6.5 Conclusions

We have investigated the properties of recently proposed distributed word representations, which have shown results in several machine learning applications. Despite their usefulness, understanding the mechanisms which afford them their characteristics is still a hard problem.

In this work, we presented an approach for viewing word embeddings as a language network. We examined the characteristics of the induced networks, and their community structure. Using this analysis, we were able to develop a procedure which develops a connected graph with meaningful clusters. We believe that this work will set the stage for advances in both NLP techniques which utilize distributed word representations, and in understanding the properties of the machine learning processes which generate them.

Much remains to be done. In the future we would like to focus on comparing word embeddings to other well known distributional representation techniques (e.g. LDA/LSA), examining the effects of different vocabulary types (e.g. topic words, entities) on the induced graphs, and the stability of the graph properties as a function of network size.

Chapter 7

Conclusions

This thesis is concerned with the scalable modeling of attributed graphs. Its main contributions include:

- DEEPWALK, the first extension of deep learning paradigms into social network analysis (Chapter 2).
- WALKLETS, a technique for building explicit multiscale representations (Chapter 3).
- Normality, a community quality measure which can be efficiently optimized, even in the presence of high-dimensional attribute spaces (Chapter 4).
- Focused Clustering, query driven local clustering and outlier detection in attributed graphs (Chapter 5).
- An analysis of the underlying structure of popular word embedding techniques (Chapter 6).

7.1 Future Work

My original work on DEEPWALK illustrated how the machinery developed originally for neural language modeling can be applied to a very general class of graph analysis problems in data mining and information retrieval. Unsurprisingly, these new domains have raised a number of interesting new research questions. Here, I briefly describe a number of areas for additional investigation.

7.1.1 Extensions to New Graph Classes

As originally proposed, DEEPWALK only considers the induction of latent representations from unweighted, undirected, connected graphs. Since its release, there has been interest expressed from a number of researchers in applying it to a variety of other graphs which naturally occur in the course of data analysis. Briefly we summarize these extensions:

- **Weighted Graphs** - It is common for an edge E_{ij} to have a scalar weight w_{ij} associated with it. The natural extension is to modify DeepWalk's random walks to make non-uniform decisions about which neighbor to visit next (e.g. sample edges with respect to their weight). Interestingly, in our initial experiments we have observed that this does not necessarily lead to better representation performance on prediction tasks.

  We propose to study this behavior more fully. Specifically, we will quantify the sufficient and necessary structural conditions for various random walk strategies to work, and use these results to extend DeepWalk to work with a variety of weighed graphs.

- **Disconnected Graphs** - Real world graphs are frequently disconnected, which can pose a problem for graph embedding methods. Potential problems include: (1) This disconnection represents a point of extreme variance, which can cause the model to devote too much space to representing it. (2) If explicit space (i.e. its own dimensions) are not allocated for a dimension, then there are no constraints about where in the embedding space a disconnected component can lie. This can cause incorrect generalizations for linear classifiers, as the decision boundary will be set primarily by the largest component.

  We have showed in our recent work [71] that it is possible to align embeddings into a unified coordinate system. We propose developing graph factorization methods which intelligently align disconnected components in a similar way to allow sensible statistical learning in the presence of disconnected components.

- **Bi-partite Graphs** - In many networks the nodes can be divided into disjoint sets, such that all edges connect between the sets (and not inside of them). This effect is actually quite common, and occurs in product recommendation graphs, such as the prolific user-movie bipartite graph. Such networks can present challenges for random walk based methods, as paths which start in a node of a particular category may not end up in nodes of the same category (which can bias the underlying representations).

  In the literature it has been proposed to use a diffusion process which has a number of steps equal to the number of disjoint sets [49] (so random walks with even length for a bipartite graph). We propose investigating the effect of such disjoint sets on representation quality

In addition to investigating these new categories of graphs, it would be very desirable to quantify the conditions under which these representation modeling techniques work. We propose developing scalable graph quality statistics which allow for the predetermination of DEEPWALK hyper-parameters.

7.1.2 Speed Enhancements

DEEPWALK's performance on classification tasks comes at a cost - the simulation of random walks is an expensive procedure (in both time and space). Our next category of proposed enhancements relates to increasing the speed at which social representations can be induced.

- **Efficient Gradient Updates** - As originally presented, DEEPWALK uses a learning rate that linearly decays over the course of the algorithm. While performant, this approach assumes that we must go over the entire corpus before any single representation is learned. However, it is likely that much of these computations are unnecessary - the representations of high-degree nodes are learned much more quickly than low degree ones.

  In the literature, a number of alternative methods for updating the gradient have been proposed. Among them, AdaGrad [40] stands out as a theoretically sound method of accounting for frequency differences in gradient updates. Using AdaGrad, the representations for frequent nodes will coalesce quickly, but infrequent nodes have the flexibility to move. We propose extending DeepWalk's gradient updates in a similar fashion, and believe that this has the potential to greatly lower the amount of simulated random walks required to learn good representations.

- **Improved Graph Sampling** - Most real world graphs have degree distributions which follow a power-law (and are therefore very unbalanced). In a random walk on a connected graph, the expected number of times that a vertex is visited is directly related to its degree. As DEEPWALK uses such uniform random walks to sample the graph, it means that a large amount of the sampling is devoted to remembering information which is actually quite simple (high degree nodes are near many others). This means that a disproportionate amount of time and space are wasted on compressing easily compressible things. We propose investigating alternative sampling procedures, specifically:

  1. **Self Avoiding Random Walks** - A self-avoiding random walk (SARW) [74] is a random walk which does not visit the same node twice. Using a SARW could allow for a much smaller training corpus, as repetitive frequent nodes could be avoided.
  2. **Emphasizing Rare Nodes** - Currently, to ensure that DEEPWALK updates the representations of every node at least k times, we must start k random walk at each node. As random walks on undirected graphs are reversible (and independent), we can start two random walks of half the length at a node and join them - placing the node of interest in the middle. Each walk will now generate at least $2w$ updates of the starting node's representation.

     (This may also have implications regarding how quickly the graph can be covered [7].)

  These improvements which may allow learning representations with orders of magnitude less data.

7.1.3 Enhanced Representations

Another avenue for future work is to develop representations which encode more than just the structural information of a simple graph G. More formally, DEEPWALK currently learns representations (Φ) which model the probability that vertex i and j will co-occur in a short random walk, $Pr(v_i|\Phi(v_j))$. We propose extending these representations to model additional events, such

as the relation between attributes of the vertices. These enhanced representations will extend DEEPWALK-inspired graph factorization to work in a number of new domains.

- **Node Attributed Graphs** - Currently DEEPWALKś representations contain concise descriptions of structural information, which is useful for prediction tasks when a network exhibits homophily (i.e. correlations between structure and the attributes).

  We hypothesize that DEEPWALK's classification performance can be vastly improved by considering correlations in the label space itself. We propose extending DEEPWALK to account for these correlations, encoding some label information inside of the embeddings themselves.

- **Edge Attributed Graphs** - Edge attributed graphs have the capacity to model different types of relations between nodes. Each edge is a member of a particular type, and can either model a distinct semantic relations, or the state of the same relation at distant points in time (i.e, a temporal graph).

  We propose extending DEEPWALK to consider multiple edge types. By capturing different correlation patterns depending on edge type, we will enable better classification performance in these domains.

- **Knowledge Graphs & Heterogeneous Networks** - There has been a recent surge in interest in semantic graphs such as Knowledge Graphs and Heterogeneous Information Networks (HIN). These graphs are typically used to store relational information at large Internet companies (e.g. Google).

  Knowledge graphs (and HIN) can be modeled as node attributed graphs, edge attributed graphs, or a combination of the two. Extending DEEPWALK to embed such knowledge graphs will open a host of additional real world applications.

Bibliography

[1] Yong-Yeol Ahn, James P Bagrow, and Sune Lehmann. Link communities reveal multiscale complexity in networks. *Nature*, 466(7307):761–764, 2010.

[2] Leman Akoglu, Mary McGlohon, and Christos Faloutsos. Oddball: Spotting anomalies in weighted graphs. In *PAKDD*, 2010.

[3] Leman Akoglu, Hanghang Tong, Brendan Meeder, and Christos Faloutsos. PICS: Parameter-free identification of cohesive subgroups in large attributed graphs. In *SIAM SDM*, pages 439–450, 2012.

[4] Leman Akoglu, Hanghang Tong, and Danai Koutra. Graph-based anomaly detection and description: A survey. *DAMI*, 28(4), 2014. doi: DOI10.1007/s10618-014-0365-y.

[5] Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. Polyglot: Distributed word representations for multilingual nlp. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 183–192, Sofia, Bulgaria, August 2013. ACL.

[6] Rami Al-Rfou, Vivek Kulkarni, Bryan Perozzi, and Steven Skiena. Polyglot-ner: Massive multilingual named entity recognition. In *Proceedings of 2015 SIAM International Conference on Data Mining*, SDM '15, 2015.

[7] Noga Alon, Chen Avin, Michal Koucky, Gady Kozma, Zvi Lotker, and Mark R. Tuttle. Many random walks are faster than one. In *Proceedings of the Twentieth Annual Symposium on Parallelism in Algorithms and Architectures*, SPAA '08, pages 119–128, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-973-9. doi: 10.1145/1378533.1378557.

[8] Reid Andersen, Fan Chung, and Kevin Lang. Local graph partitioning using pagerank vectors. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 475–486. IEEE, 2006.

[9] Lucas Antiqueira, Osvaldo N Oliveira Jr., Luciano da Fontoura Costa, and Maria das Graças Volpe Nunes. A complex network approach to text summarization. *Information Sciences*, 179(5):584–599, 2009. ISSN 0020-0255. doi: http://dx.doi.org/10.1016/j.ins.2008.10.032.

[10] Valerio Arnaboldi, Marco Conti, Andrea Passarella, and Fabio Pezzoni. Analysis of ego network structure in online social networks. In *SocialCom*, 2012.

[11] Arindam Banerjee, Sugato Basu, and Srujana Merugu. Multi-way clustering on relation graphs. In *SIAM SDM*, 2007.

[12] Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. Gephi: An open source software for exploring and manipulating networks, 2009.

[13] S. Basu, I. Davidson, and K. Wagstaff. *Clustering with Constraints: Algorithms, Applications and Theory*. 2008.

[14] Yoshua Bengio and J-S Senecal. Adaptive importance sampling to accelerate training of a neural probabilistic language model. *Neural Networks, IEEE Transactions on*, 19(4): 713–722, 2008.

[15] Yoshua Bengio, Rejean Ducharme, and Pascal Vincent. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

[16] Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain. Neural probabilistic language models. In *Innovations in Machine Learning*, pages 137–186. Springer, 2006.

[17] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. 2013.

[18] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is nearest neighbor meaningful? In *Database TheoryICDT99*, pages 217–235. Springer, 1999.

[19] Chris Biemann, Stefanie Roos, and Karsten Weihe. Quantifying semantics using complex network analysis. In *Proceedings of COLING 2012*, pages 263–278, Mumbai, India, December 2012. The COLING 2012 Organizing Committee.

[20] V.D. Blondel, J.L. Guillaume, R. Lambiotte, and E. Mech. Fast unfolding of communities in large networks. *Jour. Stat. Mech.*, 2008.

[21] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

[22] Léon Bottou. Stochastic gradient learning in neural networks. In *Proceedings of Neuro-Nîmes 91*, Nimes, France, 1991. EC2.

[23] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, NY, USA, 2004. ISBN 0521833787.

[24] Ramon Ferrer i Cancho and Richard V Solé. The small world of human language. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 268(1482):2261–2265, 2001.

[25] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pages 891–900, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3794-6.

[26] Deepayan Chakrabarti, Spiros Papadimitriou, Dharmendra S. Modha, and Christos Faloutsos. Fully automatic cross-associations. In *KDD*, pages 79–88, 2004.

[27] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41(3):15, 2009.

[28] Moses Charikar. Greedy approximation algorithms for finding dense components in a graph. In *APPROX*, 2000.

[29] Yanqing Chen, Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. The expressive power of word embeddings. In *ICML 2013 Workshop on Deep Learning for Audio, Speech, and Language Processing*, Atlanta, GA, USA, July 2013.

[30] Yanqing Chen, Bryan Perozzi, and Steven Skiena. Vector-based similarity measurements for historical figures. In *Similarity Search and Applications*, pages 179–190. Springer International Publishing, 2015.

[31] A. Clauset, M.E.J. Newman, and C. Moore. Finding community structure in very large networks. *Physical Review E*, 70(6), 2004.

[32] Aaron Clauset. Finding local community structure in networks. *Physical Review E*, 72(2): 026132, 2005.

[33] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th ICML*, ICML '08, pages 160–167. ACM, 2008.

[34] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537, 2011.

[35] Anne Condon and Richard M. Karp. Algorithms for graph partitioning on the planted partition model. *Random Struct. Algorithms*, 18(2):116–140, 2001.

[36] George E Dahl, Dong Yu, Li Deng, and Alex Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *Audio, Speech, and Language Processing, IEEE Transactions on*, 20(1):30–42, 2012.

[37] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Quoc Le, Mark Mao, Marc'Aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, and Andrew Ng. Large scale distributed deep networks. In P. Bartlett, F.C.N. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1232–1240. 2012.

[38] I.S. Dhillon, S. Mallela, and D.S. Modha. Information-theoretic co-clustering. In *KDD*, 2003.

[39] Chris H. Q. Ding, Xiaofeng He, Hongyuan Zha, Ming Gu, and Horst D. Simon. A min-max cut algorithm for graph partitioning and data clustering. In *ICDM*, 2001.

[40] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159, 2011.

[41] David Eppstein, Michael S Paterson, and F Frances Yao. On nearest-neighbor graphs. *Discrete & Computational Geometry*, 17(3):263–282, 1997.

[42] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *The Journal of Machine Learning Research*, 11:625–660, 2010.

[43] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9: 1871–1874, 2008.

[44] Gary William Flake, Steve Lawrence, and C. Lee Giles. Efficient identification of web communities. In *KDD*. 2000.

[45] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010.

[46] Francois Fouss, Alain Pirotte, J-M Renders, and Marco Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *Knowledge and Data Engineering, IEEE Transactions on*, 19(3):355–369, 2007.

[47] Esther Galbrun, Aristides Gionis, and Nikolaj Tatti. Overlapping community detection in labeled graphs. *DAMI*, 28(5), 2014.

[48] Brian Gallagher and Tina Eliassi-Rad. Leveraging label-independent features for classification in sparsely labeled networks: An empirical study. In *Advances in Social Network Mining and Analysis*, pages 1–19. Springer, 2010.

[49] Brian Gallagher, Hanghang Tong, Tina Eliassi-Rad, and Christos Faloutsos. Using ghost edges for classification in sparsely labeled networks. In *Proceedings of the 14th ACM SIGKDD*, KDD '08, pages 256–264, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-193-4.

[50] Jing Gao, Haibin Cheng, and Pang-Ning Tan. Semi-supervised outlier detection. In *ACM Symp. on Appl. Comp.*, 2006.

[51] Jing Gao, Feng Liang, Wei Fan, Chi Wang, Yizhou Sun, and Jiawei Han. On community outliers and their efficient detection in information networks. In *KDD*, pages 813–822, 2010.

[52] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6): 721–741, 1984.

[53] Lise Getoor and Ben Taskar. *Introduction to statistical relational learning*. MIT press, 2007.

[54] David F. Gleich and C. Seshadhri. Vertex neighborhoods, low conductance cuts, and good seeds for local community methods. In *KDD*, pages 597–605, 2012.

[55] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. volume 27, pages 97–110, jun 2011.

[56] S. Gunnemann, I. Farber, S. Raubach, and T. Seidl. Spectral subspace clustering for graphs with feature vectors. In *ICDM*, pages 231–240, 2013.

[57] Stephan Günnemann, Ines Färber, Brigitte Boden, and Thomas Seidl. Subspace clustering meets dense subgraph mining: A synthesis of two paradigms. In *ICDM*, 2010.

[58] Manish Gupta, Arun Mallya, Subhro Roy, Jason H. D. Cho, and Jiawei Han. Local learning for mining outlier subgraphs from network datasets. In *SIAM SDM*, pages 73–81, 2014.

[59] Michael U Gutmann and Aapo Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *The Journal of Machine Learning Research*, 13(1):307–361, 2012.

[60] Daniel Hanisch, Alexander Zien, Ralf Zimmer, and Thomas Lengauer. Co-clustering of biological networks and gene expression data. In *ISMB*, pages 145–154, 2002.

[61] Keith Henderson, Brian Gallagher, Lei Li, Leman Akoglu, Tina Eliassi-Rad, Hanghang Tong, and Christos Faloutsos. It's who you know: Graph mining using recursive structural features. In *Proceedings of the 17th ACM SIGKDD*, KDD '11, pages 663–671, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0813-7.

[62] Geoffrey E Hinton. Learning distributed representations of concepts. In *Proceedings of the eighth annual conference of the cognitive science society*, pages 1–12. Amherst, MA, 1986.

[63] Geoffrey E Hinton. Learning distributed representations of concepts. In *Proceedings of the eighth annual conference of the cognitive science society*, pages 1–12. Amherst, MA, 1986.

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

[65] Robert A Hummel and Steven W Zucker. On the foundations of relaxation labeling processes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (3):267–287, 1983.

[66] P. Iglesias, E. Müller, F. Laforet, F. Keller, and K. Böhm. Statistical selection of congruent subspaces for outlier detection on attributed graphs. In *ICDM*, 2013.

[67] U Kang, Hanghang Tong, and Jimeng Sun. Fast random walk graph kernel. In *SDM*, pages 828–838, 2012.

[68] George Karypis and Vipin Kumar. Multilevel algorithms for multi-constraint graph partitioning. In *Proc. of Supercomputing*, pages 1–13, 1998.

[69] Risi Imre Kondor and John Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *ICML*, volume 2, pages 315–322, 2002.

[70] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, volume 1, page 4, 2012.

[71] Vivek Kulkarni, Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. Statistically significant detection of linguistic change. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, New York, NY, USA, 2015. ACM.

[72] Vivek Kulkarni, Bryan Perozzi, and Steven Skiena. Freshman or fresher? Quantifying the Geographic Variation of Internet Language. In *10th International AAAI Conference on Web and Social Media*, ICWSM'16, 2016.

[73] Andrea Lancichinetti, Santo Fortunato, and János Kertész. Detecting the overlapping and hierarchical community structure in complex networks. *New Journal of Physics*, 11(3): 033015, 2009.

[74] Gregory F Lawler. Intersections of random walks. probability and its applications. birkhüser boston. *Inc., Boston, MA*, 1991.

[75] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *KDD*, pages 177–187. ACM Press, 2005.

[76] Jure Leskovec, Kevin J. Lang, Anirban Dasgupta, and Michael W. Mahoney. Statistical properties of community structure in large social and information networks. In *WWW*, pages 695–704, 2008.

[77] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems*, pages 2177–2185, 2014.

[78] Nan Li, Ziyu Guan, Lijie Ren, Jian Wu, Jiawei Han, and Xifeng Yan. gIceberg: Towards iceberg analysis in large graphs. In *ICDE*, pages 1021–1032, 2013.

[79] Nan Li, Huan Sun, Kyle Chipman, Jemin George, and Xifeng Yan. A probabilistic approach to uncovering attributed graph anomalies. In *SIAM SDM*, pages 82–90, 2014.

[80] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007.

[81] F. Lin and W.W. Cohen. Semi-supervised classification of network data using very few labels. In *Advances in Social Networks Analysis and Mining (ASONAM), 2010 International Conference on*, pages 192–199, Aug 2010. doi: 10.1109/ASONAM.2010.19.

[82] Bo Long, Zhongfei Zhang, Xiaoyun Wu, and Philip S. Yu. Spectral clustering for multi-type relational data. In *ICML*, 2006.

[83] Sofus A. Macskassy and Foster Provost. A simple relational classifier. In *Proceedings of the Second Workshop on Multi-Relational Data Mining (MRDM-2003) at KDD-2003*, pages 64–76, 2003.

[84] Sofus A Macskassy and Foster Provost. Classification in networked data: A toolkit and a univariate case study. *The Journal of Machine Learning Research*, 8:935–983, 2007.

[85] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. *Intro to Information Retrieval*. Cambridge University Press, 2008.

[86] Douglas S. Massey and Nancy A. Denton. The dimensions of residential segregation. *Social Forces*, 67(2):218–315, 1988.

[87] Julian J. McAuley and Jure Leskovec. Discovering social circles in ego networks. *TKDD*, 8 (1):4, 2014.

[88] C. McCubbin, B. Perozzi, A. Levine, and A. Rahman. Finding the 'needle': Locating interesting nodes using the k-shortest paths algorithm in mapreduce. In *Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on*, pages 180–187, Dec 2011.

[89] Mary McGlohon, Leman Akoglu, and Christos Faloutsos. Weighted graphs and disconnected components: patterns and a generator. In *KDD*, pages 524–532, 2008.

[90] Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, pages 415–444, 2001.

[91] Tomas Mikolov, Stefan Kombrink, Lukas Burget, JH Cernocky, and Sanjeev Khudanpur. Extensions of recurrent neural network language model. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5528–5531. IEEE, 2011.

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

[93] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119. 2013.

[94] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of NAACL-HLT*, pages 746–751, 2013.

[95] Andriy Mnih and Geoffrey E Hinton. A scalable hierarchical distributed language model. *Advances in neural information processing systems*, 21:1081–1088, 2009.

[96] J Moody. Race, school integration, and friendship segregation in america. *American J. of Sociology*, 107(3):679–716, 2001.

[97] Frederic Morin and Yoshua Bengio. Hierarchical probabilistic neural network language model. In *Proceedings of the international workshop on artificial intelligence and statistics*, pages 246–252, 2005.

[98] Flavia Moser, Recep Colak, Arash Rafiey, and Martin Ester. Mining cohesive patterns from graphs with feature vectors. In *SDM*, 2009.

[99] Adilson E. Motter, Alessandro P. S. de Moura, Ying-Cheng Lai, and Partha Dasgupta. Topology of the conceptual network of language. *Phys. Rev. E*, 65:065102, Jun 2002. doi: 10.1103/PhysRevE.65.065102.

[100] Emmanuel Müller, Stephan Günnemann, Ira Assent, and Thomas Seidl. Evaluating clustering in subspace projections of high dimensional data. *VLDB*, 2(1):1270–1281, 2009.

[101] Emmanuel Müller, Patricia Iglesias Sanchez, Yvonne Mülle, and Klemens Böhm. Ranking outlier nodes in subspaces of attributed graphs. In *ICDE Workshops*, pages 216–222. IEEE Computer Society, 2013.

[102] Jennifer Neville and David Jensen. Iterative classification in relational data. In *Proc. AAAI-2000 Workshop on Learning Statistical Models from Relational Data*, pages 13–20, 2000.

[103] Jennifer Neville and David Jensen. Leveraging relational autocorrelation with latent group models. In *Proceedings of the 4th International Workshop on Multi-relational Mining*, MRDM '05, pages 49–55, New York, NY, USA, 2005. ACM. ISBN 1-59593-212-7.

[104] Jennifer Neville and David Jensen. A bias/variance decomposition for models using collective inference. *Machine Learning*, 73(1):87–106, 2008.

[105] M. E. J. Newman. Assortative mixing in networks. *Physical Review Letters*, 89(20), 2002.

[106] M. E. J. Newman. Modularity and community structure in networks. *PNAS*, 103(23): 8577–8582, 2006.

[107] M. E. J. Newman. *Networks: An Introduction*. Oxford University Press, Oxford; New York, 2010.

[108] M.E.J. Newman and M. Girvan. Mixing patterns and community structure in networks. In *Statistical Mechanics of Complex Networks*, volume 625, pages 66–87. 2003.

[109] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, 2001.

[110] Caleb C. Noble and Diane J. Cook. Graph-based anomaly detection. In *KDD*, pages 631–636. ACM, 2003.

[111] Bryan Perozzi and Leman Akoglu. Scalable anomaly ranking of attributed neighborhoods. In *Proceedings of 2016 SIAM International Conference on Data Mining*, SDM '16, 2016.

[112] Bryan Perozzi and Steven Skiena. Exact age prediction in social networks. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, 2015.

[113] Bryan Perozzi, Christopher McCubbin, Spencer Beecher, and J.T. Halbert. Scalable graph clustering with pregel. In *Complex Networks IV*, volume 476 of *Studies in Computational Intelligence*, pages 133–144. 2013.

[114] Bryan Perozzi, Leman Akoglu, Patricia Iglesias Sánchez, and Emmanuel Müller. Focused clustering and outlier detection in large attributed graphs. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 1346–1355, New York, NY, USA, 2014. ACM.

[115] Bryan Perozzi, Rami Al-Rfou, Vivek Kulkarni, and Steven Skiena. Inducing language networks from continuous space word representations. In *Complex Networks V*, volume 549 of *Studies in Computational Intelligence*, pages 261–273. 2014.

[116] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 701–710, New York, NY, USA, 2014. ACM.

[117] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, New York, NY, USA, August 2014. ACM.

[118] Bryan Perozzi, Christopher McCubbin, and J.T. Halbert. Scalable graph clustering with parallel approximate pagerank. *Social Network Analysis and Mining*, 4(1):179, 2014. ISSN 1869-5450.

[119] Bryan Perozzi, Vivek Kulkarni, and Steven Skiena. Walklets: Multiscale graph embeddings for interpretable network classification. In *(submitted)*, 2016.

[120] Bryan Perozzi, Michael Schueppert, Jack Saalweachter, and Mayur Thakur. When recommendation goes wrong - Anomalous Link Discovery in Recommendation Networks. In *Proceedings of the 22th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, 2016.

[121] Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems 24*, pages 693–701. 2011.

[122] Martin Rosvall and Carl T Bergstrom. Multilevel compression of random walks on networks reveals hierarchical organization in large integrated systems. *PloS one*, 6(4):e18209, 2011.

[123] DE Rumelhart, GE Hinton, and RJ Williams. Learning internal representation by back propagation. *Parallel distributed processing: exploration in the microstructure of cognition*, 1, 1986.

[124] Holger Schwenk, Anthony Rousseau, and Mohammed Attik. Large, pruned or continuous space language models on a gpu for statistical machine translation. In *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, pages 11–19. Association for Computational Linguistics, 2012.

[125] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93, 2008.

[126] Hiroaki Shiokawa, Yasuhiro Fujiwara, and Makoto Onizuka. Fast algorithm for modularity-based graph clustering. In *AAAI*, 2013.

[127] Mariano Sigman and Guillermo A Cecchi. Global organization of the wordnet lexicon. *Proceedings of the National Academy of Sciences*, 99(3):1742–1747, 2002.

[128] Arlei Silva, Wagner Meira Jr., and Mohammed J. Zaki. Mining attribute-structure correlated patterns in large attributed graphs. *PVLDB*, 5(5):466–477, 2012.

[129] Daniel A Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 81–90. ACM, 2004.

[130] J. Tang and H. Liu. Unsupervised feature selection for linked social media data. In *KDD*, pages 904–912, 2012.

[131] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 1067–1077, Republic and Canton of Geneva, Switzerland, 2015. International World Wide Web Conferences Steering Committee. ISBN 978-1-4503-3469-3.

[132] Lei Tang and Huan Liu. Relational learning via latent social dimensions. In *Proceedings of the 15th ACM SIGKDD*, KDD '09, pages 817–826, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-495-9.

[133] Lei Tang and Huan Liu. Scalable learning of collective behavior based on sparse social dimensions. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 1107–1116. ACM, 2009.

[134] Lei Tang and Huan Liu. Leveraging social media networks for classification. *Data Mining and Knowledge Discovery*, 23(3):447–478, 2011. ISSN 1384-5810.

[135] Yingtao Tian, Vivek Kulkarni, Bryan Perozzi, and Steven Skiena. On the convergent properties of word embedding methods. In *(submitted)*, 2016.

[136] Yingtao Tian, Bryan Perozzi, Vivek Kulkarni, and Steven Skiena. Piecewise linear embedding alignment for multilingual learning. In *(submitted)*, 2016.

[137] Yuanyuan Tian, Richard A. Hankins, and Jignesh M. Patel. Efficient aggregation for graph summarization. In *SIGMOD*, 2008.

[138] Hanghang Tong and Ching-Yung Lin. Non-negative residual matrix factorization with application to graph anomaly detection. In *SIAM SDM*, pages 143–153, 2011.

[139] Charalampos E. Tsourakakis, Francesco Bonchi, Aristides Gionis, Francesco Gullo, and Maria A. Tsiarli. Denser than the densest subgraph: extracting optimal quasi-cliques with quality guarantees. In *KDD*, 2013.

[140] Johan Ugander, Brian Karrer, Lars Backstrom, and Cameron Marlow. The anatomy of the facebook social graph. *arXiv preprint arXiv:1111.4503*, 2011.

[141] Jean Véronis. HyperLex: lexical cartography for information retrieval. *Computer Speech & Language*, 18(3):223–252, 2004. ISSN 0885-2308. doi: http://dx.doi.org/10.1016/j.csl.2004.05.002.

[142] SVN Vishwanathan, Nicol N Schraudolph, Risi Kondor, and Karsten M Borgwardt. Graph kernels. *The Journal of Machine Learning Research*, 99:1201–1242, 2010.

[143] Fei Wang and Jimeng Sun. Distance metric learning in data mining. In *SIAM SDM*. Tutorials, 2012.

[144] Xi Wang and Gita Sukthankar. Multi-label relational neighbor classification using social context features. In *Proceedings of the 19th ACM SIGKDD*, pages 464–472. ACM, 2013.

[145] Joyce Whang, David Gleich, and Inderjit Dhillon. Overlapping community detection using seed set expansion. In *CIKM*, pages 2099–2108, 2013.

[146] Scott White and Padhraic Smyth. A spectral clustering approach to finding communities in graph. In *SDM*, 2005.

[147] UK WR143PS. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. 1989.

[148] Eric P. Xing, Andrew Y. Ng, Michael I. Jordan, and Stuart J. Russell. Distance metric learning with application to clustering with side-information. In *NIPS*, pages 505–512, 2002.

[149] Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1365–1374. ACM, 2015.

[150] Cheng Yang and Zhiyuan Liu. Comprehend deepwalk as matrix factorization. *arXiv preprint arXiv:1501.00358*, 2015.

[151] Jaewon Yang and Jure Leskovec. Defining and evaluating network communities based on ground-truth. In *ICDM*, 2012.

[152] Jaewon Yang and Jure Leskovec. Overlapping community detection at scale: a nonnegative matrix factorization approach. In *WSDM*, pages 587–596, 2013.

[153] W Zachary. An information flow model for conflict and fission in small groups1. *Journal of anthropological research*, 33(4):452–473, 1977.

[154] Yang Zhou, Hong Cheng, and Jeffrey Xu Yu. Graph clustering based on structural/attribute similarities. *PVLDB*, 2(1):718–729, 2009.