UNIVERSIDADE FEDERAL DE MINAS GERAIS
Instituto de Ciências Exatas
Programa de Pós-Graduação em Ciência da Computação

Filipe Barreto do Nascimento

On the impact of sample reduction strategies for heterogeneous network representation learning

Belo Horizonte
2021

Filipe Barreto do Nascimento

On the impact of sample reduction strategies for heterogeneous network representation learning

Final version

Thesis presented to the Graduate Program in Computer Science of the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Master in Computer Science.

Advisor: Rodrygo Luis Teodoro Santos

Belo Horizonte
2021

Nascimento, Filipe Barreto do.

N224o On the impact of sample reduction strategies for heterogeneous network representation learning [manuscrito] / Filipe Barreto do Nascimento. — 2021.
76 f. il.

Orientador: Rodrygo Luis Teodoro Santos

Dissertação (mestrado) — Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciência da Computação
Referências: f. 67-72

1. Computação – Teses. 2. Aprendizado de representações – Teses. 3. Redes heterogêneas – Teses. 4. Passeio aleatório (Matemática) – Teses. I. Santos, Rodrygo Luis Teodoro. II. Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciência da Computação. III. Título.

CDU 519.6*82.(043)

Ficha catalográfica elaborada pela bibliotecária Belkiz Inez Rezende Costa CRB 6/1510 Universidade Federal de Minas Gerais - ICEx

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

On the Impact of Sample Reduction Strategies for Heterogeneous Network Representation Learning

FILIPE BARRETO DO NASCIMENTO

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. RODRYGO LUIS TEODORO SANTOS - Orientador
Departamento de Ciência da Computação - UFMG

PROF. ADRIANO ALONSO VELOSO
Departamento de Ciência da Computação - UFMG

PROF. FABRÍCIO MURAI FERREIRA
Departamento de Ciência da Computação - UFMG

PROF. LEANDRO BALBY MARINHO
Departamento de Sistemas e Computação - UFCG

Belo Horizonte, 29 de Abril de 2021.

Acknowledgements

First and foremost, I would like to thank my family, in particular my parents Simone and Emídio and my brother Thiago, for their unconditional love and ever-present support. Secondly, I am extremely grateful to my girlfriend Yasmin, for standing by me, even when we were states apart, and being my safe haven, in good and bad times, over the course of this work.

I am deeply indebted to my advisor, Rodrygo Santos, for believing in my work, encouraging and guiding me in difficult moments and sharing his knowledge and ideas, which were essential to the development of this dissertation. Every class, discussion, feedback session and presentation we had during this time was instrumental in helping me grow as a scientist.

Next, I would like to thank Leila Silva, from Universidade Federal de Sergipe, who played a decisive role in my career by encouraging me to apply to the MSc program in the first place, at a moment when I thought I had no chance of getting accepted.

Last but not least, I would like to express my sincere gratitude to my friends from LATIN and DCC, whose warm reception and continued support throughout the years have made me feel at home in Belo Horizonte from the very beginning.

"It is the business of the future to be dangerous; and it is among the merits of science that it equips the future for its duties."
(Alfred North Whitehead)

Resumo

Modelos de aprendizado de representações em redes mapeiam vértices de um grafo para vetores em um espaço de baixa dimensionalidade, que por sua vez são utilizados em diversas tarefas de aprendizado de máquina, como classificação de vértices, agrupamento (*clustering*) e visualização de dados. Tendo em vista o aumento na disponibilidade de dados em larga escala sobre redes e grafos, tais técnicas devem ser capazes de lidar com conjuntos de dados cada vez maiores e ainda assim garantir a obtenção de resultados competitivos. Adicionalmente, a maioria dos grafos que modelam sistemas reais contêm informações adicionais sobre os tipos de vértices e arestas do grafo, fazendo com que esses sejam normalmente modelados como redes de informação heterogêneas. Assim, consideramos abordagens escaláveis de aprendizado de representações em redes heterogêneas, em particular aquelas baseadas em passeios aleatórios, que amostram sequências de vértices no grafo e as utilizam como entrada pra algoritmos de aprendizado de máquina. Nesta dissertação, propomos duas estratégias para reduzir a quantidade de amostras de treino utilizadas como entrada para os algoritmos, com o intuito de treinar os modelos baseados em passeios aleatórios mais rapidamente: (1) passeios baseados em centralidade, que levam em consideração a informação estrutural de centralidade associada aos nós do grafo e (2) passeios focados, que concentram sua atenção em tipos específicos de vértices a depender da tarefa em questão. Nossas descobertas apontam que ambas as estratégias contribuem para a redução no conjunto de amostras de treino necessárias e sugerem a presença de dados redundantes nos processos de amostragem tradicionais referentes a esses modelos. Experimentos em três conjuntos de dados de sistemas reais demonstram que nossas abordagens propostas são capazes de manter e, ocasionalmente, superar resultados obtidos por modelos já estabelecidos na literatura, validando assim sua adoção como novas ferramentas no projeto de algoritmos escaláveis de aprendizado de representações em redes.

Palavras-chave: Aprendizado de Representações. Redes Heterogêneas. Passeios Aleatórios.

Abstract

Network embedding models map nodes of a graph to vectors in a low dimensional space, which in turn are used for multiple machine learning tasks, such as node classification, clustering, and visualization. With the growth of available large-scale network data, such techniques must be able to deal with increasingly bigger data sets, while still maintaining competitive results. In addition, most real-world graphs contain additional information regarding node and edge types, leading those to be often modeled as heterogeneous information networks. Thus, we focus on scalable heterogeneous network embedding approaches, in particular random walk-based techniques, which sample sequences of nodes on the graph and use these as input to machine learning algorithms. In this dissertation, we propose two strategies to reduce the amount of input training data sampled from the graph, so as to achieve faster training times when using random walk-based methods: (1) centrality-based walks, which take into account structural centrality information of nodes in the graph, and (2) focused walks, which concentrate on specific node types depending on the task at hand. Our findings show that both strategies help alleviate the sample input size and suggest the presence of redundant data in traditional sampling processes for these models. Experimental results on three real-world graph data sets show that our proposed strategies are able to maintain and sometimes improve results on established network embedding methods using a reduced amount of training samples, validating their use as a novel tool in the design of scalable network embedding algorithms.

Palavras-chave: Representation learning. Heterogeneous Networks. Random Walks.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

1.1 Motivation

Real-world networks, such as social and biological networks, are typically modeled as graphs and, depending on the information captured by their nodes and edges, are either classified as homogeneous (one type of node and edge) or heterogeneous (more than one type of node and/or edge) Networks [Shi et al., 2016]. These powerful structures are able to capture arbitrary types of relations between entities, providing researchers with a valuable tool in the study of complex networks [Barabási et al., 2016].

Over the past years, different types of approaches have been proposed in order to try and capture the information contained in such networks. Due to the large scale of real networks, it is often prohibitively expensive to work with traditional graph representations and algorithms. In an attempt to solve this limitation, network representation learning methods embed nodes in a low-dimensional vector representation. The resulting latent representations manage to capture network information, while also alleviating the laborious task of feature engineering [Hamilton et al., 2017b]. Network embeddings can be used in end-to-end applications and also as input to other machine learning algorithms.

Most network representation learning techniques map nodes into a vector space such that structural properties of nodes in the original graph are preserved. This is done using varied techniques – each with its strengths and weaknesses. In this work, we base our choices by looking at these techniques/models from a time complexity perspective. We observe that matrix factorization scale quadratically with regard to the number of nodes in the network $|V|$ and linearly with regard to the dimension of the output representation d (i.e. at least $\mathcal{O}(d|V|^2)$) [Zhang et al., 2018], whereas Graph Neural Networks, such as Graph Convolutional Networks are typically bound by $\mathcal{O}(L|E|d + L|V|d^2)$, where L is the number of layers in the network, $|V|$ and $|E|$ are the total number of nodes and edges, respectively and d is the dimension of the output representations [Chiang et al., 2019; Wu et al., 2020]. This hinders the scalability of the aforementioned methods.

Due to the high computational cost of the previous approaches, we choose to focus

on another well established family of methods that makes use of node pairs extracted from random walks on the network as input samples to a language model, as first proposed in [Perozzi et al., 2014]. This approach, when used with negative sampling [Mikolov et al., 2013b] in the loss function, takes $\mathcal{O}(d|V|)$ time [Zhang et al., 2018; Cai et al., 2018]. In addition to the reduced time complexity, we choose to work with this unsupervised random walk approach due to it being (1) scalable: the walks are easily parallelizable and one does not need to resort to operations on costly structures, such as the adjacency matrix or the graph Laplacian [Chung and Graham, 1997] and (2) versatile: the learning process is such that the output representations may be used in different kinds of tasks and scenarios. We also limit our attention to heterogeneous networks, since those are a generalization of homogeneous information networks and are able to, whenever available, capture additional information, which is often the case on real-world applications.

When it comes to heterogeneous networks, it is often the case that some node types are much more frequent than others (e.g. authors and papers in comparison to publication venues), thus having higher probability of being sampled in random walks on the graph [Sun et al., 2011]. Traditional network embedding methods do not consider node types in their walks, which make the node visit distributions biased toward highly frequent node types, causing some types to "dominate" the learning process and limiting the focus on important relations between less visible nodes.

To overcome the aforementioned issue and capture the underlying information in a heterogeneous network, current embedding techniques either resort to sampling training paths by following hand-engineered meta-paths [Dong et al., 2017; Shang et al., 2016] defined over heterogeneous node types or designing random walk strategies that account for node types [Hussein et al., 2018]. We limit our attention to the latter, since we are interested in scalable methods and finding the right meta-path proves to be a complicated task due to the number of possible meta-paths growing exponentially even for a small number of node types [Huang and Mamoulis, 2017; Hussein et al., 2018], while also requiring either expert domain knowledge or some sort of predefined algorithmic strategy, contributing to the complexity of the problem.

Even though successful embedding algorithms were proposed for both homogeneous and heterogeneous networks in recent years, we note that, for most techniques, nodes are treated equally during the random walks. In contrast, the graph theory [Bondy et al., 1976] and network science [Barabási et al., 2016] literature is often interested in developing *centrality measures*, that is, coming up with ways to quantitatively assess the relative importance of a node in a network. In light of this disparity, we posit that making use of network centrality information – which is readily available and commonly used in other fields of study pertaining to networks – to guide the sampling step may allow us to reduce the number of training samples while maintaining efficacy. In addition, we hypothesize that we can take advantage of the heterogeneous structure of a network and

further reduce the number of required training samples. Thus, in this dissertation, we propose and evaluate the impact of two complimentary sample reduction strategies on random walk-based models for heterogeneous network embedding.

1.2 Dissertation Statement

Inspired by the concepts of preferential attachment [Barabási and Albert, 1999] and node entropy [Small, 2013], we posit that since central nodes present higher probability of attracting new nodes in complex networks and higher associated uncertainty (node entropy), they carry more relevant information about the network and this should be reflected in the learning process.

Additionally, we note that, while it is useful to learn embeddings for all types of nodes in a heterogeneous network, it is often the case that we are only interested in the output embeddings of a subset of nodes (e.g. users and items on a recommendation network). Therefore, we argue that, depending on the task for which the embeddings will be used, task-agnostic heterogeneous network embedding approaches may benefit from *focusing* on learning embeddings for a subset of node types, as we are able to make use of the heterogeneous structure to learn good embeddings while relieving the restriction of generating embeddings for every single node in the graph.

Thus, the statement of this dissertation is that random walk-based heterogeneous network embedding techniques may benefit from the use of sample reduction techniques. In particular, by making use of centrality information and taking advantage of the heterogeneous structure of the network, the models will require less samples to learn representations that perform just as well/better than their "naive" counterparts in terms of effectiveness.

This statement leads us to raise the following research questions, which help guide our work and are answered in the following chapters:

RQ1. To what extent can we improve efficiency without harming effectiveness?

RQ2. How do different random walk strategies impact our approach?

RQ3. How do different node properties impact our approach?

RQ4. Are traditional sampling approaches generating redundant training data?

1.3 Dissertation Contributions

In this work, we investigate the aforementioned hypotheses and their applications on heterogeneous network embeddings. As a result, the key contributions of this dissertation are:

1. Extend the idea proposed in [Gao et al., 2018] to generalize centrality-based walks for networks of any type and degree of heterogeneity. This leads to random walks that are able to consistently reduce sample size and, consequently, training time, while maintaining competitive results;

2. Propose focused walks, a task-based walk strategy for heterogeneous networks which helps reduce even further the training sample size;

3. Analyze the impact of the sample reduction approaches under different types of networks and random walk strategies, validating our hypotheses via extensive experimentation and analyses.

1.4 Dissertation Overview

The remainder of this dissertation is organized as follows:

- **Chapter 2** presents the base concepts required to work with network embeddings and random walks, along with real world applications for these techniques. We also overview and discuss relevant research work on random walk-based network embedding models;

- **Chapter 3** introduces and formalizes our two proposed techniques for efficient sample input size reduction, namely centrality-based and focused walks. We then frame our contributions in the context of a learning algorithm with a customizable random walk module in which our strategies (along with others) can be added or removed with ease;

- **Chapter 4** describes our experimental setup in terms of the datasets, tasks, metrics, baselines and train/test procedures used in obtaining our results;

- As a complement, **Chapter 5** presents our results and discusses our findings in terms of each walk strategy both individually and in conjunction. We also provide a qualitative view of our results by means of a node visualization task;
- Finally, **Chapter 6** concludes our work by summarizing our findings and contributions. It also discusses possibilities and directions for future research.

Chapter 2

Background and Related Work

This chapter presents the necessary background knowledge for the concepts discussed in this work in Section 2.1, then proceeds to illustrate common applications of network embeddings in Section 2.2. Finally, we review relevant work on random walk-based network embedding models as well as models that make use of centrality measures in Section 2.3.

2.1 Definitions

In this section, we provide definitions for concepts that are used throughout the dissertation. We also formalize the problem statement. First, we start by formalizing our notion of a network. Then we move on to define random walks on graphs along with some important properties and results.

Due to the way random walks are used in the context of network embedding techniques, we will limit our attention to simple graphs, that is, undirected graphs containing neither loops nor multiple edges between pairs of nodes. As such, the definitions that follow focus on this type of graph in particular.

Definition 1 (*Network*)**.** A network is a graph $G = (V, E, \phi, \psi)$, where V is the set of vertices (or nodes) and $E \subseteq V \times V$ is the set of edges (directed or undirected). In addition, there exist node and edge type mapping functions $\phi : V \rightarrow A$ and $\psi : E \rightarrow R$, where A and R are node type and edge type sets, respectively. If $|A| + |R| > 2$, we denote it by Heterogeneous Network, or else, by Homogeneous Network.

Definition 2 (*Neighborhood of a node*)**.** We denote by $N(v_i) = \{v_j \,|\, (v_i, v_j) \in E\}$ the neighborhood of node v_i. Intuitively, it is the set of its immediate neighbors or nodes whose edges connect with v_i.

Definition 3 (*Random Walk*)**.** A random walk [Lovász et al., 1993] on a graph G is a sequence of random nodes $(v_i : i = 0, 1, ...)$, starting at a node $v_0 \in V$, sampled from a initial distribution P_0 (note that we can assign probability 1 to some specific node so as

to fix it as a starting state). Considering that we are currently visiting node v_i at the i-th step, we proceed to move to one of its neighbors $v_{i+1} \in N(v_i)$ at step $i+1$ with probability $p(v_{i+1}|v_i)$.

Definition 4 (*Node degree*)**.** The degree $deg(v_i)$ of a node v_i is the number of its neighbors, that is, $deg(v_i) = |N(v_i)|$.

Since we aim to work with heterogeneous networks, it is useful to expand the concept of neighborhood in order to take node types into account. Therefore, we introduce the concepts of Typed Neighborhood and Typed Degree [Rossi et al., 2019]. We also define typed node subsets of a heterogeneous network.

Definition 5 (*Typed Neighborhood*)**.** Given a node $v_i \in V$, its typed neighborhood $N^t(v_i)$ is the set of nodes in its neighborhood with type t. Formally, we have $N^t(v_i) = \{v_j \,|\, v_j \in N(v_i), \phi(v_j) = t\}$.

Note that, for all $t \in A$, we have $N^t(v) \subseteq N(v)$. Also, we can define the homogeneous neighborhood of a node as $N^{\phi(v)}(v)$ and its heterogeneous neighborhood as $N(v) - N^{\phi(v)}(v)$.

Definition 6 (*Typed Degree*)**.** The typed degree $deg^t(v_i)$ of a node $v_i \in V$ is given by the number of nodes in its t-typed neighborhood, that is, $deg^t(v_i) = |N^t(v_i)|$.

Definition 7 (*Typed Node Subset*)**.** The typed node subset $V^t \subseteq V$ of a graph G is given by the set of t-typed nodes in G, that is, $V^t = \{v_i \in V \,|\, \phi(v_i) = t\}$.

It is worth noting that a random walk is a finite, time-reversible Markov chain [Norris, 1998]. Therefore, we can define the matrix $M_{ij} : v_i, v_j \in V$ of transition probabilities, with $M_{ij} = p(v_j|v_i)$, if $(v_i, v_j) \in E$ and $p_{ij} = 0$, otherwise. The choice of transition probabilities is a crucial component both in this work and also in the design of random walks in general. As such, in Section 3.1.2, we detail different strategies in the context of the random walk module, which is a component of our learning framework.

We can also consider more complex random walks by modeling them as higher order Markov chains [Li and Zhang, 2015].

Definition 8 (*Higher order Markov chain*)**.** A n-th order Markov chain defines its transition probabilities to the next state/node based on the sequence of the n previously visited states. Formally, we have that n-th order Markov chain must adhere to the following property: at step $i > n$, for all $v_i \in V$, $p(v_i|v_{i-1}, v_{i-2}, ..., v_1) = p(v_i|v_{i-1}, v_{i-2}, ..., v_{i-n})$. Note that when $n = 1$, we have the simple random walk defined above.

2.1.1 Problem Formulation

Our problem consists of learning low dimensional representations of nodes in a network. Given a network $G = (V, E, \phi, \psi)$, we would like to learn a mapping function $f : V \rightarrow \mathbb{R}^d$ from the set of nodes to d-dimensional latent representations. In particular, we would like the output representations to preserve important information (e.g. topology and node content) about the network, such that, as per some similarity definition, similar nodes should be embedded closely in the output space.

2.2 Tasks for Network Embedding

The output embeddings are commonly used as features that capture the information about a node in a network. These features aid us in solving problems in the fields of network science, recommender systems or machine learning, for instance. We are often interested in inferring information, visualizing, and analyzing different kinds of networks. In this section, we introduce three common tasks in which node embeddings are widely used. We choose to focus on these tasks due to their being the most commonly used evaluation methods in the field of unsupervised network embeddings [Zhang et al., 2018].

2.2.1 (Multi-Label) Node Classification

Node classification is a central task in the study of networks. Nodes are often associated with many types of labeled information, such as areas of activity in academic networks (e.g. a *machine learning* researcher) or video tags/categories in a streaming network. As new nodes are added to the graph, the information might not be readily available (e.g. a user might not have filled optional fields that would provide label information) and we would like to infer the label(s) of those nodes in order to improve the quality of our applications.

Thus, in this context, we have the node classification problem formulated as: given a network G with a subset of labeled nodes $L \subset V$, we would like to predict the correct/best possible label for the set $V - L$ of unlabeled nodes. In particular, if our set

of labels consists of two classes, we typically denote the task as binary classification. Conversely, if there are more than two classes, we have a multiclass classification problem. The multi-label classification problem is a variant of the more conventional binary and multiclass classification problems: in this case, our data instances can have zero to multiple associated labels.

In terms of related work, many models make use of the node classification task as a way to assess the quality of the learned representations, such as [Tang and Liu, 2009a,b; Perozzi et al., 2014; Tang et al., 2015; Kipf and Welling, 2016; Grover and Leskovec, 2016; Hamilton et al., 2017a; Dong et al., 2017; Huang and Mamoulis, 2017; Fu et al., 2017; Hussein et al., 2018].

2.2.2 Node Clustering

In some cases we do not have access to label information, which leads us to a different, but not less important, problem: how to assign our nodes to clusters such that similar nodes (based on some sense of similarity) are grouped together? This is a common unsupervised learning task denoted by clustering. When clustering nodes, we are often interested in detecting underlying patterns/connections in our nodes, such as potential extremist users and associated political echo chambers in social networks, or yet undiscovered scientific collaborators in an academic network, for instance.

While the notion of a cluster can be hard to define in quantitative terms, we usually consider a grouping of data objects for which we have labels, but we omit the label information during the entire evaluation procedure, only using such information to evaluate the quality of the output clusters. When it comes to network embeddings, node clustering is another rather common task used to evaluate embedding quality, as seen on [Sun et al., 2011; Dong et al., 2017; Huang and Mamoulis, 2017; Hussein et al., 2018], to name a few.

2.2.3 Node Visualization

Node visualization is an important task which aims to project nodes of a network into a two dimensional space for plotting purposes. It is often used to qualitatively assess the embedding capabilities of models by assuming that better visualizations (e.g. the ones

that manage to separate the nodes per label) mean better discriminative power associated with a particular model [Tang et al., 2015; Huang and Mamoulis, 2017].

Visualizations can also help in analyzing intended properties of models, such as seen in [Grover and Leskovec, 2016], where the authors color different clusterings based on different parameters to illustrate that their algorithm captured desired properties (homophily and structural equivalence), or in [Dong et al., 2017], in which a visualization of embeddings of CS venues depicts the effectiveness of the model in grouping conferences of similar topics together.

2.3 Related Work

According to Zhang et al. [2018], approaches can be categorized from an algorithmic perspective in terms of the following techniques: matrix factorization, random walk, edge modeling, deep learning and hybrid methods. Our approach in this work builds on top of homogeneous and heterogeneous random walk methods, of which we provide an overview in this section. For additional information on other network embedding techniques, we invite the reader to refer to the works by Battaglia et al. [2018], Cai et al. [2018], Cui et al. [2018], Hamilton et al. [2017b], Zhang et al. [2018] and [Wu et al., 2020], each of which does an excellent job in surveying the recent progresses in the field. We also present related work on network embedding techniques that make use of centrality information as some form of input to their learning algorithms.

2.3.1 Random Walk-based Network Embedding

Based on the idea that the distribution of vertices in short random walks follows a power-law, much like the distribution of words in natural language, Perozzi et al. [2014] proposed DeepWalk, the first random walk-based model. Influenced by the success of the Skip-Gram [Mikolov et al., 2013a,b] model in natural language processing tasks, their seminal work introduced the analogy of modeling node sequences sampled via unbiased random walks as sentences and using them as input to Skip-Gram, essentially treating individual nodes as words in a language model. In Figure 2.1, we describe the general steps of this learning procedure for heterogeneous networks.

Expanding on this idea, Grover and Leskovec [2016] introduced parameters to

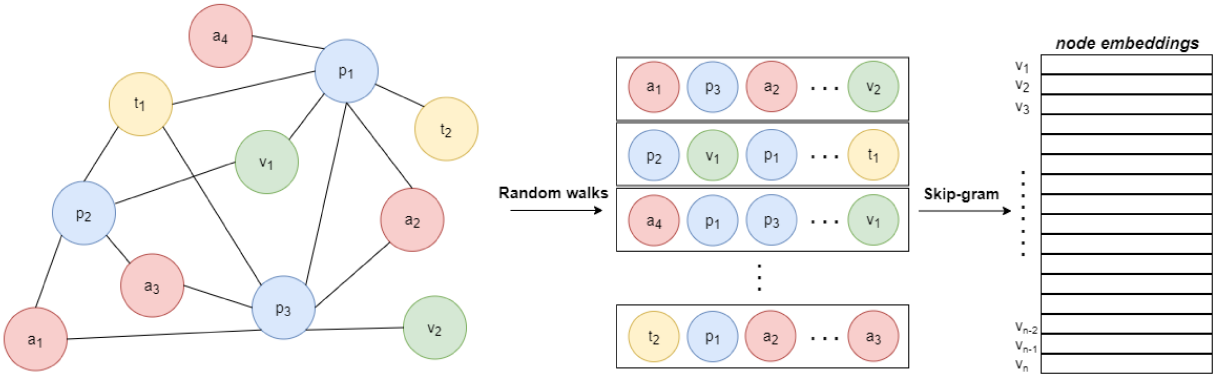


Figure 2.1: Random walk-based network representation learning. Given a heterogeneous network (here we denote different types of nodes by colors and letters), we generate random walks that are in turn used as input to a Skip-gram learning algorithm. In particular, we focus our work on the design of strategies to customize the random walk step.

control the node sampling strategies of the random walk. Their custom sampling strategy can be modeled to behave as a mixture of two classic sampling strategies: Breadth-first and Depth-first sampling. They argue that prediction tasks on nodes are often based on two types of similarities: (1) structural equivalence, implying that nodes sharing similar structural roles should be embedded with high similarity; (2) homophily, which considers nodes that are part of similar clusters to be highly similar. Their findings show that there is no single winning sampling strategy for every network embedding task.

When dealing with heterogeneous networks, Sun et al. [2011] showed that random walks tend to concentrate on highly visible types of nodes, thus failing to properly capture the heterogeneous information in the network. As a solution to this issue, Dong et al. [2017] propose meta-path based random walks. Formally, given a predefined sequence of node types – a meta-path scheme – the walk transition probabilities are defined such that at each step the walker must adhere to the node type specified in the schema, such that a particular type of semantic is captured (e.g. a meta-path **A**-**P**-**V**-**P**-**A** indicates a path between **A**uthors that published **P**apers in the same **V**enue). In addition, the authors also propose heterogeneous variants of the Skip-gram loss function and negative sampling strategy. The use of meta-paths has inspired many other works [Fu et al., 2017; Huang and Mamoulis, 2017; Shi et al., 2019; Chen and Sun, 2017; Shang et al., 2016] and remains the most adopted strategy when tackling heterogeneous networks.

In contrast, Hussein et al. [2018] argue that the finding optimal meta-paths is often difficult in practice and, since model performance is strongly dependent on the chosen meta-path(s), propose JUST (JUmp and STay walks), an alternative heterogeneous network embedding model that does not make use of meta-paths. Instead, they design a random walk that, at each step, must decide whether to *jump* to a node of different type or *stay* in the same domain and walk to a node of the same type as the current one.

They model these probabilities using exponential decay paired with an arbitrarily-sized memory queue for previously visited node types. This probabilistic modeling attempts to achieve the balanced node type distribution obtained when using meta-paths.

2.3.2 Centrality Measures for Network Representation Learning

LOG [Ma et al., 2017] leverages PageRank [Page et al., 1999] when trying to model global information, optimizing it along with a Skip-gram model and PRUNE [Lai et al., 2017] attempts to preserve global ranking by optimizing a PageRank related objective function defined in the context of a siamese neural network.

In addition, BiNE [Gao et al., 2018] is a model specifically tailored for bipartite networks. It proposes a multi-objective framework in which three objectives are optimized together. Objective O_1 models the explicit relations (first order proximity, as proposed in [Tang et al., 2015]) by minimizing the Kullback-Leibler divergence between the empirical distribution and the reconstruction distribution via parameters. Each of the other two objectives O_2 and O_3 are related to the two partitions in the graph. For each partition, they induce a homogeneous subgraph and perform biased random walks based on centrality measures to be used with the Skip-gram model sharing the same parameters with O_1.

More recently, the GraphCSC (Graph Convolutions with Structure and Centrality information) [Chen et al., 2019] framework introduced Centrality Defined Graph Convolutions, a neighborhood aggregation mechanism based on a ranking generated by centrality measures, extending the concept proposed in [Ying et al., 2018]. In addition, it uses centrality-based random walks – as proposed in [Gao et al., 2018] – and centrality-based negative sampling, paired with an additional pairwise ranking objective to learn their embeddings.

While the aforementioned models make use of centrality information in their learning pipeline, all of them are proposed within a multi-objective optimization framework, along with additional mechanisms (e.g. attention, convolutions, first-order proximity, etc.), which dilute the impact of centrality in the learning process. As such, it is difficult to pinpoint exactly what good does it do when it comes to network embedding models.

In an attempt to better understand and characterize our results, we choose to focus solely on the interaction between random walk models and centrality-based walks. By testing the walks with different models, we aim to eliminate any possible confounding factors that might cloud our results.

2.4 Summary

In this chapter, we provided necessary mathematical definitions pertaining to the concepts discussed in this work, as well as a brief presentation on common tasks in the field. We have also reviewed related work on random walk-based network embeddings and pointed to some works where centrality measures are used together with network representation learning. In particular, in Section 2.1, we presented definitions regarding graphs/networks, random walks and Markov chains, while also formally defining the node embedding problem.

We moved on to introduce tasks that are commonly seen in the literature and help motivate the use of network embeddings in Section 2.2, namely (multi-label) node classification, node clustering and node visualization. Finally, we overviewed the related work in Section 2.3. Specifically, in Section 2.3.1 we first discussed the evolution of random walk-based methods for network embeddings, ranging from simpler techniques (using first order or biased/parametrized random walks) in the context of homogeneous networks to more complex ones for heterogeneous networks, including a brief presentation of meta-paths based models. In Section 2.3.2 we presented network embedding works that make use of node centrality information at some point in the learning process (not necessarily making use of random walks). We touched on works that leverage measures such as PageRank and other centrality information to bias walks and also to define new loss functions.

In the next chapter, we shall propose a random walk-based network representation learning framework that consists of customizable modules so as to offer some degree of flexibility when learning embeddings. We also formally define some of the walk sampling strategies – introduced in this Chapter – in the context of our learning framework.

Chapter 3

Sample Reduction Strategies for Heterogeneous Network Embedding

3.1 Network Representation Learning Framework

In this section, we present a learning framework in terms of the Skip-gram with Negative Sampling model [Mikolov et al., 2013a,b] paired with a customizable random walk module and formalize three representative random walk techniques for network embeddings with the aim of providing a principled and organized way to analyze the impact of sample reduction strategies.

3.1.1 Network Embeddings with Skip-gram

Initially conceived from a language modeling perspective, the Skip-gram model aims to learn a set of parameters that maximize the probability of observing neighboring (context) words conditioned on a given word. In order to reframe this approach as a network embedding problem, one must consider nodes in networks instead of words in text *corpora*. Thus, we define S, a sampling strategy that generates a set of node sequences given a network (akin to sentences – word sequences – in some text document). Given a sequence $v_1, v_2, ..., v_n$ of length n sampled with S, we define $C(v_i) \subset V$, the context of a node v_i, considering a window of size k around the center node: $C(v_i) = \{v_{i-k}, ..., v_{i-1}, v_{i+1}, ..., v_{i+k}\}$. In practice, the context window size is sampled uniformly from $1..k$ for each node in the network [Mikolov et al., 2013a].

3.1.1.1 Basic Skip-gram Formulation

We would like to learn a function $f : V \rightarrow \mathbb{R}^d$, which can be seen as a parameter matrix of size $|V| \times d$, that maps each node $v \in V$ to a d-dimensional parameter vector (its output node embedding). In order to learn the parameters, we use a simple feedforward neural network [Rumelhart et al., 1985], with our parameter matrix being the corresponding weights of the single hidden layer in this neural network. We train the parameters using Asynchronous Stochastic Gradient Descent (ASGD) [Dean et al., 2012] with Adagrad [Duchi et al., 2011].

Formally, given the nodes $v \in V$ and their respective contexts $C(v)$, we want our neural network with parameters θ to maximize the following probability:

$$\arg\max_{\theta} \prod_{v \in V} \prod_{c \in C(v)} p(c \,|\, v; \theta) \tag{3.1}$$

Intuitively, given v, we would like to correctly predict its context (neighboring) nodes $c \in C(V)$. The probability $p(c \,|\, v; \theta)$ is modeled with a softmax function:

$$p(c \,|\, v; \theta) = \frac{exp(\theta_c \cdot \theta_v)}{\sum_{c' \in V} exp(\theta_{c'} \cdot \theta_v)}, \tag{3.2}$$

where θ_c and $\theta_v \in \mathbb{R}^d$ are vector representations for context and center nodes, respectively (the same node has two representations: one for its center role and another one used when it appears in another node's context).

Now, we plug Eq. (3.2) in Eq. (3.1) and take the logarithm to obtain the following objective function:

$$\arg\max_{\theta} \sum_{v \in V} \left[\left(\sum_{c \in C(v)} \theta_c \cdot \theta_v \right) - \left(\log \sum_{c' \in V} exp(\theta_{c'} \cdot \theta_v) \right) \right] \tag{3.3}$$

When dealing with large networks, the summation $\sum_{c' \in V} exp(\theta_{c'} \cdot \theta_v)$ in Eq. (3.3) becomes prohibitively expensive to compute in practice due to the many possible combinations of center and context nodes. In order to overcome this issue, researchers typically resort to either hierarchical softmax [Morin and Bengio, 2005] or negative sampling [Mikolov et al., 2013b] to approximate the objective function. We choose to work with the latter due to its simplicity and efficiency in comparison to the hierarchical softmax.

3.1.1.2 Negative Sampling

Proposed as a simplification of Noise-Contrastive Estimation (NCE) [Gutmann and Hyvärinen, 2012], which can be shown to approximately maximize the log probability of the softmax, negative sampling is modeled after a different task, namely, telling whether a given (center,context) pair came from the set of observed data $D = \bigcup_{v \in V} v \times C(v)$ or not. Formally, we define a binary random variable Y that models the probabilities of a sample pair (v, c) belonging $(Y = 1)$ or not $(Y = 0)$ to D, parametrized by θ. We also assume that observations are independent of each other.

Mikolov et al. [2013b] model the probability $p(Y = 1|v, c, \theta)$ with a sigmoid, as defined in Eq. (3.4). Also, note that $p(Y = 0|v, c, \theta) = 1 - p(Y = 1|v, c, \theta)$.

$$p(Y = 1|v, c, \theta) = \sigma(\theta_c \cdot \theta_v) = \frac{1}{1 + exp(-\theta_c \cdot \theta_v)} \tag{3.4}$$

We then are interested in maximizing the following (conditional) likelihood function:

$$\begin{aligned} &\arg\max_{\theta} \prod_{i=1}^{|D|} p(Y = y_i|v_i, c_i, \theta) \\ =&\arg\max_{\theta} \prod_{i=1}^{|D|} p(Y = 1|v_i, c_i, \theta)^{y_i}\, p(Y = 0|v_i, c_i, \theta)^{1-y_i} \end{aligned} \tag{3.5}$$

Note, though, that if we were to rely solely on observed data, we would not have any examples of $p(Y = 0|v, c, \theta)$, and therefore the likelihood would be simplified to (we omit the indices i due to all observed data belonging to D):

$$\begin{aligned} &\arg\max_{\theta} \prod_{(v,c) \in D} p(Y = 1|v, c, \theta) \\ =&\arg\max_{\theta} \log \prod_{(v,c) \in D} p(Y = 1|v, c, \theta) \\ =&\arg\max_{\theta} \sum_{(v,c) \in D} \log p(Y = 1|v, c, \theta) \end{aligned} \tag{3.6}$$

However, Goldberg and Levy [2014] show that if we set $\theta_v = \theta_c$ and $\theta_v \cdot \theta_c = K$ for large enough values of K, we can obtain a trivial solution to objective 3.6, such that $p(Y = 1|v, c, \theta) = 1$ for all (v, c) pairs. As such, we must find a way to come up with "negatively weighted" (v, c) pairs, that is, the set D' of negative samples. Therefore,

we generate D' by randomly sampling γ unobserved pairs from the distribution $(v, c) \sim freq(v)\frac{freq(c)^{3/4}}{Z}$, where $freq(\cdot)$ denotes the relative frequency of a node in relation to all node sequences sampled with strategy S and Z is a normalization constant. According to Mikolov et al. [2013b], the exponent of $\frac{3}{4}$ was obtained experimentally and is generally left like that in most works and implementations.

Now that we have our set D' of negative samples, we can return to Eq. (3.5) and redefine it using D':

$$
\begin{aligned}
&\arg\max_{\theta} \prod_{(v,c)\in D} p(Y=1|v,c,\theta) \prod_{(v',c')\in D'} p(Y=0|v',c',\theta) \\
=&\arg\max_{\theta} \prod_{(v,c)\in D} p(Y=1|v,c,\theta) \prod_{(v',c')\in D'} (1-p(Y=1|v',c',\theta)) \\
=&\arg\max_{\theta} \sum_{(v,c)\in D} \log\, p(Y=1|v,c,\theta) + \sum_{(v',c')\in D'} \log\,(1-p(Y=1|v',c',\theta)) \\
=&\arg\max_{\theta} \sum_{(v,c)\in D} \log\left(\frac{1}{1+exp(-\theta_c\cdot\theta_v)}\right) + \sum_{(v',c')\in D'} \log\left(1-\frac{1}{1+exp(-\theta_{c'}\cdot\theta_{v'})}\right) \\
=&\arg\max_{\theta} \sum_{(v,c)\in D} \log\left(\frac{1}{1+exp(-\theta_c\cdot\theta_v)}\right) + \sum_{(v',c')\in D'} \log\left(\frac{1}{1+exp(\theta_{c'}\cdot\theta_{v'})}\right) \\
=&\arg\max_{\theta} \sum_{(v,c)\in D} \log\,\sigma(\theta_c\cdot\theta_v) + \sum_{(v',c')\in D'} \log\,\sigma(-\theta_{c'}\cdot\theta_{v'}),
\end{aligned}
\tag{3.7}
$$

which is the formulation for Skip-gram with Negative Sampling (SGNS) used in practice.

3.1.2 Customizable Random Walks

We now present a general formulation for the walk sampling and feature learning algorithms, which take into account the sampling strategy S, as well as a set U of starting nodes, with $U \subseteq V$, and a function $r : V \to \mathbb{N}$ that defines the number of walks per node. As previously described in Figure 2.1, we are given an input (heterogeneous) graph, on which we execute our RandomWalk algorithm, as described below. We then use the output walks as input training samples to Network Embedding with Skip-gram (Eq. 3.7) to obtain our embeddings.

When designing the algorithm in such way, we distinguish our work from previous research by allowing for custom choices of U and $r(\cdot)$. In doing so, we provide additional flexibility in the design of random walks and, consequently, in generating the output

embeddings. Thus, in the following sections, we proceed to present our definitions of $r(\cdot)$ and U for heterogeneous networks. It is worth noting that this formulation also opens up new possibilities for future work in terms of novel design choices for random walks that are complementary to the ones presented here.

Algorithm 1: RandomWalk

Input : Network $G = (V, E, \phi, \psi)$; Walk length l; Walks per node $r(\cdot)$; Sampling strategy S; Set of starting nodes U.

Output: Set *walks* of sampled node sequences.

```
Initialize walks = ∅
for all nodes v ∈ U do
    for i = 1 to r(v) do
        Initialize walk = {v}
        for j = 1 to l do
            curr_node = walk[−1]
            s = S(curr_node)
            walk = walk ∪ {s}
        end
        walks = walks ∪ {walk}
    end
end
return walks
```

As reviewed in Section 2.1, the sampling strategy S is often the main research focus when it comes to network embeddings. In order to evaluate the impact of our modifications, we choose to use three well established sampling strategies from the literature, as detailed in the following section.

3.2 Sampling Strategies

Sampling strategies define the transition probabilities required in order to decide where to go next (sample the next node) during a random walk. These range from simple strategies (e.g. uniform transition probabilities with no memory) to more complex ones (higher order Markov chains with complex transitions). In this section, we present and detail three different ways of sampling nodes in a random walk, which we use to test and evaluate our work. For ease of reference, we name the strategies according to the network embedding model that first introduced them.

Note that other proposed sampling strategies can benefit from the techniques presented in the next sections. The strategies in this section are by no means exhaustive and were chosen for experimental purposes.

3.2.1 DeepWalk

Given a node v, DeepWalk [Perozzi et al., 2014] samples the next node considering a uniform distribution on its neighbors. Thus, we can define the following transition matrix:

$$D_{ij} = p(v_j|v_i) = \begin{cases} \frac{1}{deg(v_i)}, & \text{if } (v_i, v_j) \in E \\ 0 \quad , & \text{otherwise} \end{cases} \tag{3.8}$$

In this case, we make no distinction whatsoever when choosing the next node. As seen in Figure 3.1, we assign the same probability to each and every neighbor of the current node. This is the simplest type of walk we can choose for our learning process.

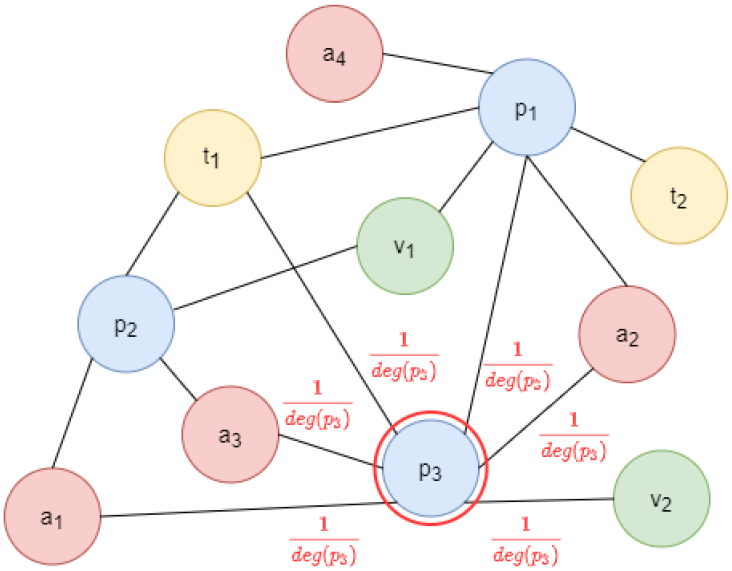


Figure 3.1: DeepWalk transition probabilities. Suppose our walker is currently waiting on node p_3, we then have that the transition probability to every neighbor is uniform and equal to $\frac{1}{deg(p_3)}$.

3.2.2 node2vec

The node2vec [Grover and Leskovec, 2016] model makes use of a 2nd-order Markov chain to model a transition probability tensor using two additional parameters p and q. With this, they attempt to provide the means to tune the walks between the traditional breadth-first and depth-first search strategies. Formally, assuming the walker has recently moved from node v_i to v_j via edge (v_i, v_j), we define the unnormalized transition probability for neighbors $v_k \in N(v_j)$ as $\pi_{ijk} = \alpha_{ijk} \cdot w_{jk}$, where w_{jk} is the weight of edge (v_j, v_k) and

$$\alpha_{ijk} = \begin{cases} \frac{1}{p}, & \text{if } (v_i, v_j) \in E, (v_j, v_k) \in E, i = k \\ 1, & \text{if } (v_i, v_j) \in E, (v_j, v_k) \in E, i \neq k, (v_k, v_i) \in E \\ \frac{1}{q}, & \text{if } (v_i, v_j) \in E, (v_j, v_k) \in E, i \neq k, (v_k, v_i) \notin E \\ 0, & \text{otherwise} \end{cases} \tag{3.9}$$

Then, we normalize the probabilities to obtain the following probability tensor:

$$N_{ijk} = p(v_k | v_i, v_j) = \frac{\pi_{ijk}}{\sum_{v_{i'} \in N(v_j)} \pi_{i'jk}} \tag{3.10}$$

In Figure 3.2, we present the values for α, given $i = p_1$ and $j = p_3$ for all possible k nodes. The *return parameter* p allows for the control of revisiting the immediate previous node v_i in the walk. Setting it high helps avoid frequent 2-hop redundancy in sampling and forces the walker to explore further. The *in-out parameter* q controls the likelihood of visiting nodes that are close to node v_i. It encourages/discourages exploration away from previously visited regions in the graph. For nodes that are adjacent to the previously visited node (t_1 and a_2 in our example), node2vec assigns a neutral weight as it considers it not to be a case of returning nor exploring ahead.

3.2.3 JUST

Specifically tailored for heterogeneous networks, JUST's [Hussein et al., 2018] *JUmp* and *STay* walks make use of the set of node types A to guide its walks and avoid prioritizing nodes of certain types due to their high frequency in the graph, along with a memory of size m, modeled as a queue Q_{hist} to keep track of the previously visited node types. This leads JUST to model the transition probabilities in two steps. Given that

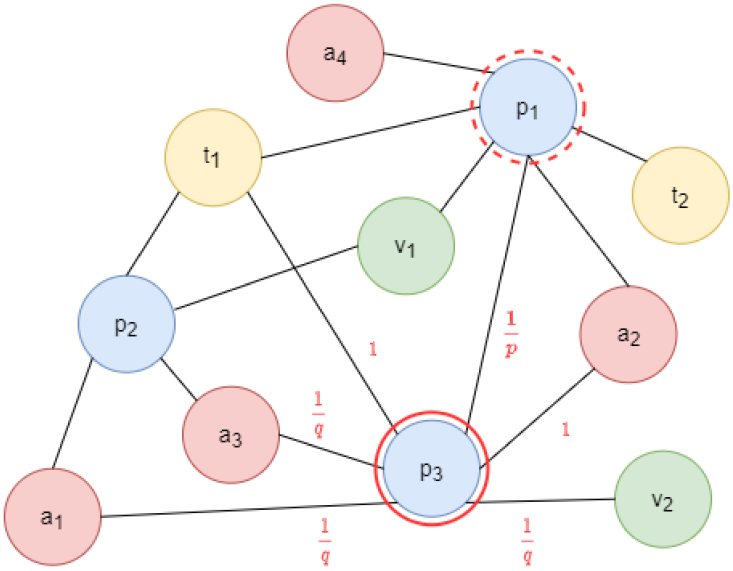


Figure 3.2: node2vec's α values given $i = p_1$ (previously visited node) and $j = p_3$ (current node). After these values are defined, we use those to compute N_{ijk} (Equation 3.10) to define our transition probabilities.

we are currently visiting node v_j with type $\phi(v_j)$, we decide whether to stay in the same type domain with probability

$$p_{stay}(v_i) = \begin{cases} 0, & \text{if } N^{\phi(v_i)}(v_i) = \emptyset \\ 1, & \text{if } N(v_i) - N^{\phi(v_i)}(v_i) = \emptyset \\ \alpha^d, & \text{otherwise} \end{cases} \tag{3.11}$$

where α is the initial stay probability that decreases with an exponential decay based on d, the number of previous consecutive nodes of same type as v_i in the walk. This helps prevent the walker from staying in a single domain for too long and encourages exploration of different node types. Once we decide to stay on the same domain, we sample the next node v_j with probability $p(v_j|v_i, stay) = \frac{1}{deg^{\phi(v_i)}(v_i)}$. Alternatively, we decide to jump to a different domain with probability $p_{jump}(v_i) = 1 - p_{stay}(v_i)$. In this case, we will try to avoid jumping to the m previously visited node types (stored in $Q_{hist} = \{\phi(v_i), ...\}$), unless there is no other option. When jumping, the next node is sampled uniformly from the following set:

$$Q_{jump}(v_i) = \begin{cases} \{v|v \in \bigcup_{t \in A - Q_{hist}} N^t(v_i)\}, & \text{if set not empty} \\ \{v|v \in N(v_i) - N^{\phi(v_i)}(v_i)\}, & \text{otherwise} \end{cases} \tag{3.12}$$

In Figure 3.3, we see an example of the JUST sampling strategy. Considering we have previously visited node a_2 and moved afterwards to node p_3, we have $Q_{hist} = \{a, p\}$. In this case, we either stay (with probability α^l) or, since we have other types of neighbors (not in Q_{hist}), if we decide to jump, we will attempt to move to either t or v nodes. In

essence, we try to sample from the nodes whose types are not in Q_{hist}. However, it might be the case that there are no nodes of that type left (i.e. the only neighbors are either homogeneous or their types were recently visited). In that case, we relax the restriction and sample uniformly from the set of heterogeneous neighbors.

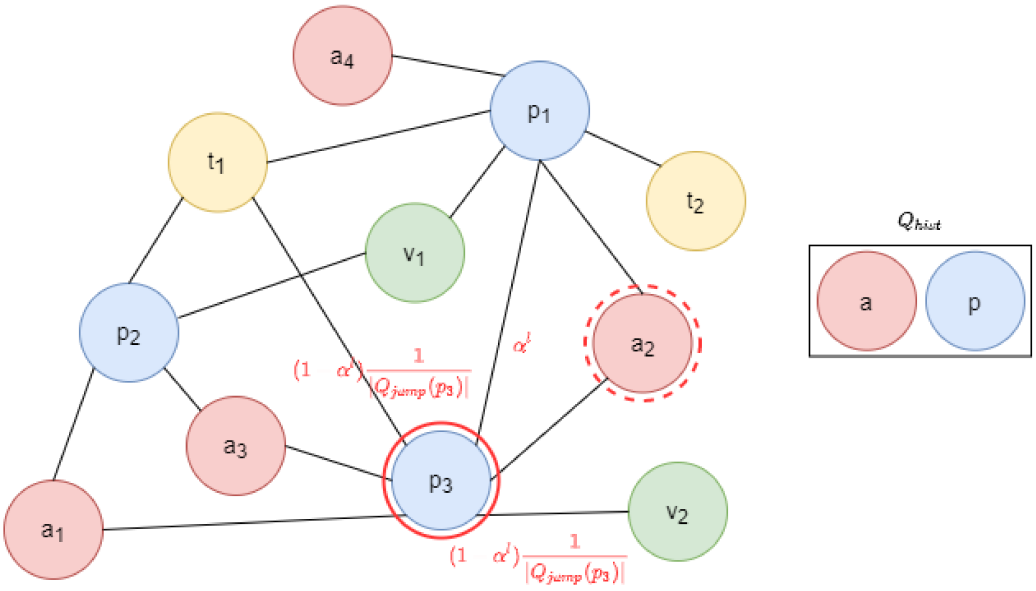


Figure 3.3: JUST transition probabilities given that we previously visited node a_2 and currently wait at node p_3.

3.3 Centrality-based Walks

In this section, we formalize and motivate our choice for centrality-based walks in arbitrary heterogeneous networks. First, we propose our definition of a walks per node function $r(\cdot)$, then we move on to draw parallels from information theory and the theory of Markov chains to bring theoretical insights to our choices.

Most models fix a number of walks starting from every node, that is, $r(u) = r(v) = \eta$, for all $u, v \in V$. Gao et al. [2018], however, acknowledge the importance of centrality in networks, and define the initial idea of centrality-based walks in bipartite networks. In their work, the number of walks starting from a vertex is a function of its centrality (their work used the HITS [Kleinberg, 1999] authority score as the centrality measure), normalized by the degrees in that vertex's partition. We extend that idea to different node types in a heterogeneous network, normalizing the centrality measure by other nodes of

the same type. Formally, we define:

$$r(v) = max(L_{min}, L_{max} \cdot \hat{c}(v)), \tag{3.13}$$

where L_{min} and L_{max} are hyperparameters that control the minimum and maximum walk lengths, respectively, and $\hat{c}(v)$ is the *heterogeneous normalized centrality* of a given node centrality measure $c(v)$, defined by:

$$\hat{c}(v) = \frac{c(v)}{\sum_{v' \in V^{\phi(v)}} c(v')} \tag{3.14}$$

With this formulation, we define the relative centrality of a node within its type subset. By doing this, we make sure to pay attention to nodes of all types, regardless of their frequency in the network, while at the same time prioritizing central nodes in comparison to nodes of that same type.

The choice of $c(v)$ is left to the user, who can then define which aspect of centrality they would like to capture, such as eigenvector centrality [Bonacich, 1987], betweenness centrality [Brandes and Pich, 2007], among others. Since we are interested in scalability, we would not like any of our strategies to require additional computations. As such, we opt to utilize the degree centrality $c(v) = deg(v)$ in our work, as it can be computed at graph construction time. Additionally, the degree of a node is associated with other quantities, which help bring insight to our work, as shown in the following sections.

3.3.1 Motivation: Stationary Distribution

Consider a random walk with uniform transition probabilities on a simple, non-bipartite, homogeneous graph. In this case, we have that the corresponding Markov chain is ergodic [Chung and Graham, 1997]. Thus, we can conclude that, regardless of the initial distribution P_0, there exists a unique stationary distribution π such that for all $v_i, v_j \in V$,

$$\lim_{t \to \infty} P_0 D_{ij}^t = \pi(v_j)$$

(recall that D_{ij} is the transition matrix defined in Section 3.2.1). Additionally, in this case, we have that the stationary distribution [Brouwer and Haemers, 2011] is given by:

$$\pi(v) = \frac{deg(v)}{2 \cdot |E|}$$

Note, however, that [Bondy et al., 1976]:

$$\sum_{v \in V} deg(v) = 2 \cdot |E|$$

Thus:

$$\pi(v) = \frac{deg(v)}{\sum\limits_{v' \in V} deg(v')} \tag{3.15}$$

which is exactly our definition of $\hat{c}(v)$.

With this in mind, let us consider the estimation of P_0. We do not know P_0 directly, but we have many generated walk samples from our graph. We assume that the probability of starting a walk on a given node v is given by the number of samples in our set whose starting node is v. Therefore, for models that fix a constant number of walks η for every node, we have:

$$P_0(v) = \frac{r(v)}{\sum\limits_{v' \in V} r(v')} = \frac{\eta}{\sum\limits_{v' \in V} \eta} = \frac{1}{|V|} \tag{3.16}$$

that is, a uniform distribution on the set of vertices.

On the other hand, if we disregard the limiting hyperparameter L_{min}, we have, in our case:

$$P_0(v) = \frac{r(v)}{\sum\limits_{v' \in V} r(v')} = \frac{L_{max} \cdot \hat{c}(v)}{\sum\limits_{v' \in V} L_{max} \cdot \hat{c}(v')} = \frac{\frac{deg(v)}{2 \cdot |E|}}{\sum\limits_{v' \in V} \frac{deg(v')}{2 \cdot |E|}} = \frac{deg(v)}{\sum\limits_{v' \in V} deg(v')} = \pi(v) \tag{3.17}$$

We observe that our approach can then be seen as an attempt to sample walks following the stationary transition probabilities on homogeneous graphs, which, given enough time, is the natural tendency that random walks follow in the network.

For heterogeneous networks, we would have to take node types into account and this result would also be valid if the sum of degrees of all nodes of a given type would equal the same value for every node type in the network.

3.3.2 Motivation: Preferential Attachment

As pointed out by Barabási and Albert [1999], in real world networks, new nodes tend to link themselves to more connected, or highly visible, nodes. For instance, in scientific networks, we are more likely to read and a cite a paper with a lot of citations, since

our time is limited and we simply cannot read each and every published paper out there. This phenomenon is observed in many types of networks, such as social networks, where famous people are followed by a huge number of regular users, and even the world wide web, where tech giants concentrate a big part of global accesses and online interactions [Moore, 2016].

On top of that, when we work with data from real networks, we are actually looking at a snapshot of the current state of that network at a certain point in time. The data on each node is by no means complete and we must take that into account when designing our algorithms. In recommender systems literature, this problem is known as cold start [Aggarwal et al., 2016], but the concept extends to any statistical inference problem. In essence, low degree vertices do not offer enough information for us to make good guesses about their real preferences/properties (e.g. other nodes that it might connect to at some point in the future). Also, the less information we have on a node, the more likely it is for us to put too much emphasis on it, which is a problem in case the information is wrong or noisy. Conversely, as high degree nodes tend to attract new edges, they should be a good source of information for our model and the information we obtain from it should be resistant to noisy inputs.

In order to model the preferential attachment behavior, the authors model the probability Π of a recently added node connecting to node v in the graph based on its degree:

$$\Pi(v) = \frac{deg(v)}{\sum_{v' \in V} deg(v')} \tag{3.18}$$

which, just as in the previous section, comprises our proposed measure of centrality for the walks.

3.3.3 Motivation: Entropy

Again, consider a random variable X_i with support V, representing the outcome of transitioning to an arbitrary neighbor v of node v_i modeled after its transition probabilities.

If we consider the uniform transition probability case (DeepWalk), we can compute the entropy of X_i, following Shannon [1948]'s definition:

$$H(X_i) = \sum_{v \in V} p(v|v_i)h(v), \tag{3.19}$$

where $h(x)$ is the Shannon information content of an outcome, defined as:

$$h(x) = \log_2 \frac{1}{p(x)} \tag{3.20}$$

Plugging Eq. (3.20) in Eq. (3.19), we obtain:

$$H(X_i) = \sum_{v \in V} p(v) \log_2 \frac{1}{p(v|v_i)} = \sum_{v \in V} \frac{1}{deg(v_i)} \log_2 deg(v_i) = \log_2 deg(v_i) \tag{3.21}$$

This quantity is also known as the *node entropy* [Small, 2013] of node v_i. Entropy is usually connected to the uncertainty associated with a random variable. Thus, Eq. (3.21), points us to the fact that, between any two nodes, the one with high degree will also have higher entropy/uncertainty.

The simple, yet useful idea of working with node degrees has proved to be effective and arise naturally in perspectives coming from three different fields, as illustrated above. This helps corroborate our choice of using the degree in our work instead of more complex centrality measures. It also provides a bridge for us to look at the theoretical and empirical results of these (and other) fields to try and understand our models with more depth.

3.4 Focused Walks

As previously mentioned in Section 3.1.2, the set U of nodes on which we start our random walks provides another opportunity for walk customization. In this section, we motivate and explain our proposed use of such subset.

While in some cases it might be useful to generate representations for nodes of all types in the same space, there are several practical applications in which we are only interested in a subset of node types for one or more tasks in particular. For instance, in the context of academic networks, we might be interested in embeddings of papers (for clustering and/or classification purposes), or perhaps authors and papers (for recommendation purposes). In the case of a social network a user might want to take advantage of the heterogeneous structure of a network (e.g. the structure and semantics of the heterogeneous edges), but only keep the representations for user and item node types. As another example, when dealing with biological (onthology) networks [Valdeolivas et al., 2019], for instance, one would focus in gene/protein and disease nodes in order to use their representations to infer new possibilities of interactions.

We posit that, if we take the task in which we would like to apply our embeddings

into account, we are able to *focus* on the right subset of node types, starting walks from the nodes for which we want to generate meaningful output representations. In doing so, we are able to relieve a strong restriction in random walk-based methods: we no longer need to guarantee an output representation for every node in our network. As we will see in the experimental results, in cases where our desired subset of output nodes is significantly smaller than the complete set of nodes, we are able to significantly reduce the number of generated samples, while still learning good representations.

It is worth mentioning that, due to the design of our learning algorithm, as long as we visit a node once, we end up generating a set of parameters for it and, thus, an output representation. As such, our parameter matrix in this case has dimensions $\nu \times d$, bounded by $|U| \leq \nu \leq |V|$.

Formally, we only require that $U \subseteq V$. In the context of our work, given a subset Γ of node types for which we are interested in generating embeddings, we then define $U = \{v \in V \,|\, \phi(v) \in \Gamma\}$.

Since the choice of Γ is task-dependent, we define the subset for each of the datasets presented in the experimental section.

Finally, even if we have a scenario where $\Gamma = V$, but we do not require every embedding to be in the same vector space, we hypothesize that it is better to break the learning process into multiple stages of learning with focused walks (each emphasizing a different subtask), as we believe it will result in better representations. This hypothesis, however, has not been tested and is left for future research.

3.5 Summary

In this chapter, we presented a framework for random walk-based representation learning, which provides abstractions of certain algorithmic steps in the form of modules/components so as to provide a degree of customizability, which are also detailed in the sections of this chapter. In particular, in Section 3.1, we presented the mathematical formulation and motivation of the basic Skip-gram model for network embeddings, as well as its updated version with negative sampling. Afterwards, we provided a general learning algorithm with its associated modules and loss function. In Section 3.2, we discussed the first component, presenting our choices of sampling strategies motivated by related work.

In the following sections, we presented our contributions in the form of the two remaining customizable modules. First, in Section 3.3, we introduced our idea of *centrality-based walks* for heterogeneous networks, which make of use of node centrality per node type in order to define the number of walks started from a given node, providing a way

to increase/reduce the number of walks based on the centrality of a given node in the network. In the same section, we also draw parallels between our chosen centrality measure and concepts from other areas to motivate our choice. Finally, in Section 3.4, we proposed *focused walks*, which consist of defining a focused subset of node types from which we start walks in our algorithm, contributing to a reduction on the total number of walks

Now that we have defined our framework, the following chapter will cover the experimental setup used in our experiments aimed at answering our research questions. We will present the chosen datasets, metrics, baseline models and training choices (hyperparameter tuning and train/test procedures).

Chapter 4

Experimental Setup

In this chapter, we present the experimental setup of our work, which helps evaluate our proposed *centrality* and *focused* walks, defined in Chapter 3. With these experiments, we wish to address the following research questions:

RQ1. To what extent can we improve efficiency without harming effectiveness?

RQ2. How do different random walk strategies impact our approach?

RQ3. How do different node properties impact our approach?

RQ4. Are traditional sampling approaches generating redundant training data?

In order to empirically verify our hypotheses, we run experiments on three publicly available real-world networks. We focus our efforts on three main tasks/applications in network embedding literature: multi-label node classification, node clustering and visualization, as presented in Section 2.2. For each of the datasets and sampling strategies defined in Section 3.2, we test our strategies both paired together and in an isolated manner.

In the following Sections, we describe the datasets, along with the corresponding evaluation metrics used for each task. We also present our choice of baseline models for result comparison. Finally, we present our training procedure in terms of train/test data splitting, learning settings and hyperparameters.

4.1 Datasets

We have conducted our experiments on three publicly available real world heterogeneous network datasets. In this section, we describe and provide statistics for these datasets. Tables 4.1 and 4.2 presents a summary of statistics of these networks. Figure 4.1 presents the degree distribution per node type for each dataset.

Table 4.1: Summary of datasets.

Dataset	DBLP	Foursquare	MOVIE
$\|V\|$	15649	29771	22857
$\|E\|$	51363	83407	83988
$\|E\|_{homogeneous}$	6984 (13.6%)	5695 (6.8%)	49634 (59.1%)
$\|E\|_{heterogeneous}$	44379 (86.4%)	77712 (93.2%)	34354 (40.9%)
# Labels	4	10	5

- MOVIE (YAGO subset) [Huang and Mamoulis, 2017]: a subset of the YAGO [Suchanek et al., 2007] knowledge base consisting of movie information. This dataset contains four node: 11069 actors (A), 7290 movies (M), 3015 directors (D) and 1483 composers (C). The heterogeneous edges on this graph are **A-M**, **D-M** and **C-M**. Additionally, there are homogeneous edges between actors **A-A** and movies **M-M**. Each movie is associated with one or more genre labels among "*action*", "*horror*", "*adventure*", "*scifi*" and "*crime*";

- DBLP 4-Area [Huang and Mamoulis, 2017; Sun et al., 2011]: a subset of the DBLP academic database comprising authors from four big computer science subareas: *database*, *data mining*, *machine learning* and *information retrieval*. There are 5237 papers (P), 5915 authors (A), 18 venues (V) and 4479 topics (T). The heterogeneous edges are **A-P**, **P-V** and **P-T**. In this dataset, there are homogeneous edges between papers **P-P**;

- Foursquare (NYC) [Yang et al., 2015, 2016]: a check-in network collected from Foursquare check-ins in New York City. It consists of 2449 users (U), 25904 check-ins (C), 1250 points of interest/venues (P) and 168 timestamps (T). Heterogeneous edges **U-C**, **P-C** and **T-C** are available, while homogeneous edges are defined between users **U-U**. Additionally, each point of interest is associated with a label among 10 categories.

We can observe in Figure 4.1 that, in all datasets, the degree distribution of most node types roughly follow a power law, which is a trait of a scale-free network [Barabási et al., 2016]. In practical terms, it means that there are a lot of nodes with low degree connectivity, and a few nodes with high degree. In particular, due to the construction of the Foursquare dataset, all Check-in nodes have exactly 3 edges.

Finally, in Figure 4.2, we provide statistics regarding the class distribution for each dataset. Due to the randomized nature of the Foursquare dataset, we lack precise label information: although we know there are ten venue categories, we do not know whether those are bars, restaurants, etc, as those were not publicly available.

Table 4.2: Node type distribution per dataset.

DBLP		Foursquare		MOVIE	
Type	Count	Type	Count	Type	Count
Papers	5237 (33.5%)	Users	2449 (8.2%)	Actors	11069 (48.4%)
Authors	5915 (37.8%)	Check-ins	25904 (87.0%)	Movies	7290 (31.9%)
Venues	18 (0.1%)	Points of interest	1250 (4.2%)	Directors	3015 (13.2%)
Topics	4479 (28.6%)	Timestamps	168 (0.6%)	Composers	1483 (6.5%)

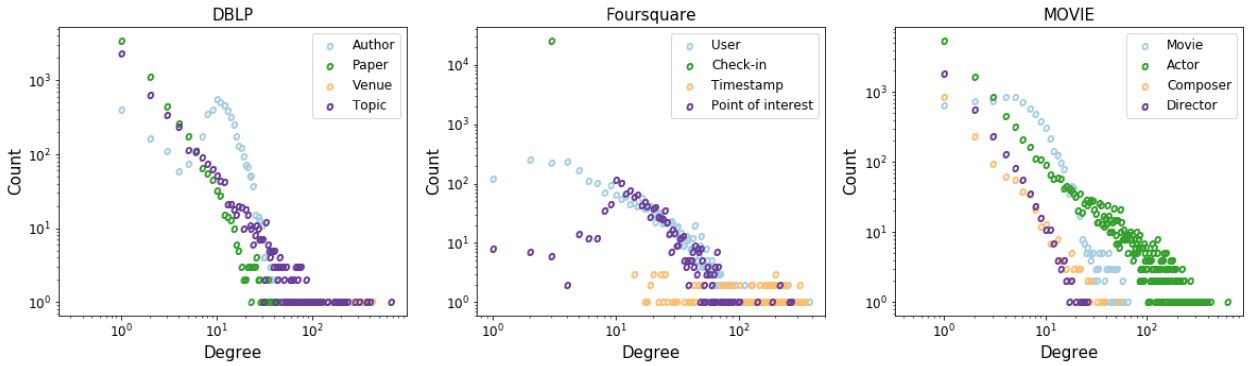


Figure 4.1: Degree Distribution per Node Type.

We observe that DBLP is very balanced in terms of labels, whereas Foursquare and MOVIE present unbalanced classes. Note that the percentage of classes in MOVIE go over 100% due to the possibility of a movie being assigned one or more genres, which leads us to a multi-label classification problem. We present label statistics for the MOVIE in Table 4.3. We note that although the vast majority of movies are associated with a single genre, about 15% of the movies have two or more labels.

Table 4.3: Frequency distribution of labels per sample for the MOVIE dataset.

# Labels	Samples
1	6216 (84.78%)
2	999 (13.63%)
3	106 (1.45%)
4	11 (0.15%)

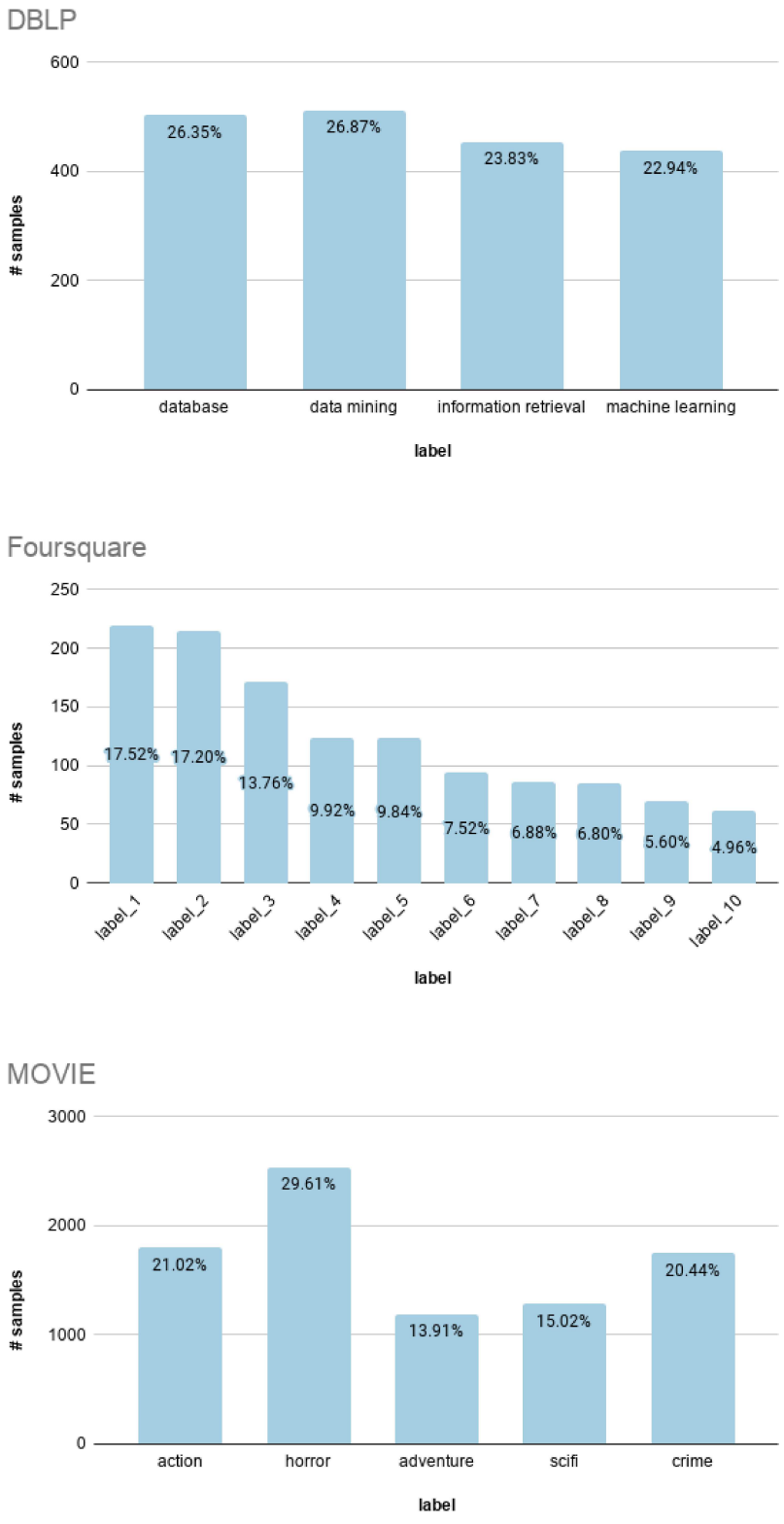


Figure 4.2: Label distribution per dataset.

4.1.1 Focused Node Subsets

For our experiments, we are interested in the embeddings of the set of nodes for which we have labels in the aforementioned datasets. Therefore, when defining our set U for the Focused walks strategy, we choose to focus on the labeled set of nodes for each dataset, as presented in the following table.

Table 4.4: Focused subset U per dataset.

Dataset	Focused subset
DBLP	Authors
Foursquare	Points of interest
MOVIE	Movies

4.2 Evaluation Metrics

4.2.1 F1-score

The F1-score [Chinchor, 1992; Van Rijsbergen, 1979] is an accuracy measure usually applied in the context of binary classification. It combines the *precision* and *recall* of the classifier, that is, the fraction of true positive examples among the ones that the model classified as positive, and the fraction of positives classified by the model among all positive examples, respectively.

We make use of the usual formulation of the F1-score, obtained by setting the F-score factor $\beta = 1$, yielding the harmonic mean of precision and recall:

$$\mathrm{F1} = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{4.1}$$

Since we are working with multi-label classification, we consider two extensions [Yang and Liu, 1999] of the F1-score: Macro-F1, where we compute metrics for each label, then calculate their unweighted mean, and Micro-F1, where we compute metrics globally by counting total true positives, false negatives and false positives (in this case, we account for class imbalance).

4.2.2 Normalized Mutual Information

Used for clustering experiments, Normalized Mutual Information (NMI) [Kvalseth, 1987] normalizes the Mutual Information [Shannon, 1948] measure, widely used in the field of Information Theory, by averaging via harmonic mean, yielding the following equation:

$$NMI(Y, C) = \frac{2 \cdot I(Y; C)}{H(Y) + H(C)}, \tag{4.2}$$

where Y and C are the class and cluster labels, respectively. $H(\cdot)$ is the entropy, as defined in Equation 3.19, and $I(Y; C)$ is the mutual information, given by:

$$I(Y; C) = H(Y) - H(Y|C) = H(C) - H(C|Y), \tag{4.3}$$

This measure gives us a good idea of the clustering quality in the occasions where we have the class labels at our disposal.

4.3 Baseline Models

Our experimental setup is meant to answer whether the use of our strategies can help in reducing the required number of samples to obtain competitive results when using random walk-based algorithms. To that end, we consider three well established models in the literature, which use the sampling strategies presented in Section 3.2: DeepWalk, node2vec and JUST. We set the standard random walk parameter settings for all baselines, that is, number of walks per node $r = 10$, walk length $l = 100$ and window size $k = 10$. We define the embedding dimension to be $d = 128$ and the number of negative samples $\gamma = 5$. Additionally, we set the parameters that are particular for each model as follows:

- node2vec: return parameter $p = 0.5$, in-out parameter $q = 0.5$;
- JUST: memory size $m = 2$, initial stay parameter $\alpha = 0.25$.

In order to further contrast the impact of our proposed strategies, in addition to comparing the models based on their usual settings, we propose a *Basic* sample reduction strategy. In order to generate a reduced number of samples for each of the baseline models, we simply define our *Basic-1* and *Basic-2* baselines by reducing the number of walks started per node (usually set to 10) to 1 and 2, respectively. With this set up, we end up generating 10% and 20% of the usual amount of samples, which is closer to

the values we obtain with our strategies. Do note that these simple strategies do not allow for a finer tuning of sample reduction such as the one we manage to achieve with the variation of hyperparameters in centrality walks. By merely changing the number of walks started per node, we are limited to having big leaps in terms of generated samples (as we have to apply the same number to every node in the graph).

4.4 Learning and Evaluation Procedures

4.4.1 Hyperparameter Tuning

Following [Grover and Leskovec, 2016], we obtain the best values for our L_{min} and L_{max} parameters using 10-fold cross validation on 10% of the available datasets with a grid search over $L_{min} \in \{1, 2, 3, 4, 5, 10, 20\}$ and $L_{max} \in \{20, 24, 28, 32, 36, 40\}$. Our final hyperparameter choice takes not only the metrics into account, but also its balance in relation to the relative number of generated samples. In general, we would like to reduce the most our number of generated samples while obtaining good results. Therefore, we define $L_{max} = 32$ for every case, and in general, we set $L_{min} = 1$. The only exceptions happen when using the Focused strategy with the Foursquare dataset: due to the small amount of target labeled nodes (4.2% of the whole network, as seen in Table 4.2), we require a significantly bigger amount of walks to obtain good results. As such, in this context we set $L_{min} = 20$ when using Focused and Centrality strategies together, and $r = 20$ when using the Focused strategy on its own.

4.4.2 Train/Test Procedure

We follow the train/test settings defined in [Perozzi et al., 2014; Dong et al., 2017; Fu et al., 2017; Hussein et al., 2018]. For the multi-label classification task, we first train the model, then proceed to use the output embeddings as input features to train a one-vs-rest[1] logistic regression classifier[2] using the *scikit-learn* implementations with default

[1] https://scikit-learn.org/stable/modules/generated/sklearn.multiclass.OneVsRestClassifier.html
[2] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

parameters. The training set is randomly sampled from the set of labeled nodes and the rest of nodes are used in the test set. We repeat this procedure 10 times and report the average output. We conduct our experiments using a 80%-20% train/test split.

Similarly, in the case of the node clustering, we use the output embeddings as input vectors to the k-means algorithm [Arthur and Vassilvitskii, 2006]. With the aim of minimizing the effect of local minima that might affect the result of k-means, we run the algorithm with 20 randomized centroid initializations and report the average of the results. Whenever not mentioned, we use the default parameters of the *scikit-learn* implementation of k-means[3].

Finally, for the qualitative node visualization task, we use the same trained embeddings as input features to the t-SNE algorithm [Maaten and Hinton, 2008], with two components and the default settings of the *scikit-learn* implementation[4]. We obtain the two-dimensional representations and plot them, coloring each node based on their label. Note that the output visualization is obtained with no label information whatsoever and has no relation to the clusterings obtained with the k-means algorithm.

To check for statistically significant differences between results for the multi-label classification and node clustering tasks, we verify the assumption of homogeinety of variance using Levene's test [Levene, 1961] and then proceed to use a One-Way analysis of variance (ANOVA) test among variants of the same algorithm (e.g. node2vec, node2vec-Basic-1, node2vec-Centrality, etc.). We then carry out post-hoc analysis using Tukey's test [Tukey, 1949]. Due to the many possibilities of combinations for statistically significant pairings, for each algorithm we use letters to distinguish between groupings of variants that present statistically significant differences in results at the 5% significance level. Each result can be labeled with zero (if it is significantly different from every other result) to one or more letters (if it is assigned to one or more groupings of variants). For instance, if a result is reported with letters bc, it means there is a statistically significant difference between results labeled a, but not ab, ac, b or c. Also, note that while we use the letters abc for every algorithm, they are not related to each other (e.g. some grouping a in the results of JUST has no relation to a grouping a reported for DeepWalk).

4.5 Summary

This chapter restated our research questions and proceeded to introduce the chosen experimental setup aimed at answering them. In Section 4.1, we presented and detailed

[3]https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html
[4]https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html

the three datasets used in our experiments (MOVIE, DBLP and Foursquare), providing information about their schemas, nodes, edges, along with label and degree distributions. Next, in Section 4.2, we introduced the metrics used for the quantitative result analysis of the classification (Micro and Macro F1) and clustering (Normalized Mutual Information) tasks. In Section 4.3, we described the baseline models against which we compare our experimental results. Aside from three standard sampling models (DeepWalk, node2vec and JUST), we also propose two variations of a "naive" undersampling strategy, named Basic-1 and Basic-2. Finally, our choices of hyperparameters, as well as the steps taken when dividing the train, test and validation sets are presented in Section 4.4.

Having defined our experimental setup, in the next chapter we will report and discuss our results, in an attempt to address the research questions (re)stated in this chapter.

Chapter 5

Experimental Results and Analyses

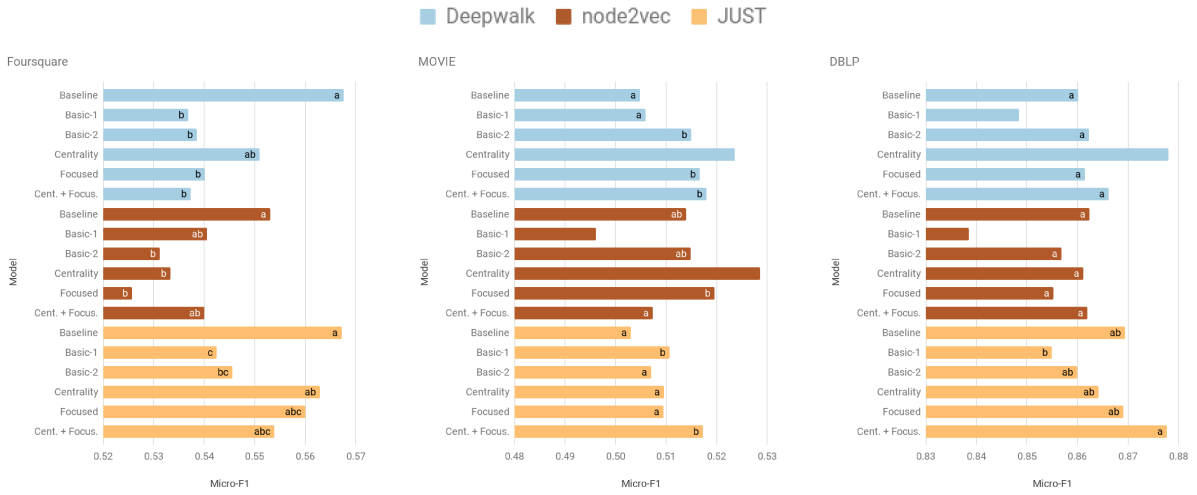
In this chapter, we present and analyze the results of our experiments, outlined in Chapter 4. With this, we aim to answer the research questions stated in the previous chapter. We start by investigating to what degree we are able to improve efficiency without harming effectiveness when reducing the number of training samples in Section 5.1. Next, in Section 5.2, we compare the impact of the sample reduction in terms of the random walk strategies. In Section 5.3, we analyze the results based on node characteristics. Finally, in Section 5.4, we attempt to quantify the degree of redundancy that happens between our sampling variations and the base strategies and understand how these relate to experimental results.

As we would like our embeddings to be versatile in terms of applications, we use the same set of learned embeddings for every task. We present the experimental results for classification in Figure 5.1 and clustering in Figure 5.2. For the complete results in tabular form, refer to Appendix A. As a reminder, we use letters to distinguish between statistically significant results, as described in Section 4.4.2.

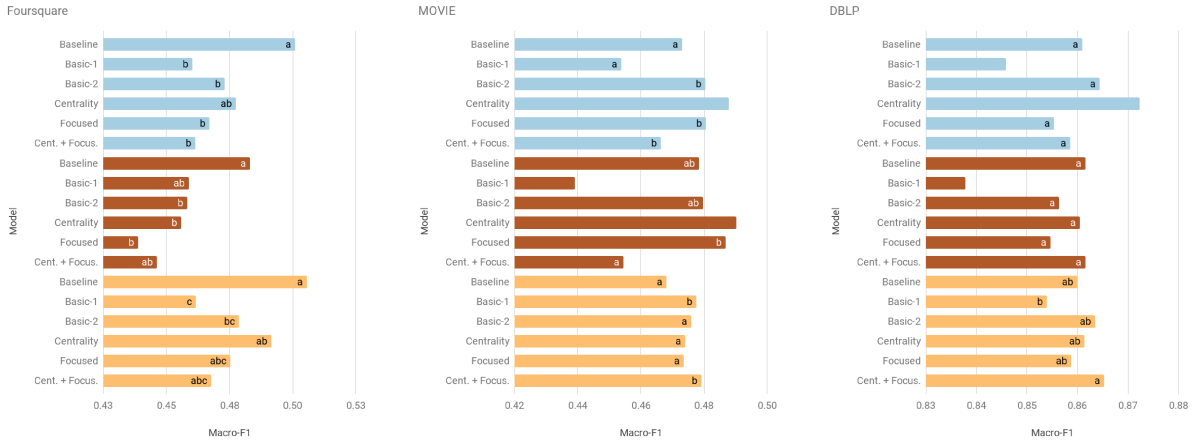
5.1 Efficiency-Effectiveness Tradeoff

The main hypothesis of our work is that we can benefit from principled ways to improve efficiency while retaining or improving effectiveness. To verify this hypothesis and address *RQ1*, we assess our results in terms of the classification and clustering tasks, while considering the reduction in the training sample size used to obtain these results.

In Table 5.1 we present the total number of sampled walks for each model and dataset. We consider the baselines to use 100% of the input sample size since there is no sample reduction attempt in those. Note that each walk contributes to multiple training samples proportionally to the walk length l and the window size k. In our experiments, we fix both l and k, which makes the total number of walks comparable in terms of the number of generated samples.



(a) Micro-F1



(b) Macro-F1

Figure 5.1: Node classification results per dataset.

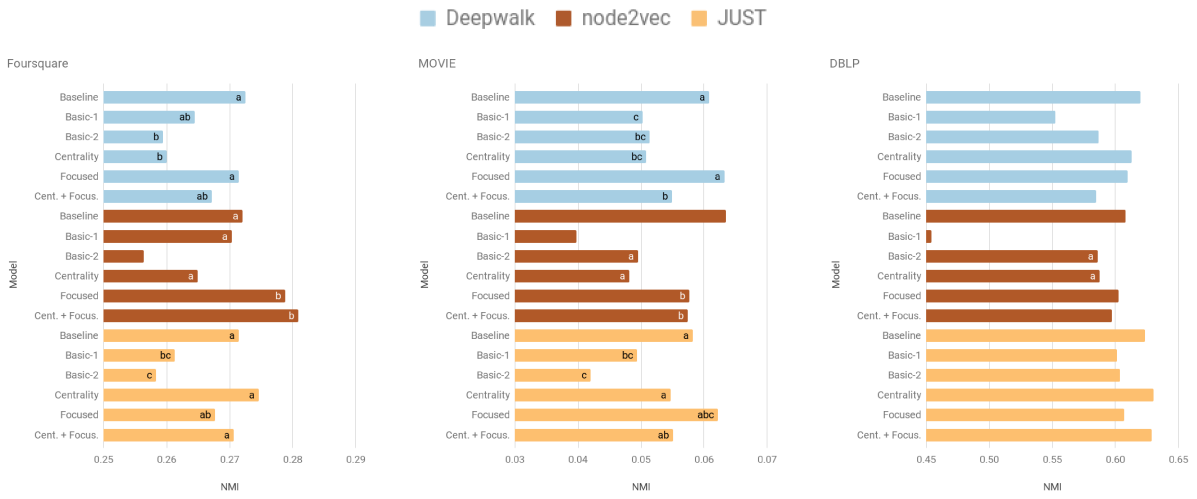


Figure 5.2: Node clustering results per dataset.

Table 5.1: Number of sampled walks per model and dataset.

Dataset	Baseline	Basic-1	Basic-2	Centrality	Focused	Cent. + Focus.
DBLP	156,490 (100%)	15,649 (10%)	31,298 (20%)	37,853 (24.19%)	59,150 (37.8%)	24,897 (15.91%)
Foursquare	297,710 (100%)	29,771 (10%)	59,542 (20%)	36,377 (12.22%)	25,000 (8.4%)	25,038 (8.41%)
MOVIE	228,570 (100%)	22,857 (10%)	45,714 (20%)	46,327 (20.27%)	72,900 (31.89%)	24,296 (10.63%)

Given the input sample reductions, we tackle *RQ1* by assessing which of our strategies achieved statistically significant improvements or at least equivalent results in comparison to the baselines for both classification and clustering. Our aim here is to *consistently* obtain results that are as good or better than the regular models with our strategies that reduce the sample input size. This analysis is presented below in Table 5.2.

Table 5.2: Overview analysis of strategy per task. We compare our results with the baselines and report a checkmark (✓) for each dataset if we observe them to be statistically superior or if there are not statistically significant differences (e.g. statistically, our results are at least as good as the baselines). We use *teal*, *violet* and *brown* for the DBLP, Foursquare and MOVIE datasets, respectively.

Classification			
Algorithm	Centrality	Focused	Cent. + Focus.
DeepWalk	✓ ✓ ✓	✓ ✓	✓
node2vec	✓ ✓	✓ ✓	✓ ✓ ✓
JUST	✓ ✓ ✓	✓ ✓ ✓	✓ ✓ ✓
Clustering			
Algorithm	Centrality	Focused	Cent. + Focus.
DeepWalk		✓ ✓	✓
node2vec	✓	✓	✓
JUST	✓ ✓ ✓	✓ ✓	✓ ✓

From Table 5.2, we note that, in general, the two strategies have been successful, showing significant improvements for every dataset and every algorithm. Breaking down results, for node classification, we observe success in all datasets, especially in MOVIE, whose results benefit from every combination of strategies and algorithms. Similarly, we also note promising results in DBLP, with 8 out of 9 checkmarks (only missing DeepWalk using Centrality + Focused walks). For node clustering, we are not successful in matching baseline results on the DBLP dataset. However, as seen in Figure 5.2, we were able to get very close and, when using the Centrality strategy together with JUST, we actually outperformed the baselines. Results on Foursquare, on the other hand, are balanced on both tasks.

In light of these results, which we managed to obtain by reducing from 24.19% up to 8.41% of the total number of walks when compared to baselines, as seen in Table 5.1, we answer *RQ1* and confirm our hypothesis that it is indeed possible to reduce training samples while consistently learning effective representations and maintaining good results in the evaluated tasks when using all base algorithms. Our results show that the strategies have been successfully applied in different tasks, datasets, and sampling strategies, which helps corroborate their strength and versatility.

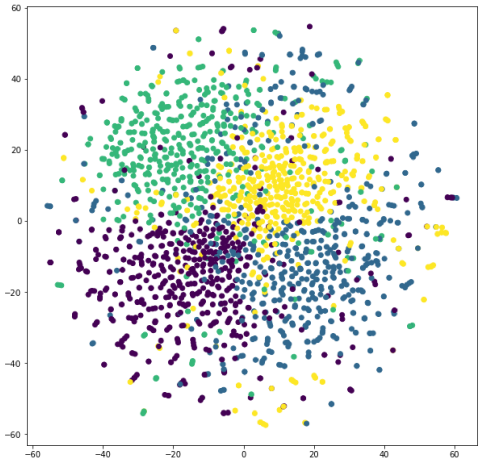
Note that for every algorithm and dataset combination, we are successful in our objective of obtaining results that are at least as good as the baselines while promoting a significant reduction in the number of training samples. While in some cases the Basic strategies also match the results of other strategies/baselines, we argue that since it does not happen regularly and, in addition, sometimes starting a single walk from each node yields better results than starting two walks, using those is unreliable and therefore should not be considered in practice. Instead, we suggest that one or either of our proposed strategies should be adopted.

5.1.1 Node Visualization

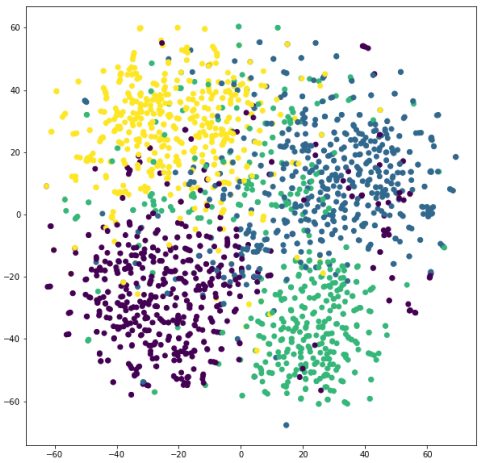
Another useful application for network embeddings is to aid in the 2D visualization of networks. It is often the case that the output representations are used as input to some dimensionality reduction algorithm such as PCA or t-SNE. We now take a look at the efficiency-effectiveness tradeoff by qualitatively addressing the visualization task as a way to corroborate the usage of our strategies from a different perspective. To do so, we used our learned embeddings as input to t-SNE in order to be able to compare the overall visual quality of the output visualizations. We color each node based on its label, which is unknown to the algorithm during training. The results are presented in Figure 5.3.

As seen in the plotted visualizations, it seems that each of our proposed techniques affect the visualizations in a different way. Centrality walks (Figure 5.3.d) do not seem to behave differently from the baselines by itself, whereas Focused walks (Figure 5.3.e) tend to push nodes of different types away from each other.

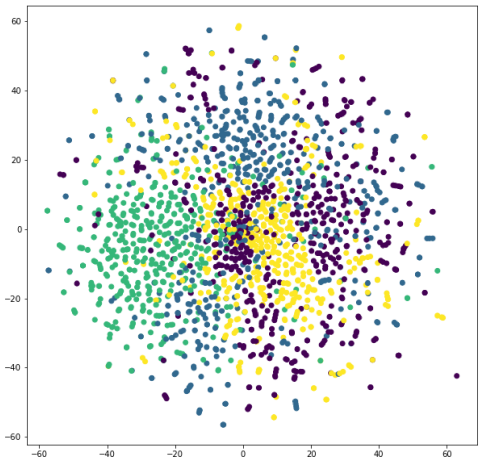
The baseline approaches, in particular DeepWalk and node2vec, manage to separate the classes to a point, like when using the Centrality approach, but there is still a lot of overlapping in the central regions of the plots. By using JUST with both strategies, however, we are able to obtain a clearer separation of author nodes by their areas, as seen in Figure 5.3.f. We posit that both effects listed above are combined to contribute to a better visualization. Note that even though some of these areas are highly correlated,



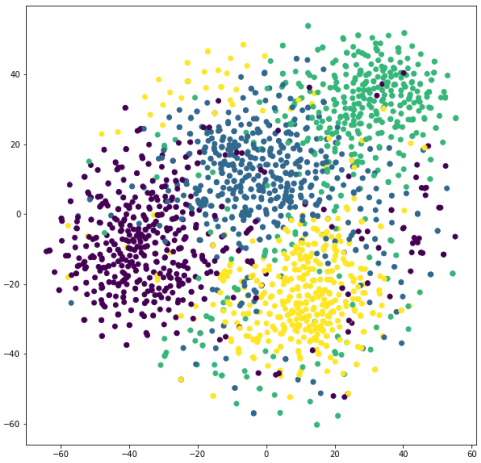
(a) DeepWalk



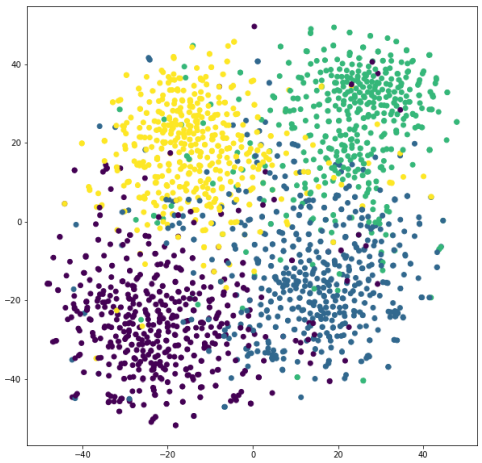
(b) node2vec



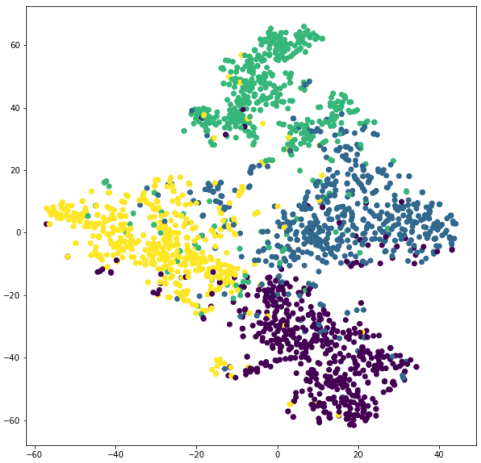
(c) JUST



(d) JUST + Centrality walks



(e) JUST + Focused walks



(f) JUST + Centrality and Focused walks

Figure 5.3: Visualization of DBLP author nodes via t-SNE. Purple: *Database*, blue: *Data Mining*, yellow: *Information Retrieval*, turquoise: *Machine Learning*.

we were still able to obtain a good result while only using 15.91% of the total amount of walks used to train the baselines.

The effects discussed above also happen when using our strategies with DeepWalk and node2vec, but in a less significant way. To keep the comparison clear and easy to interpret, we only report the visualizations obtained by using our strategies with JUST.

Now, we would like to address the other research questions. To do so, in the following sections we look at our findings from a different perspective: we analyse results from a sampling strategy point of view as well as break them down by node degree.

5.2 Impact of Sampling Strategies

In the previous section, we have seen experimental results indicate the possibility of reducing the number of input samples without harming the effectiveness of the models. We now move on to *RQ2*, with the aim of determining the impact of the proposed strategies in relation to the sampling strategies used in this work, which vary in complexity. To this end, we return to the experimental results (Figures 5.1 and 5.2) and analyze them by considering the three sampling techniques: DeepWalk, node2vec and JUST.

Prior to discussing the results, it is worth noting that with regard to Micro-F1 versus Macro-F1 performance, the results happen to be very similar in absolute terms in the case of DBLP (due to the label distribution being balanced), slightly less similar on the MOVIE dataset and very different in the case of Foursquare, which has an unbalanced class distribution. For both metrics, though, the relative results between models and strategies are similar, which help indicate that no class is overly favored over the other in any of the strategies.

According to our experimental results, we observe that, in terms of multi-label classification, the centrality approach seems to lead to a good trade-off between results and sample reduction on all sampling strategies. In particular, for DeepWalk, using centrality by itself is consistently better than the other choices, managing to outperform the baseline in the MOVIE and DBLP datasets, also being the only approach to match the baseline on Foursquare. It is worth noting, though, that using focused walks and pairing the two approches together also yields good results and even more sample economy. On the other hand, when it comes to clustering, the focused approach is the better choice for DeepWalk-based sampling, whereas centrality showed a decrease in NMI on MOVIE and Foursquare. Even though the baseline was slightly better on DBLP, both centrality and focused walks by themselves proved to be good choices for sample reduction while staying quite close to the baseline in terms of effectiveness.

In terms of classification with node2vec, which is more complex than its predecessor, centrality is still the best choice for MOVIE, but we can notice that our choices are not as consistent, since the "naive" approaches seem to perform just as well on the other two datasets. Still, combining both strategies consistently match the baseline performance, which is desirable. As with DeepWalk, using focused walks is better for clustering with node2vec. However, our results show that combining centrality and focused walks is also a good choice for node2vec when clustering, but not for classification.

Finally, JUST seems to benefit more consistently from the combination of both centrality and focused walks than the previous strategies for both classification and clustering, which is a good outcome, since in this case we are able to reduce the number of walks even further. Still, we also note more variation in the behavior of all approaches, with the Basic ones sometimes performing well, but not as consistently as our techniques, but also focused and centrality not being consistent over different datasets. This might happen due to the complexity of JUST-based sampling, which is able to drastically change the walks and that seems to have a bigger effect on our techniques.

In trying to answer *RQ2* we observe that as the complexity of sampling strategies grow, there is a higher degree of fluctuation in results, which sometimes make our techniques present less of an impact. Nevertheless, we posit that our proposed techniques offer a positive contribution to be seen in almost every case, as using them to reduce the required number of samples provide more consistent results than using the naive approaches.

5.3 Impact of Node Properties

Section 5.1 provided evidence that we can consistently improve training efficiency without harming effectiveness, while Section 5.2 showed the impact of different sampling strategies in relation to our approaches. In this section, we tackle *RQ3* and analyze our approaches at the node level. To do so, we make use of the previously trained embeddings, while switching metrics to average accuracy – defined as the ratio of correct predictions in all samples of the test set – which takes the individual contribution of a node and, as such, can be broken down by node properties. That way, we are able to group our results by degree ranges on the classification task. Since clustering metrics evaluate the predictions in terms of groupings of nodes instead of their individual labels, we do not conduct a breakdown analysis on clustering results. We define four categories based on the degree ranges that might arise in real world scenarios:

- ($Degree = 1$): "extreme" cold-start nodes, for which we have little information that might not be reliable;
- ($2 \leq Degree \leq 5$): nodes for which we have some information, but are still seen as cold-start nodes due to their low degrees;
- ($6 \leq Degree \leq 15$): regular nodes with moderate degrees and associated information;
- ($Degree > 15$): high-degree nodes that centralize a lot of information.

We have separated the results for each dataset, such that Figures 5.4, 5.5 and 5.6 present the results for Foursquare, MOVIE and DBLP, respectively. In each of these figures, we also break down the contributions per sampling strategy and present the output accuracy for each model. Since we run the experiments multiple times, we report the average amount of nodes in the test set per range so as to illustrate their contribution in terms of "global" accuracy results. Due to the amount of results to address, we choose to present our discussions based on the degree ranges we have just defined.

When analyzing our first category ("extreme" cold start nodes), we note that for the Foursquare dataset we have a low amount of 1-degree nodes (with an average of 1.2 per test set), which makes the data unreliable. As such, we focus our analysis on the other two datasets. We observe that, when it comes to cold start nodes, the baseline models are a solid choice, consistently yielding good accuracy results. It is worth noting, however, that both of our strategies seem to work very well for these nodes, with the focused strategy being particularly good for the MOVIE dataset, whose amount of 1-degree nodes is considerably lower in relation to other degree ranges, which gives us a boost in accuracy by focusing on nodes that are deemed unimportant from a centrality point of view. On the other hand, in the case of DBLP, the amount of cold start nodes is bigger in comparison to the overall graph, which leads to them having a bigger impact in temrs of centrality and, as such, it turns out that using the centrality approach is better than the focused one in this case, and pairing both is even better. This leads us to consider that, even if we break down nodes by their degrees, we must also take their relation to the global graph structure when analyzing the impact of our strategies.

Albeit slightly bigger, the average node amount for the 2-5 range is still too small for the Foursquare dataset, as in the 1-degree case. Thus, we keep our attention on DBLP and MOVIE for this degree category as well. In this case, we see that the impact of centrality walks improve on the MOVIE dataset. We posit that this happens due to the increase in the proportion of nodes for this bracket. The same effect happens when we look at the DBLP results, where we also see a big amount of nodes on average in this bracket, which corroborates the idea that centrality walks tend to perform better when we have more nodes of that "centrality range" in the graph. Focused walks lose a bit in

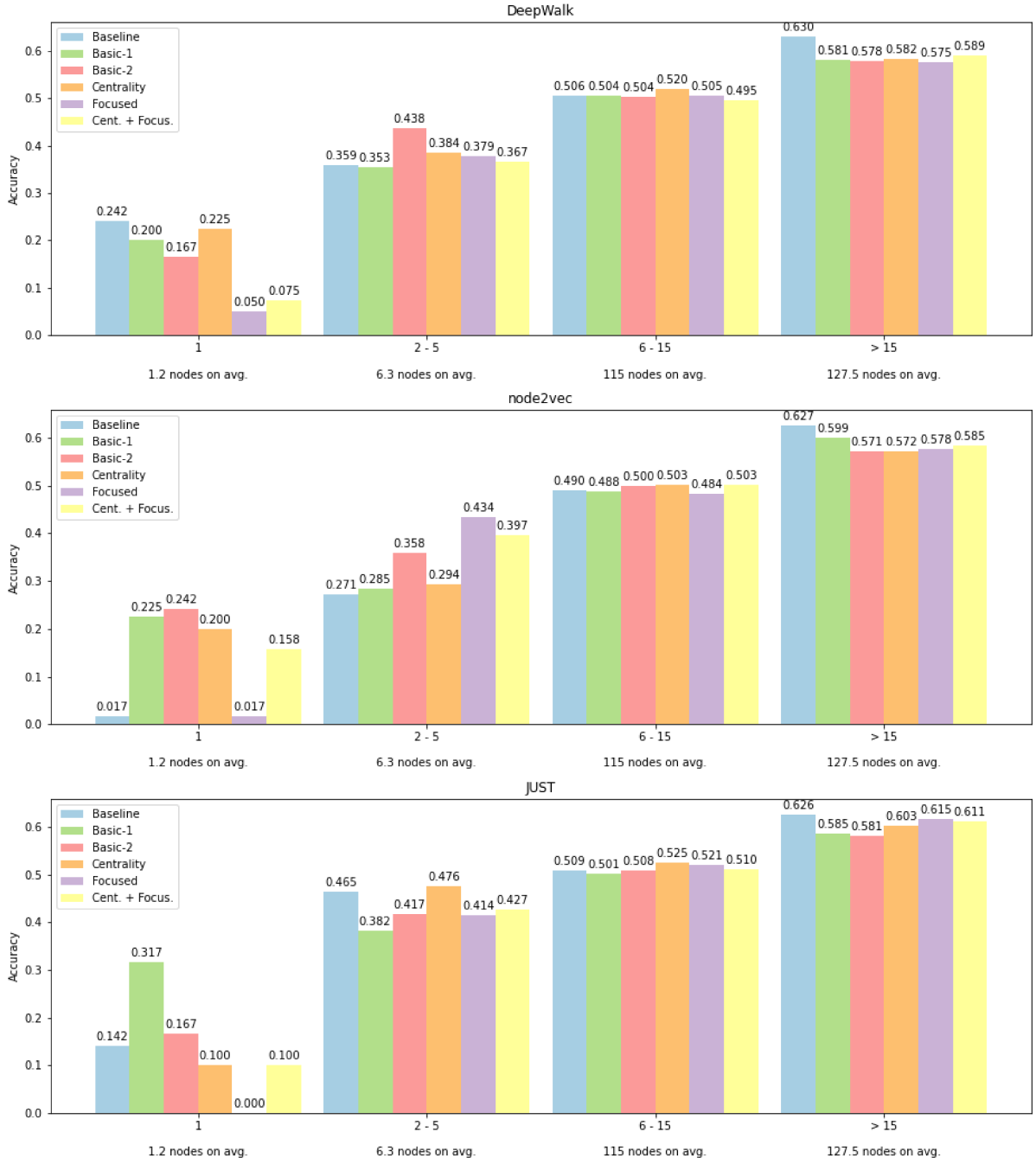


Figure 5.4: Accuracy breakdown by node degree ranges for the Foursquare dataset.

terms of accuracy in this range, but combining the strategies seem to be a good approach, as we obtain a good tradeoff by taking advantage of the centrality effect to keep accuracy high while reducing walks with both strategies, which is the best outcome in terms of efficacy in this case.

Moving on to the 6-15 degree interval, we now start looking at the Foursquare dataset along with the other two. The reduced amount of nodes within this category in the DBLP dataset makes it so that the Basic-2 baseline performs well on it with node2vec

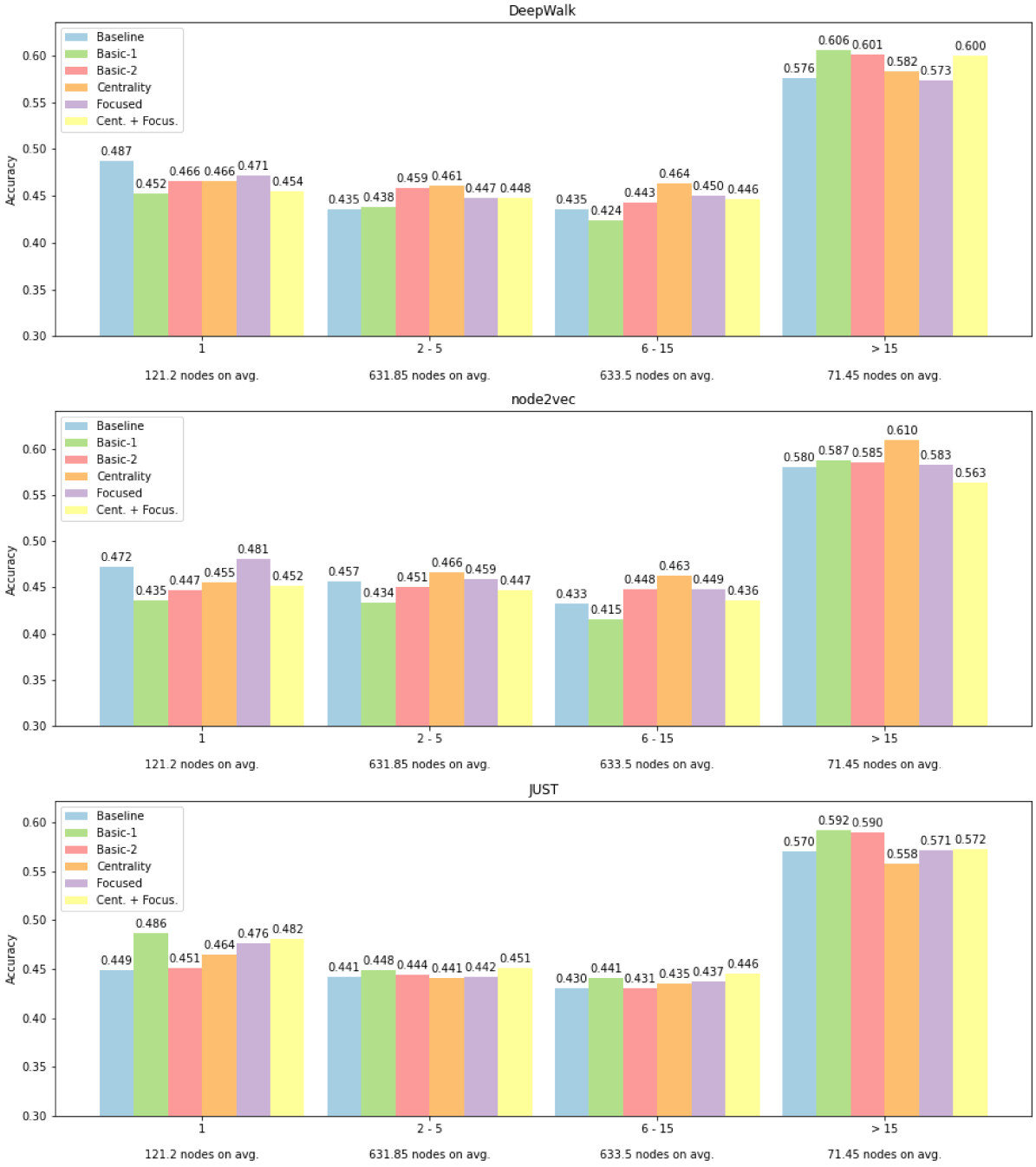


Figure 5.5: Accuracy breakdown by node degree ranges for the MOVIE dataset.

and JUST. On the other two datasets, however, whose amount of nodes within this range is significantly bigger – yielding more trustworthy results – we notice that centrality works well in this range. We think this might be the case due to the proportion hypothesis mentioned in the previous analyses, as we have the highest proportion in the 6-15 bracket on MOVIE and the second highest on Foursquare. As in the case of the previous interval, the combination of both strategies seem to be a fine compromise between effectiveness and efficacy when we have enough nodes within that particular degree range.

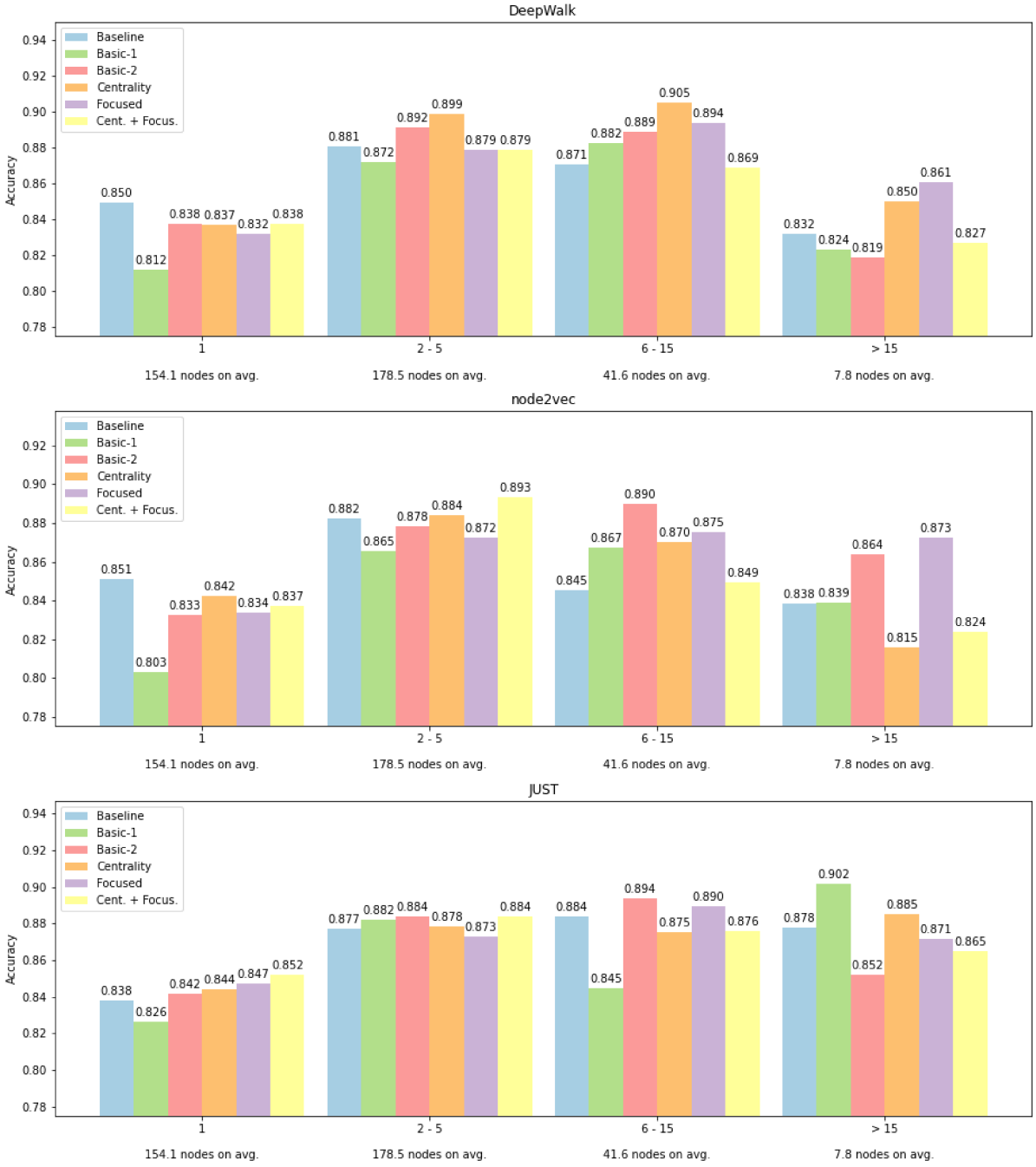


Figure 5.6: Accuracy breakdown by node degree ranges for the DBLP dataset.

Finally, for the high degree bracket, we forgo the analysis on DBLP due to the low amount of average nodes on this interval for this dataset. We also note that this bracket is the least representative for MOVIE as well, which might help explain why the results fluctuate so much in this case. On Foursquare, we see that in the case of high degree nodes, the baseline is the absolute best and that it is better to combine both strategies when trying to reduce the number of sampled walks. In an attempt to understand what makes this interval different from the others, we make two observations. First, it has

no upper bound in terms of node degree, which might make its results vary significantly within the bracket. To better understand this effect, we would need to define additional categories depending on the node distribution on a graph-by-graph basis. Such specificity would certainly be interesting, but it does not fall within the scope of our study. Secondly, we hypothesize that the high degree of these nodes make it so that our strategies could be insufficient in understanding their complex neighborhoods, which is better achieved by thoroughly sampling walks on the whole graph as done by the baselines. While we cannot explain this effect with certainty, we believe that it suggests that different centrality intervals would profit by making use of different embeddings. While this idea would go against our efficacy requirement, as it would involve training multiple models on the whole graph, it could be an interesting future research direction if one is not limited by computing constraints.

In summary, when addressing RQ3 we note that, in terms of consistency, we obtain our best results from the mid categories 2-5 and 6-15. Those also happen to be representative in all datasets (2-5 for DBLP and MOVIE and 6-15 for Foursquare and MOVIE), which helps explain why we sometimes outperform other models when looking at our general classification metrics. In future work, we would like to draw inspiration from these breakdown analyses and design hybrid approaches that behave differently by changing the sampling strategy and/or the window size based on the centrality of a node.

5.4 Sample Redundancy Analysis

With *RQ4*, we would like to investigate the hypothesis that there are redundant samples being generated by the base models. In this analysis, we consider that a base model generates redundant samples if we are able to obtain a high fraction of the same node pairs while using a smaller quantity of random walks, as in our approaches.

Thus, with the aim of comparing our samples to their corresponding baselines, we analyze the walks generated by the models during the experiments. Given the walks for each model, we generate node pairs following the Skip-gram algorithm. However, due to the combinatorial explosion of big window sizes, we limit our analysis to node pairs generated with a window of size 2 (in our experiments, we use a window of size 10).

In order to quantify the degree of redundancy, we compute the Jaccard coefficient (Eq. (5.1)) between the set of node pairs from the base model (A) and from the different

models (B) we are using.

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{5.1}$$

We report the average value of 5 different generated walks for each strategy and dataset. The results are presented in Table 5.3.

Table 5.3: Comparison of Jaccard coefficient of node pair sets between each proposed strategy and the base sampling model for each dataset.

DBLP					
Base model	Basic-1	Basic-2	Centrality	Focused	Cent. + Focus.
DeepWalk	0.3884	0.4879	0.5136	0.5681	0.3026
node2vec	0.3736	0.4719	0.4975	0.5469	0.3474
JUST	0.3493	0.4527	0.4839	0.5520	0.2728
MOVIE					
Base model	Basic-1	Basic-2	Centrality	Focused	Cent. + Focus.
DeepWalk	0.5069	0.6360	0.6491	0.7046	0.4979
node2vec	0.4830	0.6100	0.6252	0.6778	0.4876
JUST	0.6155	0.6962	0.7064	0.7356	0.6066
Foursquare					
Base model	Basic-1	Basic-2	Centrality	Focused	Cent. + Focus.
DeepWalk	0.2993	0.4250	0.3327	0.1767	0.2661
node2vec	0.2725	0.3863	0.3027	0.1613	0.2419
JUST	0.2912	0.4099	0.3230	0.2215	0.2607

We note from the results that, for DBLP and MOVIE, our strategies exhibit a higher degree of sample similarity in comparison to the baselines, while the opposite happens in the case of Foursquare, which is different in structure from the other datasets by both its *Check-in* and *Timestamp* node types, which do not follow regular centrality distributions, as seen in Figure 4.1. Still, while in some cases there exists a high degree of redundancy, as in the case of MOVIE, where we have a Jaccard coefficient above 70% with our strategies, these results are not consistently correlated with the performance observed in our experimental results. For instance, combining our strategies, which proves to be a good choice experimentally, as seen in the previous sections, often presents the lowest Jaccard coefficient here.

With this in mind, we have tried taking the amount of repeated samples into account when defining our idea of redundancy so as to look for additional answers. To do so, we built vectors of node pair frequencies (e.g. each dimension is associated with a pair of node ids, with the vector dimension being the number of unique node pairs) and computed the cosine similarity between these vectors. Since we obtained a similarity above 0.9 for every comparison, we opt to not report the results individually here. We

found out that this effect happens due to high-frequency node pairs – which show up in all strategies – contributing significantly to the cosine similarity computation. This leads us to believe that the difference in results lies in how each strategy handles lower frequency pairs.

As such, in trying to address *RQ4*, we argue that even though we were able to find signs of sample redundancy, we still need to investigate further before drawing further conclusions. Our analysis indicates that lower frequency node pairs might be a good place to look for answers and we leave these considerations to future work. We believe that a thorough comprehension of these samples and their relation to the training process can provide valuable information and help develop better sampling techniques.

5.5 Summary

In this chapter, we reported and discussed the experimental results which helped us in validating our proposed ideas.

In Section 5.1, we addressed our first research question, concerning the tradeoff between efficiency and effectiveness. We verified that we were able to achieve statistically equivalent or better effectiveness in most experiments with varied amounts of sample reduction, as well as provided a qualitative analysis in the form of a visualization experiment, showing that we can drastically reduce the number of sampled walks while sacrificing little predictive power (or none whatsoever).

In Section 5.2, we answered our second research question by analyzing experimental results from the point of view of different sampling strategies. We have presented results for classification and clustering and shown that our techniques work in most cases, while showing a higher degree of fluctuation in results as the complexity of sampling strategies increases.

In Section 5.3, we observed how node degrees impact our strategies by breaking down our experimental results by node ranges, addressing our third research question. We saw that our approaches are more consistent in terms of results within intermediate degree ranges and suggested ways how hybrid strategies could take advantage of this behavior.

Finally, in Section 5.4, we addressed our last research question by providing an analysis of sample redundancy. We compared the samples generated by the base models with our strategies and discussed how further analyses could possibly approach the redundancy problem so as to better characterize this phenomenon.

We conclude the experimentation and analyses of our proposed sample reduction strategies and corresponding research questions in this chapter. In the next chapter, we

review the contributions of this work and discuss possibilities for future work.

Chapter 6

Conclusions and Future Work

With the ever growing usage and size of graph structured data, network representation learning techniques have become commonplace both in industry and academia due to their strength in capturing complex information in a compact data structure. Because of that, the need for scalable algorithms is paramount for many real world applications and, as such, every contribution toward efficiency proves to be extremely important in present times. In light of this, we proposed two sample reduction strategies for heterogeneous network embedding, which is known to be a complex task (and sometimes computationally costly) due to the multiple types of information inherent to its structure. We also proposed the organization of random walk-based models into separate customizable steps, with the aim of providing an easy way to experiment and come up with novel approaches for future applications.

The following sections summarize our contributions and findings, present our concluding remarks and provide directions for future research.

6.1 Summary of Contributions

In this section, we summarize the main contributions of this work.

1. In Chapter 3, we generalized centrality-based walks for heterogeneous networks, contributing with a way to capture centrality information for networks of any type and degree of heterogeneity. We also proposed focused walks, a task-based strategy that defines a subset of starting node types. The first technique is useful when we want to concentrate our attention on nodes following some predefined measure of centrality per node type, whereas the latter is particularly effective when we want to learn embeddings for node types that are underrepresented in our graph;

2. Also in Chapter 3, we framed the aforementioned centrality and focused walks as part of a customizable random walk module that allows for custom choices of the set

of starting nodes and the function that defines the number of walks starting from a given node. In doing so, we offer flexibility in the design of sampling strategies and make it easy to extend our work;

3. In Chapter 5, we thoroughly validated the proposed strategies experimentally and analyzed to which degree they could be used as sample reduction alternatives for random walk-based heterogeneous network embedding techniques from both quantitative and qualitative points of view. We also evaluated the relation of our strategies to different types of sampling approaches and their effectiveness in learning node representations of different degree ranges. Finally, we attempted to quantify the degree of sample redundancy present in regular sampling approaches.

6.2 Summary of Conclusions

In this work, we presented two sample reduction strategies for heterogeneous network embedding, along with the generalization of certain steps in the learning framework that allow for further customization. Our approaches are based on using centrality measures to define the number of walks starting from each node in the network and making use of the heterogeneous type information to define a subset of starting nodes. While these were proposed in the context of this work, we encourage further experimentation with the many possible ways to define the number of starting walks and the starting subset. Extensive experimentation on multiple publicly available datasets demonstrated the effectiveness of our strategies in reducing the required number of samples to obtain good embeddings, which helps pave the way for their usage in the context of regular network embedding methods, as well as being part of more complex techniques.

6.3 Future Work

We believe our strategies to be generally useful for network representation learning. In order to confirm this hypothesis, we must evaluate them on various methods and approaches. As such, in the future, we aim to experiment with the proposed strategies on other types of models that make use of random walks, either directly or indirectly, such as GraphSAGE [Hamilton et al., 2017a].

Another direction for future work is to experiment with different ways to define $r(\cdot)$ and U. As previously mentioned, there are many possibilities when using centrality information to define these. One might even focus on ways that do not make use of centrality, or define different centrality measures depending of the node type and its degree distribution or overall presence in the graph. Another possible approach would be to propose a hybrid method that makes use of our methods along with novel ideas. As seen in Section 5.3, results seem to vary based on the node degree, which could point that defining a set of functions to be chosen depending on the centrality of a node would improve further results.

Finally, we would like to pursue a theoretically principled way to quantify and answer the following research question: up to which point can we use our strategies without affecting the generalization properties of the base algorithms? In particular, we believe that a possible answer might be obtained by trying and relating our techniques to known bounds for sample complexity in representation learning [Seibert, 2019]. In general, results in learning theory would be a good place to start looking for insights regarding this question.

Bibliography

Aggarwal, C. C. et al. (2016). *Recommender systems*. Springer.

Arthur, D. and Vassilvitskii, S. (2006). k-means++: The advantages of careful seeding. Technical report, Stanford.

Barabási, A.-L. and Albert, R. (1999). Emergence of scaling in random networks. *science*, 286(5439):509--512.

Barabási, A.-L. et al. (2016). *Network science*. Cambridge university press.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. (2018). Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*.

Bonacich, P. (1987). Power and centrality: A family of measures. *American journal of sociology*, 92(5):1170--1182.

Bondy, J. A., Murty, U. S. R., et al. (1976). *Graph theory with applications*, volume 290. Macmillan London.

Brandes, U. and Pich, C. (2007). Centrality estimation in large networks. *International Journal of Bifurcation and Chaos*, 17(07):2303--2318.

Brouwer, A. E. and Haemers, W. H. (2011). *Spectra of graphs*. Springer Science & Business Media.

Cai, H., Zheng, V. W., and Chang, K. C.-C. (2018). A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering*, 30(9):1616--1637.

Chen, H., Yin, H., Chen, T., Nguyen, Q. V. H., Peng, W.-C., and Li, X. (2019). Exploiting centrality information with graph convolutions for network representation learning. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 590--601. IEEE.

Chen, T. and Sun, Y. (2017). Task-guided and path-augmented heterogeneous network embedding for author identification. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 295--304.

Chiang, W.-L., Liu, X., Si, S., Li, Y., Bengio, S., and Hsieh, C.-J. (2019). Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 257--266.

Chinchor, N. (1992). Muc-4 evaluation metrics in proc. of the fourth message understanding conference 22–29.

Chung, F. R. and Graham, F. C. (1997). *Spectral graph theory*. Number 92. American Mathematical Soc.

Cui, P., Wang, X., Pei, J., and Zhu, W. (2018). A survey on network embedding. *IEEE Transactions on Knowledge and Data Engineering*, 31(5):833--852.

Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., Ranzato, M., Senior, A., Tucker, P., Yang, K., et al. (2012). Large scale distributed deep networks. In *Advances in neural information processing systems*, pages 1223--1231.

Dong, Y., Chawla, N. V., and Swami, A. (2017). metapath2vec: Scalable representation learning for heterogeneous networks. pages 135--144. ACM.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7).

Fu, T.-y., Lee, W.-C., and Lei, Z. (2017). Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1797--1806. ACM.

Gao, M., Chen, L., He, X., and Zhou, A. (2018). Bine: Bipartite network embedding. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 715--724.

Goldberg, Y. and Levy, O. (2014). word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*.

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

Gutmann, M. U. and Hyvärinen, A. (2012). Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *The journal of machine learning research*, 13(1):307--361.

Hamilton, W., Ying, Z., and Leskovec, J. (2017a). Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024--1034.

Hamilton, W. L., Ying, R., and Leskovec, J. (2017b). Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*.

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

Hussein, R., Yang, D., and Cudré-Mauroux, P. (2018). Are meta-paths necessary?: Revisiting heterogeneous graph embeddings. pages 437--446. ACM.

Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Kleinberg, J. M. (1999). Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604--632.

Kvalseth, T. O. (1987). Entropy and correlation: Some comments. *IEEE Transactions on Systems, Man, and Cybernetics*, 17(3):517--519.

Lai, Y.-A., Hsu, C.-C., Chen, W. H., Yeh, M.-Y., and Lin, S.-D. (2017). Prune: Preserving proximity and global ranking for network embedding. In *Advances in neural information processing systems*, pages 5257--5266.

Levene, H. (1961). Robust tests for equality of variances. *Contributions to probability and statistics. Essays in honor of Harold Hotelling*, pages 279--292.

Li, C.-K. and Zhang, S. (2015). Stationary probability vectors of higher-order markov chains. *Linear Algebra and its Applications*, 473:114--125.

Lovász, L. et al. (1993). Random walks on graphs: A survey. *Combinatorics, Paul erdos is eighty*, 2(1):1--46.

Ma, Y., Wang, S., Ren, Z., Yin, D., and Tang, J. (2017). Preserving local and global information for network embedding. *arXiv preprint arXiv:1710.07266*.

Maaten, L. v. d. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579--2605.

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

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

Moore, M. (2016). Tech giants and civic power. *Centre for the study of Media, Communication & Power, King's College London. Retrieved February*, 5:2020.

Morin, F. and Bengio, Y. (2005). Hierarchical probabilistic neural network language model. In *Aistats*, volume 5, pages 246--252. Citeseer.

Norris, J. R. (1998). *Markov chains*. Number 2. Cambridge university press.

Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab.

Perozzi, B., Al-Rfou, R., and Skiena, S. (2014). Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701--710. ACM.

Rossi, R. A., Ahmed, N. K., Carranza, A., Arbour, D., Rao, A., Kim, S., and Koh, E. (2019). Heterogeneous network motifs. *arXiv preprint arXiv:1901.10026*.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1985). Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science.

Seibert, M. (2019). *Sample Complexity of Representation Learning for Sparse and Related Data Models*. PhD dissertation, Technische Universität München.

Shang, J., Qu, M., Liu, J., Kaplan, L. M., Han, J., and Peng, J. (2016). Meta-path guided embedding for similarity search in large-scale heterogeneous information networks. *arXiv preprint arXiv:1610.09769*.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell system technical journal*, 27(3):379--423.

Shi, C., Hu, B., Zhao, W. X., and Philip, S. Y. (2019). Heterogeneous information network embedding for recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 31(2):357--370.

Shi, C., Li, Y., Zhang, J., Sun, Y., and Philip, S. Y. (2016). A survey of heterogeneous information network analysis. *IEEE Transactions on Knowledge and Data Engineering*, 29(1):17--37.

Small, M. (2013). Complex networks from time series: Capturing dynamics. In *2013 IEEE International Symposium on Circuits and Systems (ISCAS2013)*, pages 2509--2512. IEEE.

Suchanek, F. M., Kasneci, G., and Weikum, G. (2007). Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697--706.

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

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., and Mei, Q. (2015). Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pages 1067--1077.

Tang, L. and Liu, H. (2009a). Relational learning via latent social dimensions. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 817--826.

Tang, L. and Liu, H. (2009b). 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.

Tukey, J. W. (1949). Comparing individual means in the analysis of variance. *Biometrics*, pages 99--114.

Valdeolivas, A., Tichit, L., Navarro, C., Perrin, S., Odelin, G., Levy, N., Cau, P., Remy, E., and Baudot, A. (2019). Random walk with restart on multiplex and heterogeneous biological networks. *Bioinformatics*, 35(3):497--505.

Van Rijsbergen, C. J. (1979). Information retrieval. 2nd. newton, ma.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Philip, S. Y. (2020). A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*.

Yang, D., Zhang, D., Chen, L., and Qu, B. (2015). Nationtelescope: Monitoring and visualizing large-scale collective behavior in lbsns. *Journal of Network and Computer Applications*, 55:170--180.

Yang, D., Zhang, D., and Qu, B. (2016). Participatory cultural mapping based on collective behavior data in location-based social networks. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 7(3):1--23.

Yang, Y. and Liu, X. (1999). A re-examination of text categorization methods. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 42--49.

Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., and Leskovec, J. (2018). Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 974--983.

Zhang, D., Yin, J., Zhu, X., and Zhang, C. (2018). Network representation learning: A survey. *IEEE Transactions on Big Data*.

Appendix A

Experimental Results

Table A.1: Comparison of Micro-F1 results for node classification per model and dataset.

Foursquare						
Model	Baseline	Basic-1	Basic-2	Centrality	Focused	Cent. + Focus.
DeepWalk	**0.5676**^{a}	0.5368^{b}	0.5385^{b}	**0.5509**^{ab}	0.5401^{b}	0.5373^{b}
node2vec	**0.5560**^{a}	0.5284^{ab}	0.5384^{b}	0.53^{b}	0.5356^{b}	**0.5464**^{ab}
JUST	**0.5672**^{a}	0.5424^{c}	0.5456^{bc}	**0.5629**^{ab}	**0.56**^{a}	**0.5539**^{abc}
# Walks	297710 (100%)	29771 (10%)	59542 (20%)	36377 (12.22%)	25000 (8.4%)	25038 (8.41%)
MOVIE						
Model	Baseline	Basic-1	Basic-2	Centrality	Focused	Cent. + Focus.
DeepWalk	0.5048^{a}	0.5059^{a}	0.5149^{b}	**0.5236**	0.5167^{b}	0.518^{b}
node2vec	0.514^{ab}	0.4961	0.5148^{ab}	**0.5287**	0.5196^{b}	0.5073^{a}
JUST	0.503^{a}	**0.5107**^{b}	0.5071^{a}	0.5096^{a}	0.5094^{a}	**0.5173**^{b}
# Walks	228570 (100%)	22857 (10%)	45714 (20%)	46327 (20.27%)	72900 (31.89%)	24296 (10.63%)
DBLP						
Model	Baseline	Basic-1	Basic-2	Centrality	Focused	Cent. + Focus.
DeepWalk	0.8602^{a}	0.8484	0.8623^{a}	**0.878**	0.8615^{a}	0.8662^{a}
node2vec	**0.8624**^{a}	0.8385	**0.8569**^{a}	**0.8612**^{a}	**0.8552**^{a}	**0.862**^{a}
JUST	**0.8694**^{ab}	0.855^{b}	**0.8599**^{ab}	**0.8641**^{ab}	**0.8691**^{ab}	**0.8777**^{a}
# Walks	156490 (100%)	15649 (10%)	31298 (20%)	37853 (24.19%)	59150 (37.8%)	24897 (15.91%)

Table A.2: Comparison of Macro-F1 results for node classification per model and dataset.

Foursquare						
Model	Baseline	Basic-1	Basic-2	Centrality	Focused	Cent. + Focus.
DeepWalk	**0.5009**	0.4604^{a}	0.4730^{a}	0.4775^{a}	0.4671^{a}	0.4615^{a}
node2vec	**0.4830**	0.4589^{a}	0.4583^{a}	0.4559^{a}	0.4389^{b}	0.4463^{ab}
JUST	$\mathbf{0.5055}^{a}$	0.4616^{c}	0.4788^{bc}	$\mathbf{0.4915}^{ab}$	0.4753^{bc}	0.4678^{c}
# Walks	297710 (100%)	29771 (10%)	59542 (20%)	36377 (12.22%)	25000 (8.4%)	25038 (8.41%)
MOVIE						
Model	Baseline	Basic-1	Basic-2	Centrality	Focused	Cent. + Focus.
DeepWalk	0.4731^{ab}	0.4538	$\mathbf{0.4805}^{bc}$	$\mathbf{0.4878}^{c}$	$\mathbf{0.4806}^{bc}$	0.4663^{a}
node2vec	0.4784^{a}	0.4391	0.4797^{ba}	$\mathbf{0.4902}^{c}$	$\mathbf{0.4869}^{bc}$	0.4545
JUST	0.4682^{a}	$\mathbf{0.4776}^{b}$	0.476^{ab}	0.474^{ab}	0.4736^{ab}	$\mathbf{0.4792}^{b}$
# Walks	228570 (100%)	22857 (10%)	45714 (20%)	46327 (20.27%)	72900 (31.89%)	24296 (10.63%)
DBLP						
Model	Baseline	Basic-1	Basic-2	Centrality	Focused	Cent. + Focus.
DeepWalk	0.8610^{a}	0.8459^{b}	$\mathbf{0.8643}^{ac}$	$\mathbf{0.8723}^{c}$	0.8553^{ab}	0.8586^{a}
node2vec	$\mathbf{0.8616}^{a}$	0.8378	$\mathbf{0.8564}^{a}$	$\mathbf{0.8605}^{a}$	$\mathbf{0.8546}^{a}$	$\mathbf{0.8616}^{a}$
JUST	0.86^{a}	0.8539^{a}	0.8635^{a}	0.8613^{a}	0.8587^{a}	0.8653^{a}
# Walks	156490 (100%)	15649 (10%)	31298 (20%)	37853 (24.19%)	59150 (37.8%)	24897 (15.91%)

Table A.3: Comparison of Normalized Mutual Information results for node clustering per model and dataset.

Foursquare						
Model	Baseline	Basic-1	Basic-2	Centrality	Focused	Cent. + Focus.
DeepWalk	**0.2725**^{a}	**0.2644**^{ab}	0.2594^{b}	0.26^{b}	**0.2715**^{a}	**0.2672**^{ab}
node2vec	0.272^{a}	0.2703^{a}	0.2564	0.2649^{a}	**0.2788**^{b}	**0.2809**^{b}
JUST	**0.2714**^{a}	0.2613^{bc}	0.2583^{c}	**0.2746**^{a}	**0.2676**^{ab}	**0.2707**^{a}
# Walks	297710 (100%)	29771 (10%)	59542 (20%)	36377 (12.22%)	25000 (8.4%)	25038 (8.41%)
MOVIE						
Model	Baseline	Basic-1	Basic-2	Centrality	Focused	Cent. + Focus.
DeepWalk	**0.0608**^{a}	0.0503^{c}	0.0514^{bc}	0.0508^{bc}	**0.0632**^{a}	0.0549^{b}
node2vec	**0.0634**	0.0397	0.0495^{a}	0.0481^{a}	0.0577^{b}	0.0574^{b}
JUST	**0.0582**^{a}	0.0493^{bc}	0.0420^{c}	**0.0547**^{a}	**0.0622**^{abc}	**0.0551**^{ab}
# Walks	228570 (100%)	22857 (10%)	45714 (20%)	46327 (20.27%)	72900 (31.89%)	24296 (10.63%)
DBLP						
Model	Baseline	Basic-1	Basic-2	Centrality	Focused	Cent. + Focus.
DeepWalk	**0.6196**	0.5525	0.5869	0.6128	0.6097	0.5848
node2vec	**0.6081**	0.4540	0.5858^{a}	0.5874^{a}	0.6025	0.5972
JUST	0.6236	0.6013	0.6035	**0.6302**	0.6069	0.6289
# Walks	156490 (100%)	15649 (10%)	31298 (20%)	37853 (24.19%)	59150 (37.8%)	24897 (15.91%)