SYSTEMATIC EVALUATION OF GRAPH SAMPLING METHODS

by

Jasper Forth

under the supervision of

Prof. Dr. Holger Dell

and

Dennis Vetter

A document submitted in partial fulfillment of the requirements for the degree

Bachelor of Science

at

Goethe University Frankfurt

Abstract

In this work, we comprehensively evaluate the *node2vec* and the *CrossWalk* random walk-based sampling for generating graph embeddings of social networks. To this end, we have investigated various combinations of several *hyperparameter* configurations to generate different embeddings for various datasets. For selecting the underlying datasets, we defined subsets of a social network in terms of certain features. The resulting embeddings were systematically examined concerning their *fairness* and accuracy. For this purpose, we used a semi-supervised node classification as a downstream task to evaluate the quality of the embeddings generated by different sampling methods and the trade-off between accuracy and fairness. It was shown that the configuration of the *hyperparameters* of *node2vec* and *CrossWalk* significantly affects the resulting graph representations in both directions and thus either increases or decreases the prediction accuracy or the *fairness* for selected features.

Zusammenfassung

Die vorliegende Arbeit ist eine umfassende Bewertung von *node2vec* und *CrossWalk* und einhergehenden modifizierten *Random-Walk*-Strategien zur Erzeugung von Graph-Einbettungen sozialer Netzwerke. Hierzu haben wir verschiedene Kombinationen von unterschiedlichen *Hyperparameter*-Konfigurationen untersucht, mit dem Ziel, verschiedene Einbettungen für mehrere unterschiedliche Datensätze zu erzeugen. Für die Auswahl der zugrundeliegenden Datensätze wurden Teilmengen eines sozialen Netzwerks hinsichtlich bestimmter Merkmale definiert. Die resultierenden Einbettungen wurden systematisch hinsichtlich ihrer *Fairness* und ihrer Genauigkeit untersucht. Zu diesem Zweck haben wir als nachgelagerte Aufgabe eine *semi-supervised* Knotenklassifikation eingesetzt, um so Qualität der durch verschiedene *Sampling-Methoden* erzeugten Einbettungen und den Zielkonflikt zwischen Genauigkeit und Fairness zu bewerten. Es konnte gezeigt werden, dass die Konfiguration der *Hyperparameter* von *node2vec* und *CrossWalk* die resultierenden Graph-Einbettungen in beide Richtungen signifikant beeinflusst und somit entweder die Vorhersagegenauigkeit oder die *Fairness* in Bezug auf bestimmte Merkmale deutlich verbessern.

Contents

List of Figures

List of Tables

1 Introduction

In any development regarding human-aligned artificial intelligence (*AI*) systems in general and machine learning models in particular, fairness should be of essential concern to ensure robust and secure systems. Whether it is to comply with legal regulations, meet users' demands, or increase the acceptance of AI systems. Concerning AI-related tasks for social networks, two questions are of particular interest:

(1) How can such networks best be represented by computers, i.e., using the rich and highly complex structural information of such networks in the most effective way?

(2) How can we ensure that for the automated predictions and recommendations based on such complex structures, the machine learning algorithms do not reveal specific sensitive information or discriminate against individual users based on such sensitive information?

Recently, with the growing interest regarding fairness in machine learning, some solutions have been proposed to address the fairness issue in network representations. For the application of machine learning tasks, as part of *AI* systems, on network structures, it is necessary to somehow represent such network features with reduced complexity. With this representation, it is essential to convey as much accurate and relevant information as possible from the network. Regarding their quality, these network representations require to be evaluated in terms of both asked questions, accordingly, for (1) accuracy and (2) fairness.

This work comprehensively evaluates the *node2vec* and *CrossWalk* random walk biasing methods for generating node embeddings. Therefore, we searched through various combinations of multiple hyperparameters to generate multiple embeddings on various datasets. We selected the underlying datasets from a real-world social network based on specific graph features. We evaluate the resulting embeddings systematically in terms of fairness and accuracy.

To base the introduction of our experiments and results, we first outline the theoretical foundation of the corresponding areas, primarily based on graph theory, machine learning, and fairness. After that, we consider related work with a strong focus on the examined algorithms. Finally, we present our experimental setup and results and conclude with a discussion of our findings.

2 Preliminaries and Assumptions

In this chapter, we concisely outline the preliminaries and assumptions underlying this paper. We do not aim to provide a detailed elaboration of all corresponding foundations, as this would exceed the scope of this paper. These foundations include multiple domains: graph theory, machine learning, and fairness. This chapter provides a concise overview of each area and a founding intuition to place the following definitions and methods.

Therefore, we first briefly overview the graph-related foundations and assumptions. Further, we introduce graph sampling and graph representation. Then we provide a brief overview of this paper's machine learning foundations and methods. Moreover, finally, we introduce fairness definitions and methods.

2.1 Graph Representation

Graphs, as a non-linear data structure, represent a universal language for the modeling and description of complex relational systems from a broad spectrum of domains. These domains include networks modeling physical structures, information networks, and networks representing social structures [ZYZZ18]. Within the scope of this paper, we use them to represent social networks.

Graphs

Let $G = (V, E)$ denote a connected graph containing a set of nodes, with $v_i, i \in \mathbb{N}$, $n := |V|$, E and a set of edges $e_{ij}, (i, j) \in V \times V$, $m := |E|$. Moreover, for this paper, we assume graphs to be bi-directional, as this fits our use case for social networks. With bi-directional, we refer to a graph where the edges are bi-directional, i.e., $e_{ij} = e_{ji}$, see figure 2.1b, where $G = G'$, as in figure 2.1, if and only if the graph is unweighted or all weights are equal.

(a) Undirected graph G. (b) Bi-directed graph G'.

Figure 2.1: Undirected- and (bi-)directed graphs.

Hence, we assume graphs as weighted; the edge weight, see equation (2.1), can differ for each direction. With edge weight, we refer to a function $w : E \to \mathbb{R}$, which assigns a real value to each edge, s.t.

$$e_{ij} = \begin{cases} w(ij) & if\ (v_i, v_j) \in E \\ 0 & \text{otherwise,} \end{cases} \tag{2.1}$$

with $w(ij) \geq 0$. If $(v_i, v_j) \in E$ and $w(ij)$ is undefined, we assume $w(ij) = 1$. Further, let

$$\mathcal{N}(v_i) = \{v_j | (v_i, v_j) \in E\}, \tag{2.2}$$

denote node v_i's immediate neighborhood, the set of nodes that are directly connected to v_i by an edge e_{ij}. Moreover, we denote the number of direct neighbors $|\mathcal{N}(v)|$. Additionally, concerning the assumed bi-directional graphs, the degree of node v, $\deg(v)$, defined as the number of edges incident to v, is equal to the number of direct neighbors $|\mathcal{N}(v)|$. Therefore, $|\mathcal{N}(v)| = \deg(v)$. Consequently $|\mathcal{N}(v)|$ is the 1^{st}-order neighborhood of v and e_{ij} the 1^{st}-order proximity of v_i and v_j. With the 1^{st}-order proximity, we emphasize pairwise node similarities [Xu21, ZYZZ18].

Homophily and Structural Equivalence

A node's immediate neighborhood is strongly connected to its features. Regarding social networks, this is often referred to as homophily, describing the tendency of direct connections to similar nodes and as structural equivalence, i.e., the tendency to have the same neighborhood structure as similar nodes [TMKM18].

(a) Label colors reflecting homophily. (b) Label colors reflecting structural equivalence.

Figure 2.2: Different structural properties of nodes in the same graph [TMKM18].

For instance, regarding social networks, the homophily hypothesis addresses this pairwise node similarity locally, i.e., see figure 3.4a. In general, theories about homophily suggest that nodes with similar labels are more likely to be connected than nodes with different labels.

The structural equivalence hypothesis addresses global similarities based on neighborhood structures [MCC+03, Xu21, ZYZZ18]. For instance, regarding social networks, the structural equivalence hypothesis, see figure 3.4b, describes that nodes with similar labels tend to have similar neighborhood structures [HGER+12].

Graph Sampling

We need information about pairwise node similarities to represent social network graphs concerning homophily and structural equivalence features. The classic approach, using neighborhood matrices, becomes unfeasible complex for large sparse graphs, i.e., real-world social networks [LL10]. One solution to this problem is to sample graph neighborhoods.

In this work, we study such sampling methods. These graph sampling methods aim to approximate the structural properties via pairwise neighborhood similarities using neighborhood samples. The authors of [HL13, WCA+16] group the typical neighborhood sampling approaches for graphs into three categories: (i) vertex sampling, (ii) edge sampling, and (iii) traversal-based sampling.

This work uses traversal-based sampling methods, precisely (biased) random walks. Note that the classic breadth-first search (BFS) and depth-first search (DFS), which are traversal-based sampling methods, but not random walks, can approximately be seen as extreme cases of random walks due to their properties [GL16]. Therefore, before we define (biased) random walks and how we approximate BFS and DFS, we concisely outline them.

Breadth-First-Search and Depth-First-Search

BFS and DFS differ in the way they reveal the graph structure. One exposes homophily, the other structural equivalence. Referencing figure 2.3 and starting with node $u := v$, in both

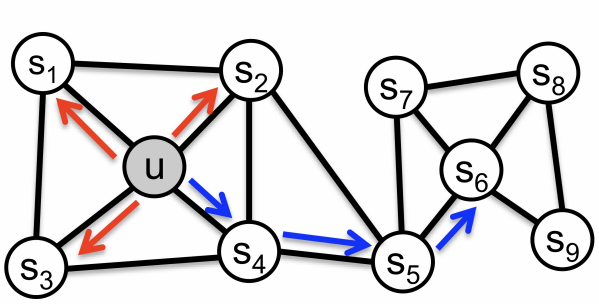


Figure 2.3: Breadth-First-Sampling and Depth-First-Sampling [GL16].

strategies, each neighbor $\mathcal{N}(u)$ is considered by the search. Accordingly, which node is taken into the sample by the algorithm and is visited differs for BFS and DFS. The BFS algorithm

searches in-breadth first, i.e., if the walk length $k = 3$, see figure 2.3 (red), the search is limited to the first-order neighborhood $\mathcal{N}(u) = \{s_1, s_2, s_3\}$. Only if all nodes in $\mathcal{N}(u)$ were visited and $|\mathcal{N}(u)| > k$, BFS considers the farther neighborhood, and so on. Accordingly, the sample contains the k-many nearest neighbors of the root node v. In contrast, the DFS algorithm searches in-depth first. Therefore, DFS visits only one node in $\mathcal{N}(u)$, and after that, visits again only one node in this node's neighborhood, i.e., $\mathcal{N}(u_1) \setminus \{u\}$, and so on. The DFS search distances quickly from the root node u.

Regarding the representation of graph features and structural properties, it was shown that BFS emphasizes structural equivalence properties, e.g., nodes with a high degree $\deg(v)$ tend to be similar, see s_3 and s_8 in figure 2.3. DSF, however, is more likely to reveal homophily properties since nodes in the same neighborhood tend to be similar [GL16]. With this, if a graph representation is, i.e., a priori, supposed to represent either homophily features or structural equivalence features, BFS and DFS can be used as references for random walk biasing.[1]

Random Walks

Let $v \in V$ be an arbitrary node in the graph G and

$$\mathcal{W}^k(v) = (u_0, u_1, ..., u_{k-1}), \tag{2.3}$$

be a k-step random walk starting at node v, with $u_0 := v$ and $k \ll n$, meant to sample a small portion of the graph. [GL16]. Nodes in a walk-sequence do not need to be unique, e.g., $\mathcal{W}^k(v)$ can contain multiple occurrences of the same node.

Given a node u_{t-1} in a walk-sequence $\mathcal{W}^k(v)$, see equation (2.4), we select the node u_t from the neighborhood of the previous node $\mathcal{N}(u_{t-1})$ with probability

$$P(u_t = y | u_{t-1} = x) = \begin{cases} \frac{\pi_{xy}}{\mathcal{Z}} & \text{if } (x, y) \in E \\ 0 & \text{otherwise,} \end{cases} \tag{2.4}$$

where π is the unnormalized transition probability, with $\pi_{xy} = w(xy)$, and $\frac{\pi_{xy}}{\mathcal{Z}}$ the normalized transition probability, with $\mathcal{Z} = \sum_{z \in \mathcal{N}(x)} w(xz)$ [KKB+22, GL16]. Random walks are more likely to traverse edges with higher weights.

[1] The classic BFS and DFS algorithms are deterministic and sample each node only once per traversal. This is true for the resulting sample, even if the search is repeated multiple times and is testing nodes multiple times in one search. This is different for random walks.

If modified systematically, biased random walks can reveal or mask certain information in graph representations. We discuss random walk biasing methods in chapter 3.

Graph Embeddings

The objective of graph embeddings is the reduction of the complexity of a graphs' representation and, at the same time, maintaining the graphs' structure. One reason for graph representation in a latent vector space is to enable the use of machine learning algorithms. Machine learning algorithms, however, are not able to process graphs directly due to their high complexity.

With graph embedding, we describe the projection of all nodes in a given graph into a latent vector space, the embedding space. Assume graph $G = (V, E)$ and let $\phi : V \to \mathbb{R}^{n \times \dim}$ denote the projection to the embedding, which maps each node $v_i \in V$ to a vector $\phi(v_i) = \boldsymbol{z}_i \in \mathbb{R}^{\dim}, i = 1, \ldots, n$, with dim as the embedding dimensionality. Figure 2.4 illustrates an exemplary embedding process based on random walks. The encoding model *Skip-gram* is described regarding related work in chapter 3.

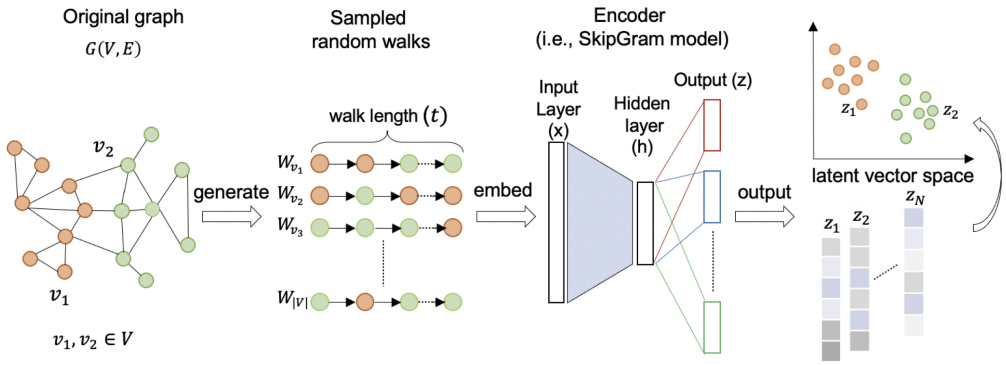


Figure 2.4: Pipeline for random walk based graph embeddings [Xu21].

In general, graph embeddings should capture and represent the structural properties of a graph, as effective and efficient as possible [Xu21]. These graph properties are strongly related to pairwise similarities between nodes and can be either or both, of (i) local origin, in case of homophily, and of (ii) global origin, in case of structural equivalence.

If encoded in a latent vector space, node similarities can approximate the actual graph's structure based solely on these embeddings by using similarity measures such as, i.e., the dot-product or the euclidean distance [Xu21, HYL17]. Accordingly, graph representations in such embedding spaces enable downstream machine learning tasks on graph data and social networks [HYL17]. The three main approaches for the encoding of graphs are:

(i) Matrix-factorization-based methods, (ii) random-walk-based methods, and (iii) neural-network-based methods [Xu21]. In the context of this paper, we limit ourselves to the second approach, namely random-walk-based methods, as in figure 2.4, which we further examine in chapter 3, related work.

2.2 Node Classification

Machine learning models have become part of many facets of our everyday lives. We typically use machine learning for problems being too complex to solve deterministically. One typical application is classification. In this work, we use node classification to evaluate the quality of graph representations resulting from various graph sampling strategies. The following briefly outlines the classification concept, its application to graphs and common classification metrics. For a more in-depth understanding, we refer to the textbooks understanding machine learning [SSBD14] and an introduction to statistical learning [JWHT13], on which we based our introduction.

Machine Learning

In general, learning describes a process considered successful when a learner gains expertise from some experience. Concerning machine learning, the model is successful if it can generalize from training samples to unknown problems [Wig19]. The samples and unknown problems need to fulfill particular requirements. Those requirements concern that data points are independent and identically distributed (i.i.d.) with a measurable distance between them, e.g., located in euclidean space. Therefore, we assume the data points as a vector in $\mathbb{R}^n, n \in \mathbb{N}$ [SSBD14]. This assumption is fulfilled by the graph representation in a latent vector space, as described.

Any machine learning model can be described by a function $h : Z \to Y$, which denotes a hypothesis, where Y is the set of labels and classes, with $\boldsymbol{y}_{lc} \in Y \subset \mathbb{R}^{\mathcal{Y}}$, where $\mathcal{Y}$ describes the labels and classes. Z is the set of features, with $\boldsymbol{z}_i \in Z \subset \mathbb{R}^n,\ n \in \mathbb{N}$. We assume Z as the set of all possible features, i.e., the above-mentioned node embedding, and Y is the set of all possible labels.

Let h_S be a hypothesis trained on a training set $S = \{(\boldsymbol{z}_1, \boldsymbol{y}_1), ..., (\boldsymbol{z}_m, \boldsymbol{y}_m)\} \subset Z \times Y$, where $\boldsymbol{z}_i \in Z$ and $y_i \in Y$. The training set S should be as informative, independent, discriminating, and representative as possible regarding Z and the underlying distribution of Y. We denote the underlying distribution, the real concept c, by $c : Z \to Y$. Machine

learning aims to find a hypothesis h_S that generalizes well to new, unseen data points, i.e., $h_S \approx c$ [Wig19]. With this, we define the generalization error as

$$\mathcal{E}_c(h) = \underset{\boldsymbol{z}\sim c}{P}[h(\boldsymbol{z}) \neq c(\boldsymbol{z})]. \tag{2.5}$$

The generalization error is the probability that the hypothesis h makes a wrong prediction. However, since $c : Z \to Y$ is unknown to the learner, we cannot compute the generalization error directly.

Supervised- and Unsupervised Classification

We can assign a training process of machine learning classification to numerous various categories, of which we only want to elaborate on two, namely supervised learning and unsupervised learning. In supervised learning, the labels are known for all training samples, s.t. $\boldsymbol{y}_i = c(\boldsymbol{z}_i) : \forall \boldsymbol{z}_i \in S$. With this, we provide the learner with correct examples of the target output for training purposes.

Accordingly, the learner is trained by minimizing the empirical loss, denoted

$$\mathcal{E}_s(h_S) := \sum_{i=1}^{m} l_s(h_S(\boldsymbol{z}_i), \boldsymbol{y}_i), \tag{2.6}$$

where l_s is a loss function for supervised learning, i.e., the entropy loss, which is defined by $l_s(h(\boldsymbol{z}_i), \boldsymbol{y}_i) := -\log(h(\boldsymbol{z}_i)\boldsymbol{y}_i)$ [ITAC19]. To optimize the model $h_S : Z \to Y$, the learner aims to minimize the empirical error in equation (2.6) and thus, to maximize the prediction

$$\hat{\boldsymbol{y}} = \arg\max_{h_S} h_S(\boldsymbol{z}). \tag{2.7}$$

Unsupervised learning, in contrast, provides the learner with an input set of raw, unlabeled data $S \subset Z$. The learner then aims to find patterns to group the data points into clusters. With this, the learner aims to minimize the consistency loss, which encourages coherence among different transformations of samples and is appliable to labeled and unlabeled data. Therefore, let

$$\mathcal{E}_u(h_S) := \sum_{i=1}^{m} l_u(h_S(\boldsymbol{z}_i), h_{\hat{S}}(\hat{\boldsymbol{z}}_i)), \tag{2.8}$$

denote the unsupervised loss, where $h_{\hat{S}}$ can be equal to h_S as well, as a different, set of functions. Additionally, $\hat{\boldsymbol{z}}_i$ is a transformed version of $\boldsymbol{z}_i$. Further, l_u describes the loss function for the unsupervised learning, i.e. the euclidean distance, $l_u(h_S(\boldsymbol{z}_i), h_{\hat{S}}(\hat{\boldsymbol{z}}_i)) := \|\boldsymbol{z}_i - \hat{\boldsymbol{z}}_i\|_2$ [ITAC19].

Validation

With learning by a sample set S, there are two common problems. Let over-fitting and under-fitting describe biases caused by characteristics of the training set [JWHT13, SSBD14]. Over-fitting occurs when the learner is too specific about the training data. As a result, the error is minimal on the training data, but the model cannot generalize well to unknown problems. Under-fitting occurs when the training data is not specific enough for the learner, resulting in a high variance in predictions on the unknown concept $c : Z \to Y$.

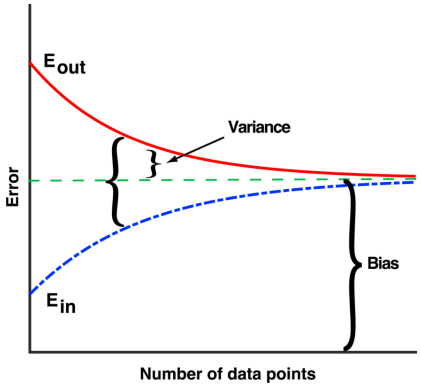


Figure 2.5: Bias in machine learning: over-fitting and under-fitting. Training (E_{in}) and generalization (E_{in}) error as function of training size. [MBW+19].

To further evaluate the desired generalization of the so inferred function $h_S : S \to Y$, we define a test set $T \subset Z \setminus S$. T has to be unknown by the learner. The test set T is used to evaluate the generalization error of the hypothesis h_S. Considering the training biases of a machine learning classification model, splitting the data into a training set and a test set only once can lead to unbalanced data, negatively impacting the representation. To avoid possible incorrect representations by the splitting, we can use cross-validation.

Cross-validation describes the systematic resampling of the training sample into several chunks of training and test data, i.e., splitting S into a training set $S_i = S \setminus T_i$ and a test set $T_i = S \setminus S_i$. Cross-validation is used for the best possible evaluation of the generalization error of a model, especially if the data, or the classes, are imbalanced. Ideally, splitting the training and test set is done so that every $(\boldsymbol{z}_i) \in Z$ is at least once a test sample and a training sample. With this, we can apply appropriate metrics to evaluate the model's generalization [SSBD14].

Classification Metrics

In order to evaluate the performance of a node classification model, let $h_S : Z \to Y$ be a classification model trained on a training set $S \subset Z$ and tested on $T \subset Z \setminus S$.

We further assume binary labels $(\boldsymbol{z}, \boldsymbol{y}) \in \{0, 1\}$, and denote the predicted label by $\hat{\boldsymbol{y}}$. Consequently, a prediction, is correct if $((\boldsymbol{x}, \hat{\boldsymbol{y}}), (\boldsymbol{x}, \boldsymbol{y})) = (0, 0)$ or $(1, 1)$ and is not correct if $((\boldsymbol{z}, \hat{\boldsymbol{y}}), (\boldsymbol{z}, \boldsymbol{y})) = (0, 1)$ or $(0, 1)$. With this we distinguish the following categories [HM15, VR18]:

(i) true positive (TP): $((\boldsymbol{z}, \hat{\boldsymbol{y}}), (\boldsymbol{z}, \boldsymbol{y})) = (1, 1)$,
the predicted label is positive, and the actual label is positive,

(ii) true negative (TN): $((\boldsymbol{z}, \hat{\boldsymbol{y}}), (\boldsymbol{z}, \boldsymbol{y})) = (0, 0)$,
the predicted label is negative, and the actual label is negative,

(iii) false positive (FP): $((\boldsymbol{z}, \hat{\boldsymbol{y}}), (\boldsymbol{z}, \boldsymbol{y})) = (1, 0)$,
the predicted label is positive, and the actual label is negative, and

(iv) false negative (FN): $((\boldsymbol{z}, \hat{\boldsymbol{y}}), (\boldsymbol{z}, \boldsymbol{y})) = (0, 1)$,
meaning the predicted label is negative, and the actual label is positive.

These four categories can be extended to multi-class labeling problems [KLPS12, KBŠBŠ20, PVG+11]. Therefore, in table 2.1 we introduce the multi-class confusion matrix for $(\boldsymbol{z}, \boldsymbol{y}) \in 0, 1, 2, 3$ regarding label-prediction. In terms of a multi-class evaluation, we capture the re-

		Predicted Class												
		Label 0				Label 1				Label 2				...
		0	1	2	4	0	1	2	4	0	1	2	4	...
Actual Class	0	TP	FN	FN	FN	TN	FP	TN	TN	TN	TN	FP	TN	...
	1	FP	TN	TN	TN	FN	TP	FN	FN	TN	TN	FP	TN	...
	2	FP	TN	TN	TN	TN	FP	TN	TN	FN	FN	TP	FN	...
	3	FP	TN	TN	TN	TN	FP	TN	TN	TN	TN	FP	TN	...

Table 2.1: Multi-class confusion matrix

lation of the predicted label to the actual label for every class separately, Meaning, as in table 2.1, if we evaluate label 2, we consider label 2 as True and labels 0, 1, and 3 as False. We count occurrences for each class individually for the categories TP, TF, FP, and FN.

Based on the named categories, we can obtain common classification measures: Equation (2.9) and equation (2.11) are measures for the positive class, i.e., the class with the true label. Equation (2.10) is a measure of the model's overall performance. Equation (2.12) is

a measure for the overall performance of the model and is a combination of equation (2.9) and equation (2.11).

$$\text{Precision} : \frac{TP}{TP+FP}, \tag{2.9}$$

$$\text{Recall} : \frac{TP}{TP+FN}, \tag{2.11}$$

$$\text{Accuracy} : \frac{TP+TN}{TP+TN+FP+FN}, \tag{2.10}$$

$$\text{F1} - \text{Score} : \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}. \tag{2.12}$$

Note that in the case of a multi-class evaluation, accuracy is not a suitable metric for imbalanced data because it is not sensitive to class distribution. Especially if the data is imbalanced, i.e., the positive class is underrepresented, a model can achieve high accuracy by predicting the negative class for all instances in this class. In this case, the model does not detect the positive class but achieves high accuracy. Therefore, in multi-class labeling and for unbalanced datasets, the F1-Score is a more appropriate metric to evaluate the performance of a classification model as it combines precision and recall, the measures for positive labels, and better fits imbalanced class distribution [HM15, VR18].

We further distinguish three cases of averaged measures for each for labels-based classification metrics [PVG+11, KBŠBŠ20]. Therefore, let h_S be the classifier over set S and $\mathcal{B}(h_S)$ any measure, defined in equation (2.9) till equation (2.12), for the classifier h_S. Further we assume C the classes for S, w_c the weight of class $c \in C$, with $\sum_{c \in C} w_c = 1$, and $\mathcal{B}_c$ the measure for class c, with $\mathcal{B}_c(h_S) = \sum_{s \in S} \mathbb{I}(s = c)$. With this in mind, we denote the average values for the above-defined metrics as follows:

$$\mathcal{B}_{\text{weighted}}(h_S) = \sum_{c \in C} w_c \cdot \mathcal{B}_c(h_S), \tag{2.13}$$

as the weighted sum for each class separately. With equation (2.13), we can account for unbalanced datasets with the weighted measure by considering the relative proportion.

$$\mathcal{B}_{\text{micro}}(h_S) = \mathcal{B}\Big(\sum_{i \in S} \text{TP}_i, \sum_{i \in S} \text{TN}_i . \sum_{i \in S} \text{FP}_i, \sum_{i \in S} \text{FN}_i\Big). \tag{2.14}$$

The micro measure equation (2.14) is the global measure for all classes combined.

$$\mathcal{B}_{\text{macro}}(h_S) = \frac{1}{|S|} \sum_{i \in S} \mathcal{B}(\text{TP}_i, \text{TN}_i, \text{FP}_i, \text{FN}_i). \tag{2.15}$$

With the macro measure equation (2.15), we describe the unweighted mean of the measures, which can be problematic with unbalanced datasets. We further describe our application of metrics to measure the accuracy, regarding the experimental setup of this work, in section 4.4

2.3 Fairness

In machine learning, the term fairness is often used synonymously with justice. These two terms are different, as justice is a normative concept, while fairness is a descriptive concept [KKBK21]. Consequently, fairness involves morality, and justice involves justification, e.g., ethics and law. In this work, we use the term fairness to describe the ethical concept of justice. Further, as the fairness-enhancing sampling method we study is based on group fairness, we focus on group-related fairness.

Group Fairness

Generally, group-related fairness is concerned with the distribution of outcomes among groups of individuals, also called distributive justice, describing the allocation of goods and services in a society [KKBK21]. The main concepts of distributive justice are:

(i) Egalitarianism: everyone is equal and should be treated equally.

(ii) Equality of opportunity: everyone should have an equal chance to succeed.

(iii) Desert: everyone should be rewarded according to their merits.

(iv) Sufficientarianism: everyone should be rewarded according to their needs.

(v) Prioritanism: everyone should be rewarded according to their priority.

In machine learning, we can consider the allocation of resources as a consequence of predictions or decisions made by a machine learning model as a form of distributive justice [KKBK21]. Fairness intervention approaches in machine learning can be parted into three categories, see figure 2.6: (i) pre-processing, concerning the data collection and preparation, i.e., the data itself, (ii) in-processing, concerning the model training, i.e., the model and hyperparameter settings, and (iii) post-processing, concerning the model outcome, i.e., emphasizing interventions [CLL22, CH20].

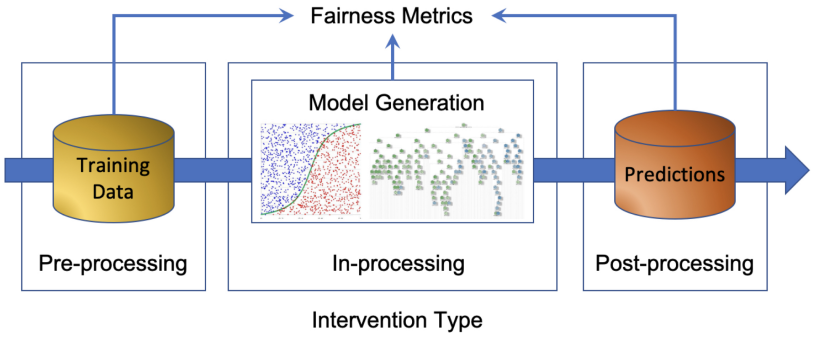


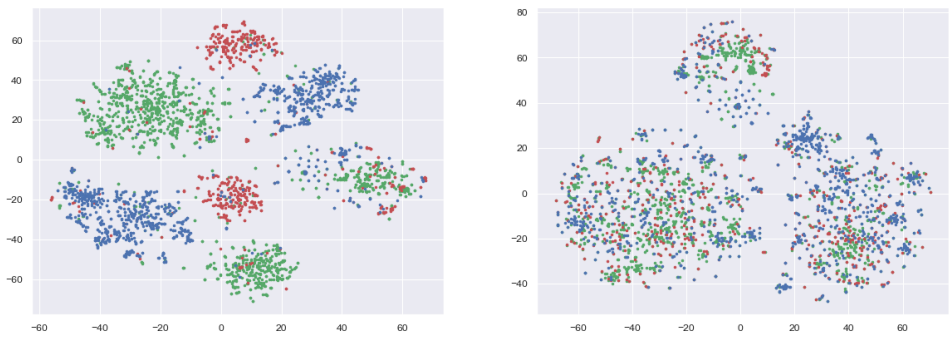
Figure 2.6: Categories of fairness interventions in machine learning [CH20].

According to the survey [CH20], most fairness approaches in machine learning are based on the notion of sensitive attributes, which also can define unprivileged groups. This sensitive attribute can be, i.e., age, gender, religion, or sexual orientation, and should best be masked in the graph representation to avoid discrimination and bias. Hiding these attributes is challenging with graphs, as graph structures contain complex information about proxies and relations for sensitive attributes, which can reveal sensitive attributes [CH20].

In this paper, we focus on group fairness based on sensitive attributes and the in-processing category, as we aim to evaluate different sampling methods. For further discussion of other fairness concepts in machine learning. [CH20, CLL22, LRKS18].

Fairness Metrics

According to [CH20], there are various fairness metrics for in-processing-based approaches, which need to be more consistent in their definition or interpretation of what fair means. As we aim to evaluate the fairness of graph representations with different sampling methods, we focus on the invariance of graph embeddings to sensitive attributes [BH19]. In other



(a) Clustered by sensitive attribute classes. (b) Invariant to sensitive attribute classes.

Figure 2.7: Graph embeddings colored accordingly to sensitive attribute classes.

words, we consider a graph representation fair if it does not reveal sensitive attributes. With this, concerning social networks, we include three aspects of fairness: (i) privacy, concerning the protection of sensitive attributes, (ii) bias, the unfair treatment of groups based on sensitive attributes, and (iii) discrimination, the unfair treatment of individuals based on their belonging to a sensitive attribute class.

Therefore, we focus on particular fairness based on the representational invariance of graph embeddings regarding sensitive attributes. This invariance can be seen as a pragmatic user-centric approach, i.e., if the user is concerned with the privacy of sensitive attributes and

therefore does not want to have such attributes revealed, the graph representation should be invariant to this very attribute [BH19].

Modeling of Fairness

Let $G = (V, E, Y, A)$ be a graph, where V is the set of nodes, E is the set of edges, $\boldsymbol{y}_i \in Y \subset \mathbb{N}$ as the target attributes, $\boldsymbol{a}_v \in A \subset \mathbb{N}$ as the sensitive attributes of node $v \in V$, and $\hat{\boldsymbol{y}}_v$ and $\hat{\boldsymbol{a}}_v$ the corresponding predicitons. Further, let $\phi(v) = \boldsymbol{z}_v \in \mathbb{R}^{\text{dim}}$ denote the embedding of any node $v \in V$.

Let the pairwise independence of the node embeddings and the sensitive attributes be

$$\boldsymbol{z}_v \perp\!\!\!\perp \boldsymbol{a}_v, \quad \forall \boldsymbol{v} \in V, \tag{2.16}$$

denoting the representational invariance of a node embedding $\boldsymbol{z}_v$ to a sensitive attribute $\boldsymbol{a}_v$. With equation (2.16), we assume the mutual information $I(\boldsymbol{a}_v, \boldsymbol{z}_v) \approx 0$ and expect the sensitive attribute as not revealed by the node representation. This invariance as pragmatic fairness approach [BH19] satisfies various group-related fairness criteria, i.e., the classification of the target attribute can not be influenced by the sensitive attribute if the underlying graph representation is invariant to this embedding.

From $I(\boldsymbol{a}_v, \boldsymbol{z}_v) \approx 0$ and equation (2.16) follows that for every $v \in V$ any classification, or prediction, $\hat{\boldsymbol{y}}_v$ on a given node representation $\boldsymbol{z}_v$, with target label $\boldsymbol{y}_v$ and sensitive attribute $\boldsymbol{a}_v$, satisfies the notions of

(i) statistical parity $P(\hat{\boldsymbol{y}}_v|\boldsymbol{a}_v) = P(\hat{\boldsymbol{y}}_v)$,

(ii) equality of odds $P(\hat{\boldsymbol{y}}_v|\boldsymbol{a}_v, \boldsymbol{y}_v) = P(\hat{\boldsymbol{y}}_v|\boldsymbol{y}_v)$,

(iii) equality of accuracy $P(\hat{\boldsymbol{y}}_v|\boldsymbol{a}_v, \boldsymbol{y}_v) = P(\hat{\boldsymbol{y}}_v|\boldsymbol{y}_v)$,

(iv) equality of opportunity $P(\hat{\boldsymbol{y}}_v|\boldsymbol{a}_v, \boldsymbol{z}_v) = P(\hat{\boldsymbol{y}}_v|\boldsymbol{z}_v)$,

among others concerning the sensitive attribute.

Consequently, discrimination, i.e., the unfair treatment of individuals, and bias, i.e., the unfair treatment of groups, based on a specific attribute, hardly emerge if the graph representation is invariant to this very attribute. In the section 4.4, we outline how we evaluated the invariance of the graph embeddings to sensitive attributes.

3 Related Work

Among the various graph representation methods outlined in section 2.1, we focus only on the most related work concerning graph embeddings created from random walks. In the following, we introduce *DeepWalk*, as it is the pioneering method to generate graph embeddings from random walks with a natural language processing approach. After that, we outline *node2vec*, a more recent and generalized method that improves upon *DeepWalk*. Finally, concerning the fairness of graph representations, we introduce *CrossWalk*, a method that enhances the fairness of any random walk-based graph embedding algorithm.

3.1 DeepWalk

The authors of *Deepwalk: Online Learning of Social Representations* [PARS14] aimed to learn graph representations of social networks by sampling uniform random walks. The resulting *DeepWalk* algorithm is a natural language processing-based approach for learning graph embeddings from local neighborhoods generated by truncated random walks. This approach is extendable to not only social networks but various graphs.

The underlying natural language processing (NLP) method is *Skip-gram*. This neural network-based method to learn word embeddings was introduced by Mikolov et al. [MCCD13]. The general idea behind *Skip-gram* is that if words appear in a similar context, they tend to have similar meanings. Therefore, the aim is basically to predict the context of a word, thus, the most probable neighboring words in a sentence. This idea was shown to translate well to the graph domain if we think of words as nodes, sentences as walks, and the context of a node as a set of neighbors.

Modeling of *DeepWalk*

Let $G = (V, E)$ be a bi-directional weight graph with V as the nodes and E as the edges. Assume $\mathcal{W}^k(v) = (u_0, u_1, ..., u_{k-1})$, with $w_i \in V$, $w_0 := v$, and a walk length k for an arbitrary node $v \in V$. If this walk is thought of as a sentence in a unique language, the analog to the NLP models, see figure 3.1 [1], is to estimate the likelihood of the next node w_k given the previous $k-1$ nodes, leading to $P[w_k|w_0, ..., w_{k-1}]$ [PARS14].

[1] In this context, a window describe a short sequence of words with a specified length.

However, since the goal of *DeepWalk* is to learn graph embeddings, we are interested in the probability of a node, given its context, which is the set of the nodes' neighbor's node embeddings. Therefore, let $\phi : v \in V \rightarrow \mathbb{R}^{n \times \dim}$, $n = |V|$ denote a node embedding,

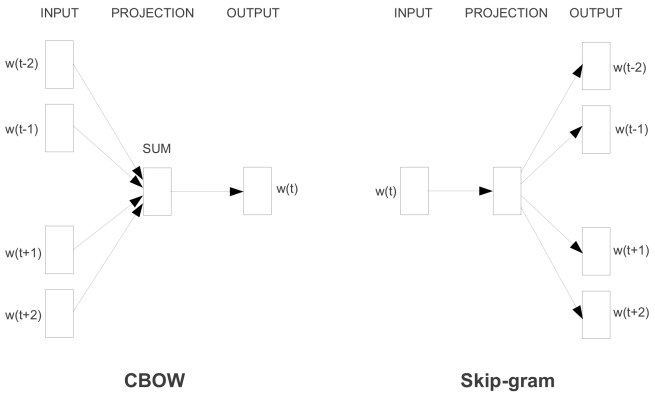


Figure 3.1: *Continuous Bag-of-Word Model* (CBOW) and *Skip-gram* [MCCD13].

see section 2.1. In accordance to the *Continuous Bag-of-Word Model* (CBOW) [MCCD13], the problem can be formulated as

$$P[w_t \mid (\phi(w_0), \phi(w_1), \ldots, \phi(w_{t-1}))], \tag{3.1}$$

where w_t is the next node in the sequence, and $\phi(w_{1\ldots w-1})$ being the node embeddings. Hence, since the projection ϕ is represented by a matrix of $n \times \dim$ free parameters, the computation of equation (3.1) becomes unfeasible for large graphs [PARS14].

DeepWalk's solution to this problem mentioned above is to reverse equation (3.1). In line with *Skip-gram*, see figure 3.1, we can write this negative sampling approach as

$$\min_{\phi} -\log \mathcal{P}[\{w_{i-k/2}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{i+k/2}\} \mid \phi(w_t)]. \tag{3.2}$$

Consequently, the prediction is conditioned by the node w_t rather than the entire neighborhood. The authors of *Deepwalk* then use the very *Skip-gram* algorithm [MCCD13], to solve equation (3.2), which can be formulated as

$$\max_{\phi} \sum_{v \in V} \log \mathcal{P}(\mathcal{W}^k(v) \mid \phi(v)), \tag{3.3}$$

describing the log probability of observing a neighborhood of a node v given its embedding [GL16]. *Hierarchical Softmax* is further used to speed this process up. For this, a sequence of

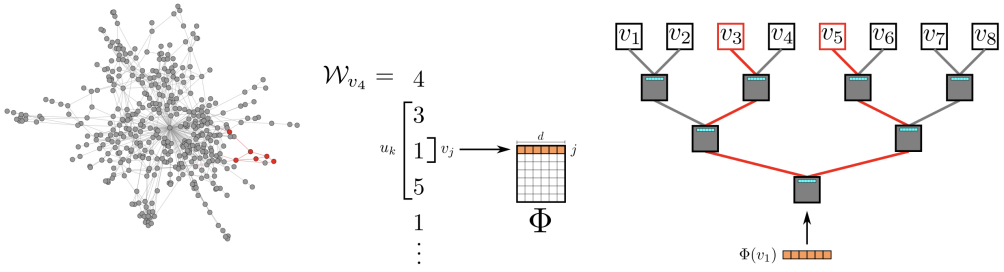
nodes (window) in a walk are grouped into a binary tree, see figure 3.2. With this, the problem is reduced to a binary classification problem. Assume a specific path $(b_0, b_1, ..., b_{\lceil \log n \rceil})$ in the tree; then the optimization problem can be written as the probability

$$P[w_t \mid \phi(w_u)] = \prod_{l=1}^{\lceil \log n \rceil} P[b_l \mid \phi(w_u)], \tag{3.4}$$

with $w_t = b_{\lceil \log n \rceil}$ and w_u being the parent node of w_t in the binary tree [PARS14]. Stochastic gradient descent is then used to optimize the parameters of the neural network. We refer to [GL16, MCCD13, PARS14] for a more detailed description of the algorithm.

The *DeepWalk* Algorithm

Altogether, the *Deepwalk* algorithm samples a stream of short, uniform random walks rooted at randomly chosen nodes on the given graph G. The algorithm then updates the encoding, respectively the node embedding, by optimizing the *Skip-gram* likelihood objective equation (3.3), more precisely equation (3.4). Figure 3.2 illustrates the steps *DeepWalk* takes to learn graph embeddings.



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

Figure 3.2: Overview of the *DeepWalk* algorithm [PARS14].

In conclusion, *DeepWalk* is a scalable unsupervised method to learn latent graph embeddings. It was shown by the authors of [PARS14] that the algorithm performs well, even on sparsely labeled data. This performance is somehow surprising, considering that the *DeepWalk* algorithm only considers the 1st-order proximity of nodes in a graph and, therefore, captures more local structure information, and the global structure information is probably not well-represented [Xu21].

3.2 Node2vec

The authors of *node2vec: Scalable Feature Learning for Networks* [GL16] proposed a generalization of *DeepWalk*. The *node2vec* algorithm is, like *DeepWalk*, an unsupervised method to learn latent graph embeddings from social networks. Moreover, just as well as *DeepWalk*, it applies to various graphs.

The main difference between *node2vec* and *DeepWalk* is the opportunity to modify random walks by introducing two parameters p and q. This generalization allows biased random walks, in contrast to the uniform random walks of *DeepWalk*. The authors stated that *node2vec* is designed as '[...] a flexible neighborhood sampling strategy which allows us to smoothly interpolate between BFS and DFS.' [GL16]. Besides the ability to simulate several sampling strategies, *node2vec* is based on the same objective, precisely equation (3.3), the negative sampling, and the *Skip-gram*. Hence, in contrast to *DeepWalk*, the objective is not optimized by hierarchical softmax but by stochastic gradient descent. For a further description of this optimization, we refer to [GL16], as it is not the focus of this paper.

Modeling of *node2vec*

Assume a similar foundation as in DeepWalk, see section 3.1, with a bi-directional weighted graph $G = (V, E)$, walks $\mathcal{W}^k(u)$, and we assume $\mathcal{N}(v)$ the neighborhood of a node v, see equation (2.2). Let π_{vx} denote the unnormalized transition probability for random walks from node v to node x, with $(v, x) \in \mathcal{W}^k(u)$, and $\pi_{vx} = w(vx)$, according to equation (2.4).

To modify random walks in such a way as to influence representations of graph structures in encoded latent representations, the authors of [GL16] introduced what they named a 2^{nd}-order random walk. This 2^{nd}-order random walk aims to modify walks based on a search bias, which accounts for a broader neighborhood.

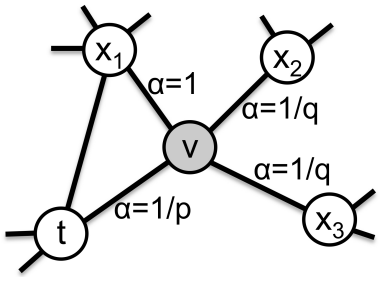


Figure 3.3: The search bias β in *node2vec* [GL16].

Therefore, let $\beta_{pq}(t, x)$ denote the search bias, which is based on two hyperparameters p and q, and $\mathcal{N}(v) \cup \mathcal{N}(t)$, the union, and therefore the 2^{nd} order neighborhood of the actual

node v and the previous node in the walk, which we denote t. Further, let $x \in \mathcal{N}(v)$ be the next possible node in the walk.

With this, the new transition probabilities for the random walk are set by $\pi_{vx} := w(vx) \cdot \beta_{pq}(t, x)$, where

$$\beta_{pq}(t,x) := \begin{cases} \frac{1}{p} & \text{if } \mathrm{dist}(tx) = 0, \\ 1 & \text{if } \mathrm{dist}(tx) = 1, \\ \frac{1}{q} & \text{if } \mathrm{dist}(tx) = 2. \end{cases} \tag{3.5}$$

In equation (3.5) $\mathrm{dist}(tx)$ denotes the shortest distance between t and x in the graph, see figure 3.3.

Consequently, with the two hyperparameters p and q, we can control the search bias α. Whereas p controls the probability of returning to the previous node in the walk and q controls the probability of exploring a new node, which leads to the following intuitions, based on figure 3.3 and according to [GL16]:

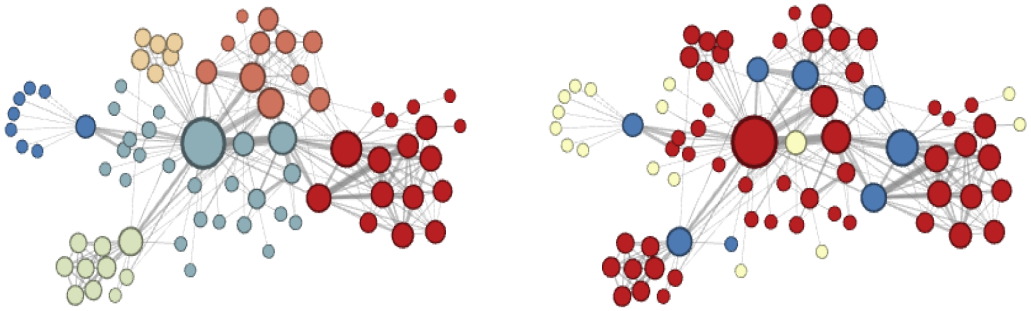
- $\boldsymbol{q}$: With $\boldsymbol{q > 1}$, the random walk tends to explore the 1st order neighborhood of t, resulting in a more inward-, or local walk, s.t. the random walk approximates the breadth-first search (BFS) strategy.

  With $\boldsymbol{q < 1}$, the random walk tends to visit nodes farther away from the previous node t, leading to a more outward walk, s.t. the random walk approximates the depth-first search (DFS) strategy in a broader sense.

- $\boldsymbol{p}$: For any $\boldsymbol{p > \max(q, 1)}$, the walk is less likely to turn back to the previous node t. Consequently, a larger q encourages exploration and avoids the walk redundancy.

  For any $\boldsymbol{p < \min(q, 1)}$, the random walk is more likely to turn back to the previous node t. Consequently, a smaller q encourages backtracking, which leads to a more local walk in the root node neighborhood.

For example, regarding figure 3.4a, the parameter where set to $p = 1$ and $q = 0.5$, resulting in a biased random walk tending towards a DFS approximation, and therefore reveals the homophily structure in the graph. In figure 3.4a, the parameter where set to $p = 1$ and $q = 2$, resulting in a biased random walk tending towards a BFS approximation, which reveals the structural equivalence in the graph. Moreover, the random walk with $p = 1$ and $q = 1$ is the equivalent to the uniform random walk of *DeepWalk* [GL16].



(a) Label colors reflecting homophily. (b) Label colors reflecting structural equivalence.

Figure 3.4: Embeddings of the *Les Misérables* co-appearance network concerning different structural properties [GL16].

The *node2vec* Algorithm

Altogether, *node2vec* is a two-step algorithm, which is based on the same objective as *DeepWalk* [PARS14], see equation (3.3). The first step is to generate multiple truncated random walks, which are biased based on 2nd-order neighborhoods and the hyperparameter p and q. The second step is to train a *Skip-gram* model to generate a latent graph embedding that reveals the graph structure controlled by hyperparameter settings.

The authors of [GL16] have shown, that *node2vec* can smoothly interpolate between the approximations of the BFS and DFS strategies and the uniform random walk. Consequently, *node2vec* can reveal homophily and structural equivalence in a latent graph representation. The authors further demonstrated that *node2vec* outperforms *DeepWalk* and various other, not random, walk-based, methods for multi-label classification and link prediction.

3.3 CrossWalk

To our knowledge, there are only two methods directly related to the fairness of random walk-based graph embeddings. The first method was *Fairwalk* [RSBZ19] and, its generalization, the second method, was *CrossWalk* [KKB+22]. Our focus is on *CrossWalk* as it was the only published algorithm for enhancing fairness in random walk-based graph embeddings concerning node classification.

With *CrossWalk: Fairness-enhanced Node Representation Learning* the authors [KKB+22] introduced a method designed to modify random walks by grouping nodes to modify random walk processes systematically. This modification is based on the assumption that a random walk process, initiated in a given sensitive attribute group, biased towards nodes on the group boundary or in other groups, leads to fairness-enhanced graph representations.

Therefore, the core idea of this fairness-enhancing approach is to systematically re-weight edges based on their proximity to group boundaries before applying any random walk-based graph embedding methods, such as *DeepWalk* or *node2vec*.

Modeling of *CrossWalk*

Let $G = (V, E, A)$ be a bi-directional weighted graph with a set of nodes V, a set of edges E, and the sensitive attribute classes $a \in A$. With this, we assume nodes are assigned to groups, representing the sensitive classes $a \in A$. We write these groups of nodes as $\{V_1, \ldots, V_c\}$ for $c := |A|$. Additionally, a_v denotes the group to which the node v belongs. Further we assume random walks $\mathcal{W}^k(v)$, rooted in node $v \in V$, with length k and unnormalized transition probabilities $\pi_{xy} = w(x, y)$, with $(x, y) \in V$, according to equation (2.4) [KKB+22].

The authors of [KKB+22] proposed the modification of the transition probabilities $\hat{\pi}_{xy}$ based on the proximity of node x to the group boundary. Therefore, let

$$\text{cfn}(x) = \frac{\sum_{i \in (1,\ldots,r)} \sum_{y \in \mathcal{W}^l(x)} \mathbb{I}[a_x \neq a_y]}{r \cdot l}, \tag{3.6}$$

denote the proximity of a node x to other groups based on r-many shortened random walks of length l, where $\mathbb{I}[a_x \neq a_y]$ is the indicator for visited nodes assigned to other groups. Consequently, $\text{cfn}(x)$ describes the expected number of times the walk $\mathcal{W}^l(x)$ visits a node from a different group. With this, nodes closer to group boundaries tend to have a more colorful neighborhood and, therefore, a higher $\text{cfn}(x)$ value [KKB+22].

For example, in figure 3.5 node $u \in S$, which is close to the group boundary of the red class S, is better connected to the blue and green groups than $w \in S$, which would lead to a higher $\text{cfn}(u)$, than $\text{cfn}(w)$ value.

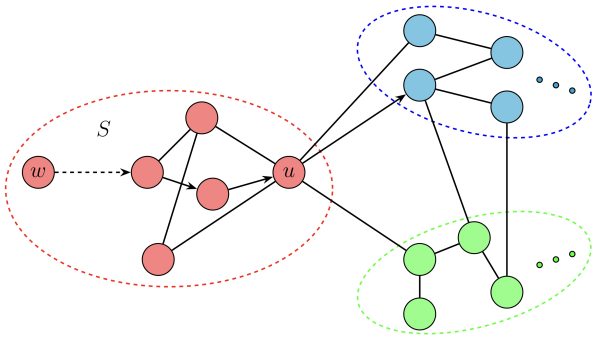


Figure 3.5: Neighborhood proximity in the *CrossWalk* algorithm [KKB+22].

In addition to the proximity of a node to other groups, the authors of [KKB+22] introduced sets of nodes defined by the neighborhood groups. Therefore, let

$$\mathcal{C}_x = \{\cup_{y \in \mathcal{N}(x)} c_y \mid c_x \neq c_y\}, \tag{3.7}$$

denote the set of different groups in the 1st-order neighborhood of node x and $|\mathcal{C}_x|$ their count, let

$$\mathcal{N}_x^+ = \{y \in \mathcal{N}(x) \mid c_x = c_y\}, \tag{3.8}$$

denote the set of all nodes from the same group in the 1st-order neighborhood of node x, and let

$$\mathcal{N}_x^- = \{y \in \mathcal{N}(x) \mid c_x \neq c_y\}, \tag{3.9}$$

be the set of all nodes from different groups in the 1st-order neighborhood of node x.

For example, in figure 3.5, we have the following sets:

$$\begin{aligned}
\mathcal{C}_u &= \{\cup_{y \in \mathcal{N}(x)} c_y \mid c_x \neq c_y\} = \{\text{blue}, \text{green}\}, \text{ with } |\mathcal{C}_u| = 2, \\
\mathcal{N}_u^+ &= \{y \in \mathcal{N}(u) \mid c_u = c_y\} = \{\text{three red nodes}\}^2, \text{ and} \\
\mathcal{N}_u^- &= \{y \in \mathcal{N}(u) \mid c_u \neq c_y\} = \{\text{one green, and two blue nodes}\}.
\end{aligned}$$

The authors of [KKB+22] introduced $\alpha \in (0, 1)$ and $\exp \geq 1$, to control random walks based on $\text{cfn}(x)$, equation (3.6) and the neighborhood groups, $\mathcal{C}_x$, $\mathcal{N}_x^+$, and $\mathcal{N}_x^-$. Let

$$\hat{\pi}_{xy} = \begin{cases} \pi_{xy} \cdot (1 - \alpha) \cdot \frac{\text{cfn}(y)^{\exp}}{\sum_{z \in \mathcal{N}_x^+} \pi_{xz} \cdot \text{cfn}(z)^{\exp}} & \text{if } y \in \mathcal{N}_x^+ \\ \pi_{xy} \cdot \alpha \cdot \frac{\text{cfn}(y)^{\exp}}{|\mathcal{C}_x| \cdot \sum_{z \in \mathcal{N}_x^-} \pi_{xz} \cdot \text{cfn}(z)^{\exp}} & \text{if } y \in \mathcal{N}_x^-, \end{cases} \tag{3.10}$$

denote the new modified and normalized transition probability for an edge $(x, y) \in V$, with $1 - \alpha$ as the sum of edges only connecting a node x to the same group, see, i.e., (v_1, v_7) and (v_1, v_8) in figure 3.6, and $\frac{\alpha}{|\mathcal{C}_x|}$ as the sum of edges connecting a node x to different groups, see, i.e., $(v_1, v_i),\ i = 2, \ldots, 6$ in figure 3.6, where $w_{xy}, w'_{xy} \equiv \pi_{xy}, \hat{\pi}_{xy}$.

The authors of [KKB+22] showed that larger (i) α up weight edges connecting different groups and down weights edges connecting nodes from the same group, and (ii) exp controls the influence of the group proximity value $\text{cfn}(x)$ on the transition probability. Consequently, larger values for α and exp lead to a higher probability of visiting nodes from different groups, as shown in figure 3.6.

[2] Assuming bi-directional edges.

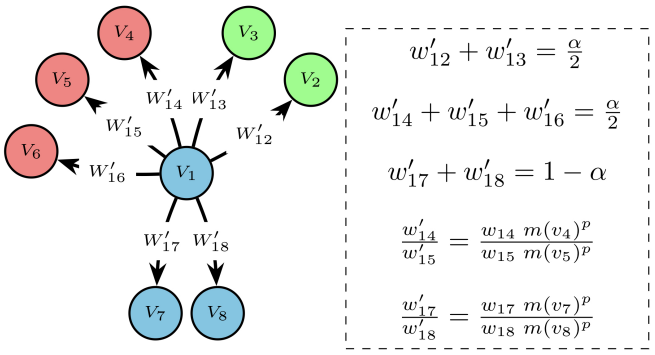


Figure 3.6: *CrossWalk*'s α' influence on edge weights connecting different groups [KKB+22].

The *CrossWalk* Algorithm

The *CrossWalk* algorithm takes a weighted graph $G = (V, E, A)$, with specified node groups based on sensitive attributes, as input. To calculate cfn(x) *CrossWalk* initiates random r-many short random walks, see equation (3.6), for every node $v \in V$. Then *CrossWalk* calculates the new transition probabilities $\hat{\pi}_{xy}$ for every edge $(x, y) \in E$ based on equation (3.10). Finally, *CrossWalk* returns a modified graph G_{fair} with the modified edgeweights $w_{fair}(x, y) = \hat{\pi}_{xy},\ \forall (x, y) \in E$. On this resulting graph G_{fair}, we can apply random walk algorithms, such as *DeepWalk* and *node2vec*, to obtain fairness-enhanced node embeddings.

The authors of [KKB+22] demonstrated that *CrossWalk* enhances the fairness, as statistical parity, of random walk-based node embeddings for node classification and link prediction tasks.

4 Experimental Setup

In the following, we present the experimental setup. We first define the data sets used in this work. Then we describe the sampling procedure and the evaluation metrics. Finally, we describe the evaluation process.

4.1 Data

Before we define the data generation process, we introduce the *Pokec* social network used for our experimentations. *Pokec* [1] is the most popular online social network in Slovakia, and was even before the rise of *facebook* [Fac04], regarding *Stanford Network Analysis Project* (SNAP) [LK14]. The anonymized dataset we use is a snapshot of the network, crawled in May 2012 by the authors of [TZ12]. See table 4.1 and figure 4.1 for some of the *Pokec* network snapshots characteristics[2].

	Nodes	Edges	Friends	Pair-dist.*	Men	Woman	Public-contacts	Public-age
Abs.	1,632,803	30,622,564	–	4.67	802,556	831,725	1,088,838	1,125,734
Rel.	–	–	–	–	0.4911	0.5089	0.6662	0.6888
Avg.	–	–	22.181	4.67	–	–	–	–
Max.	–	–	13,840	6	–	–	–	–

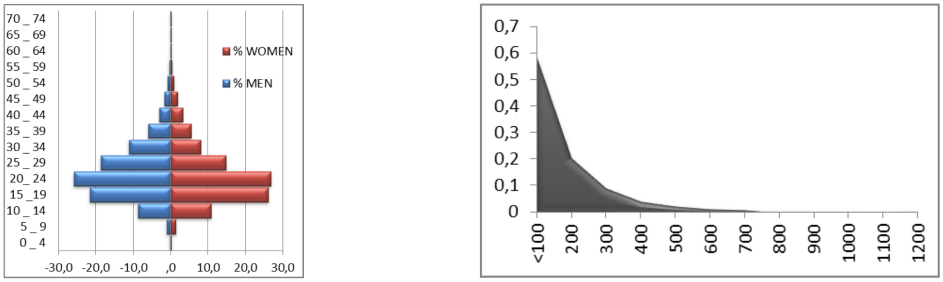
*Avg. of 100 randomly chosen pairs

Table 4.1: Characteristics of the *Pokec* network snapshot [TZ12].

The anonymized data is available from *SNAP*[3], (i) as a directed edge list, with anonymized id's instead of user names, containing all connected node pairs, and (ii) a file containing the user profile data with multiple attributes for each user id. We filtered this social network graph to generate sub-graphs that meet various structural properties which fit our purpose. The three main goals for the filtering of the data are:

(i) The reduction of the network's complexity, respectively size, to run the experiments on the available hardware within the scope of this work.

[1] https://pokec.azet.sk
[2] The *Pokec* dataset distinguishes friendship relations, as *is-friend* and *has-friend*. For simplification, we assume a bi-directional connection in either case
[3] http://snap.stanford.edu



(a) Age distribution

(b) Node degree distribution

Figure 4.1: Distributions of age and node degree in the *Pokec* snapshot [TZ12].

(ii) To generate graphs with a specific, semi-synthetic[4] structure;

(iii) The generation of classes for the target attributes and sensitive attributes, tending somewhat to be slightly unbalanced in order to simulate real-world data and to be able to assess sampling methods concerning unbalanced data.

Note that we expect the predictions to be accurate for the target attributes. For the sensitive attributes, we expect the predictions inaccurate to fit the invariance aspect of fairness, see section 2.3. Concerning this, we filter the *Pokec* graph by two attributes, one is the region, describing the place of living and the other one is the age; both attributes we expect to correlate with friendship connections, respectively pairwise node similarities. To further generate three unbalanced classes for the groups based on the sensitive labels, we used the following thresholds for the attribute age:

- group 0 with age ≤ 19,
- group 1 with age $\in [20, 22]$, and
- group 2 with age ≥ 23.

The so-filtered network contains only nodes with public age information; we further filtered for regions. For this, we selected regions in such a way as to generate three subgraphs with expected properties described in the following. For a visualization of the resulting graphs, see figure 4.2, generated with *igraph* and DrL[5] [CN06]. The three subgraphs are:

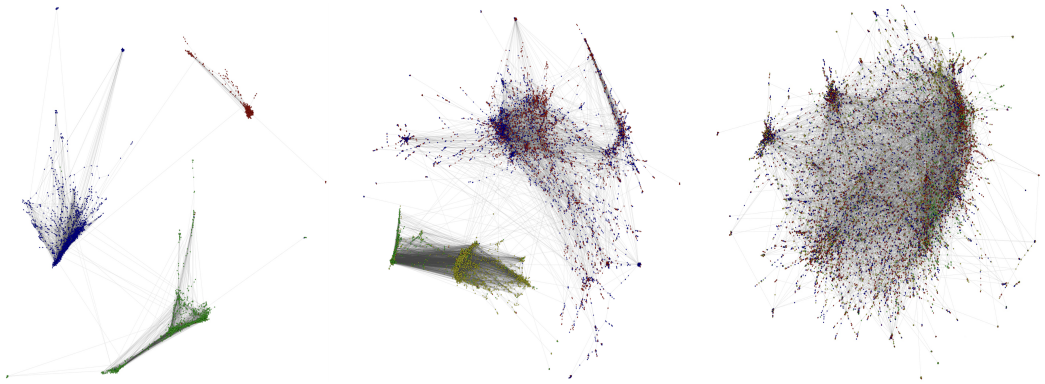
(a) the *distinct*; three distinct small towns with few connections, see figure 4.2a,

(b) the *semi-distinct*; two regions with two towns in each region and therefore more connections between the town-pairs and less between the regions, see figure 4.2b, and

(c) the *mixed*, five adjacent city districts with many connections, see figure 4.2 (c).

[4] With semi-synthetic, we describe real-world data selected to approximate desired structural characteristics.
[5] DrL is a force-directed graph layout toolbox focused on real-world, large-scale graphs.

For each sub-graph, we labeled the selected region classes with numeric values, represented as colors in the visualizations, in figure 4.2c.



(a) The *distinct* sub-graph. (b) The *semi-distinct* sub-graph. (c) The *mixed* sub-graph.

Figure 4.2: The generated *Pokec* sub-graphs.

Table 4.2, table 4.3, and table 4.4 summarize the characteristics of each of the sub-graphs, assuming $G = (V, E)$, with $|\mathcal{C}_v \in V|$, see equation (3.7), as the number of groups in the 1^{st}-order neighborhood $\mathcal{N}(v)$.

Distinct sub-graph (a)	Sensitive attribute groups			Target attribute groups		
	0	1	2	0	1	2
Nodes	3,156					
Edges	13,667					
Density	0.002745					
Path len. (avg.)	5.9579					
$\lvert\mathcal{C}_v \in V\rvert = 0$	3,087			1,736		
$\lvert\mathcal{C}_v \in V\rvert = 1$	571			19		
$\lvert\mathcal{C}_v \in V\rvert = 2$	849			50		
Group (abs.)	1,089	602	1,456	416	1,493	1,247
Group (rel.)	0.3479	0.1907	0.4613	0.1318	0.4730	0.3951

Table 4.2: Characteristics of the *distinct Pokec* sub-graph (a).

Due to the different structures of the data sets, we can consider not only different parameter settings but also the influence of different network structures on random walk samplings within our experiments.

Semi-distinct sub-graph (b)

	Sensitive attribute groups			Target attribute groups			
	0	1	2	0	1	2	3
Nodes	$10,812$						
Edges	$39,840$						
Density	0.000681						
Path len. (avg.)	6.5558						
$\lvert\mathcal{C}_v \in V\rvert = 0$	$6,425$			$7,050$			
$\lvert\mathcal{C}_v \in V\rvert = 1$	$1,613$			$2,093$			
$\lvert\mathcal{C}_v \in V\rvert = 2$	$2,774$			631			
$\lvert\mathcal{C}_v \in V\rvert = 3$	–			$1,038$			
Group (abs.)	$2,726$	$1,875$	$6,211$	$3,568$	$3,477$	980	$2,787$
Group (rel.)	0.2521	0.1734	0.5744	0.3300	0.3216	0.0906	0.2578

Table 4.3: Characteristics of the *semi-distinct Pokec* sub-graph (b).

Mixed sub-graph (c)

	Sensitive attribute groups			Target attribute groups				
	0	1	2	0	1	2	3	4
Nodes	$16,695$							
Edges	$28,556$							
Density	0.000205							
Path len. (avg.)	6.6091							
$\lvert\mathcal{C}_v \in V\rvert = 0$	$11,601$			$8,645$				
$\lvert\mathcal{C}_v \in V\rvert = 1$	$1,957$			$3,147$				
$\lvert\mathcal{C}_v \in V\rvert = 2$	$3,137$			$1,453$				
$\lvert\mathcal{C}_v \in V\rvert = 3$	–			$3,079$				
$\lvert\mathcal{C}_v \in V\rvert = 4$	–			317				
Group (abs.)	$2,232$	$2,427$	$12,036$	$6,171$	$4,201$	$1,880$	$4,008$	435
Group (rel.)	0.1337	0.1454	0.7209	0.3696	0.2516	0.1126	0.2401	0.0260

Table 4.4: Characteristics of the *mixed Pokec* sub-graph (c).

4.2 Implementation

In order to meet the objectives of this work, namely the analysis of different sampling methods, we selected two graph sampling algorithms that offer the most generalized approaches to the simulation of random walks.

(i) The *node2vec* algorithm is the most generalized random walk-based algorithm available, see section 3.2,. With *node2vec*'s ability to generate various graph embeddings by biasing random walks and the resulting samplings, we can compare the results of different sampling methods, as, i.e., *DeepWalk*, BFS, and DFS, approximated by specific hyperparameter settings. (ii) The *CrossWalk* algorithm is the most generalized available algorithm for the modification of random walks targeting fairness enhancing of resulting graph embeddings.

While *node2vec* is available as a stable *GitHub* repository [6] [Coh18] based on [GL16], *Crosswalk* was available embedded in a *Deepwalk* implementation, regarding the related publication [KKB+22] [7] [Kha21]. To better match our purpose, we extended the *node2vec* implementation to include the *CrossWalk* algorithm. The modified node2vec method allows us to perform the intended experiments with or without the CrossWalk modification through additional parameter settings.

4.3 Hyperparameter

Table 4.5 is a summary of the hyperparameter used in our experiments, with the actual possible values for each parameter and the neutral value if existent. Each parameter fits the respective introduction of the algorithms in chapter 3.

Parameter	Parameter Name	Possible values	Neutral value	Intuition
p	Return-	$p > 0, p \in \mathbb{R}$	1 if $q = 1$	small p: local walks, larger p: exploring walks.
q	In-out-	$q > 0, q \in \mathbb{R}$	1 if $p = 1$	$q < 1$: approx. DFS, $q > 1$: approx. BFS.
α	Alpha	$\alpha \in (0, 1)$	–	α biases walks to cross-group borders.
exp	Exp	$\exp \geq 1, \exp \in \mathbb{R}$	1	exp biases the walk towards group borders.
k	Walk length	$w > 0, w \in \mathbb{N}$	–	Walk length for $\mathcal{W}$, see equation (2.3).
l	Pre-walk length	$l > 0, w \in \mathbb{N}$	–	Walk length for cfn, see equation (3.6).
dim	Dimensions	$\dim > 0, dim \in \mathbb{N}$	–	Dimension for the node embeddings.

Table 4.5: Hyperparameter characteristics.

[6] https://github.com/eliorc/node2vec
[7] https://github.com/ahmadkhajehnejad/CrossWalk

Table 4.6 presents the values used for the experiments, separately for each, the pure sampling setting with *node2vec*, and the fairness-enhancing sampling setting, *node2vec* extended by *CrossWalk*. For the targeted values, we performed a grid search over the sets in the column *experimental values*. We set other parameters, not in the scope of this paper, to the default values regarding the source implementations. These default parameters, including such for *Skip-gram*, set by the actual *node2vec* implementation mentioned above. Further, we did not change the parameter for the number of walks per node, which we set to the default value 10, and for dim, the dimension of the node embeddings, which we set to the default value 128.

	Parameter	Experimental values	Idea
node2vec	p	$0.1, 0.25, 1, 2.5, 10$	Cases for returning, exploring, and unbiased walks
	q	$0.1, 0.25, 1, 2.5, 10$	Cases for returning, exploring, and unbiased walks
	k	$40, 80, 120, 160$	Examine the effect of the walk length
CrossWalk	p	$0.1, 1, 10$	Cases for returning, exploring, and unbiased walks
	q	$0.1, 1, 10$	Cases for DFS and BFS approximation, and unbiased walks
	k	80	Default 80
	l	$4, 6, 8$	Examine the pre-walk length regarding cfn
	α	$0.01, 0.05, 0.1, 0.25, 0.4, 0.5, 0.6, 0.75, 0.9, 0.99$	Examine the group-border crossing
	exp	$1, 1.5, 2.5, 4, 6, 10, 15, 25$	Examine the biasing toward other groups.
	Training-split	$0.1, 0.25, 0.5, 0.75$	Examine the effect of the labeled samples proportion.

Table 4.6: Experimental hyperparameter settings.

4.4 Evaluation

In this section, we describe how we evaluate the performance of the examined sampling methods for (i) fairness, the embedding of the sensitive attribute, and for (ii) accuracy, the embedding of the target attribute. In both cases, we use the same evaluation setup and the same evaluation metrics, as described in the following.

Node Classification

For the evaluation of the resulting graph embeddings, we used a semi-supervised classification task, namely *LabelPropagation* for node classification [ZG02]. For semi-supervised *LabelPropagation*, assume that the labels of nodes are only known for some of the nodes. With

this, the machine learning model is supposed to iteratively group the data into categories based on similarities in their embedded features and a few labels, see figure 4.3.

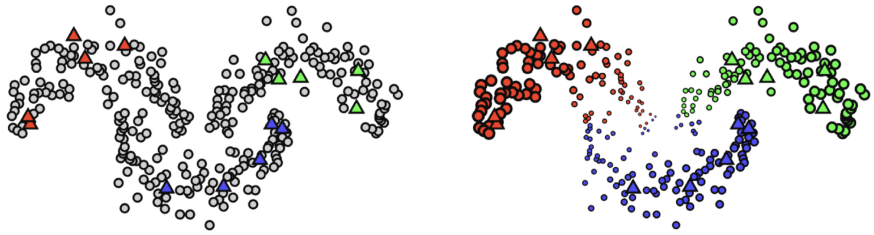


Figure 4.3: *LabelPropagation* with few labeled nodes [ITAC19].

Further, we used cross-validation to evaluate the performance of the embedding methods on the node classification task in a general way. With this, we created various sets with different train and test set sizes. As the train set, we provided node embeddings with known labels, and as the test, we provided node embeddings with unknown labels and used the labels only for the evaluation.

For *LabelPropagation* [8] and *Stratified Cross Validation*[9] we used the implementation provided by [PVG+11]. With this classification setting, we trained various classification models with different train-test-split ratios and evaluated them as described in the following section 4.4.

Accuracy Evaluation

We decided to use the *F1-score* as the main evaluation metric for the quality of the embedding methods regarding the various parameter settings, as the *F1-score* is the harmonic mean of the *precision* and *recall*, see equation (2.12). We evaluated the weighted, the macro, and the micro version of the *F1-score* and the individual scores for every group.

As further reference, we also measured the accuracy, see equation (2.10) and the variance in accuracy regarding the groups of the respective other attributes, meaning the sensitive groups for the target attribute and vice versa.

Fairness Evaluation

To assess the fairness and the invariance of a graph embedding, we use the F1-score, which we utilize to measure the invariance as a proxy for the fairness of the embedding. Therefore, we classify and evaluate the sensitive attribute using two reference values. One is absolute,

[8] https://scikit-learn.org/stable/modules/generated/sklearn.semi_supervised.LabelPropagation.html
[9] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedShuffleSplit.html

measured by the relative share of the class regarding the whole set of attribute holders, and the other is relative to a specific node sampling setup.

(i) In the first case, the embedding would be invariant to the sensitive attribute class if the F1-score corresponds to the class proportion, thus being equally distributed so that the outcome could just as well have been guessed randomly, i.e., if the class proportion is 0.5, the F1-score should be 0.5.

(ii) In the second case, we measure the effect of hyperparameter settings, which explicitly influence the fairness of embeddings, with a referential baseline, i.e., a related sampling with constant hyperparameters. With this, we can measure the effectiveness of the hyperparameter settings on the invariance of the embedding regarding the sensitive attribute class, i.e., by comparing the F1-score for each parameter setting of the embedding with the F1-score of the baseline.

For the comparison of the results, we should keep in mind that the biasing of random walks by the *CrossWalk* is based on the sensitive attribute, which is not the case for the *node2vec* method. We examine the embedding for the effect of this very biasing on the whole embedding. We, therefore, use the quality of the target attribute as a proxy and *node2vec* as a baseline.

5 Results

In the following, we outline the achieved results. In every case, we compare the two main methods, namely the *node2vec* and *CrossWalk*. Also, we compare the results for the different label classes, the target, and the sensitive attributes. Additionally, we review results for several hyperparameters, as walk length, p, and q in the case of *node2vec* and the pre-walk length, α, and exp for *CrossWalk*. Regarding the closer look at the selected hyperparameter, we also consider the influence of the training split on the classification results.

5.1 Results Overview

The following tables show the calculated mean, median, minimum, and maximum of the mean F1-scores for the results of node classifications. These node classifications, based on a 50 % labeled training set and five-fold cross-validation, were learned on various embeddings. These embeddings were generated on all scheduled hyperparameters settings, as listed in table 4.6. We calculated the F1-scores as weighted, micro, and macro, as well as for each group separately. The F1-scores are further extended by the overall accuracy and the variance, as mentioned in the experimental setup. See, table 5.1 for the *distinct* sub-graph (a), table 5.2 for the *semi-distinct* sub-graph (b), and table 5.3 for the *Mixed* sub-graph (c).

Moreover, we illustrated the abovementioned F1-scores as line plots; see figure 5.3 for sub-graph (a), figure 5.6 for sub-graph (b), and figure 5.9 for sub-graph (c). For these plots, we used the mean of the F1-scores, for the same classification setting as above, as the y-axis. Additionally, we projected the proportion of the attribute class for each group related F1-Score. For *CrossWalk*, the x-axis represents the value of parameter α, and the different colors represent the values of parameter exp. The parameters p and q are set to 1, and the pre-walk-length to 6 in this setting. For *node2vec*, the x-axis represents the value of parameter q, and the different colors represent the values of parameter q. The walk-length: 80 is fixed in this setting. The columns further alternate over the different attributes.

To further visualize the effect of *CrossWalk* on the invariance of embedding regarding the sensitive attribute, we visualized two embeddings for each graph with colored sensitive groups and reduced dimensionality with t-SNE; see figure 5.2 for sub-graph (a), figure 5.5 for sub-graph (b), and figure 5.8 for sub-graph (c).

	Mean		Median		Min		Max	
	CrossWalk	node2vec	CrossWalk	node2vec	CrossWalk	node2vec	CrossWalk	node2vec
	Sensitive attribute		Sensitive attribute		Sensitive attribute		Sensitive attribute	
$F1_{weighted}$	0.6846	0.7302	0.6464	0.7305	0.5046	0.7181	0.8901	0.7406
$F1_{macro}$	0.6343	0.6818	0.5766	0.6824	0.4301	0.6649	0.8795	0.6954
$F1_{micro}$	0.7048	0.7387	0.6738	0.7392	0.5398	0.7269	0.8905	0.7490
Acc.	0.7048	0.7387	0.6738	0.7392	0.5398	0.7269	0.8905	0.7490
Var.	0.0026	0.0032	0.0013	0.0031	0.0000	0.0015	0.0111	0.0059
$F1_{group0}$	0.7329	0.7896	0.7176	0.7899	0.5505	0.7750	0.9059	0.8022
$F1_{group1}$	0.4070	0.4576	0.2605	0.4583	0.0799	0.4143	0.8310	0.4880
$F1_{group2}$	0.7630	0.7980	0.7435	0.7980	0.6241	0.7867	0.9060	0.8094
	Target attribute		Target attribute		Target attribute		Target attribute	
$F1_{weighted}$	0.9792	0.9806	0.9795	0.9807	0.9691	0.9774	0.9858	0.9832
$F1_{macro}$	0.9730	0.9748	0.9734	0.9751	0.9567	0.9692	0.9826	0.9789
$F1_{micro}$	0.9792	0.9806	0.9795	0.9807	0.9687	0.9773	0.9858	0.9831
Acc.	0.9792	0.9806	0.9795	0.9807	0.9687	0.9773	0.9858	0.9831
Var.	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0002	0.0001
$F1_{group0}$	0.9527	0.9560	0.9540	0.9562	0.9157	0.9421	0.9745	0.9658
$F1_{group1}$	0.9851	0.9862	0.9854	0.9862	0.9747	0.9819	0.9914	0.9903
$F1_{group2}$	0.9810	0.9822	0.9813	0.9822	0.9681	0.9770	0.9881	0.9871

Table 5.1: Results of a classification with a 50% training-set and five-fold cross-validation on the *distinct* sub-graph (a).

Additionally, we compared the results of the *CrossWalk* and *node2vec* embeddings, directly for selected hyperparameter, see figure 5.1, figure 5.4, and figure 5.7. Thereby we iterated through parameter $p \equiv q$ with exp$= 4$ and $\alpha \in \{0.1, 0.4, 0.6, 0.9\}$ for *CrossWalk*.

Before we examine the parameter sensitivity and the results for selected hyperparameter combinations, we first discuss the results concerning the abovementioned tables and illustrations in the following.

Results Overview for the *Distinct* Sub-Graph (a)

In table 5.1, we see that the results for the sensitive attribute and the embeddings generated by *CrossWalk* show lower mean values, which indicates a higher invariance of the embedding towards this attribute. In particular, the maximum and minimum values differ from *node2vec*. This is especially noticeable for the minority class, group 2.

For the target attribute, however, these results are less different. Even if *CrossWalk* achieves somewhat lower values, this indicates that while *CrossWalk*'s impact on the sensitive attribute on the extreme values is significant, the embedding quality concerning the target attribute does not change much.

Node2vec, on the other hand, shows only minor deviations between the minimum and maximum values for both attributes. Hence, for the *distinct* sub-graph (a) setting, it can be seen

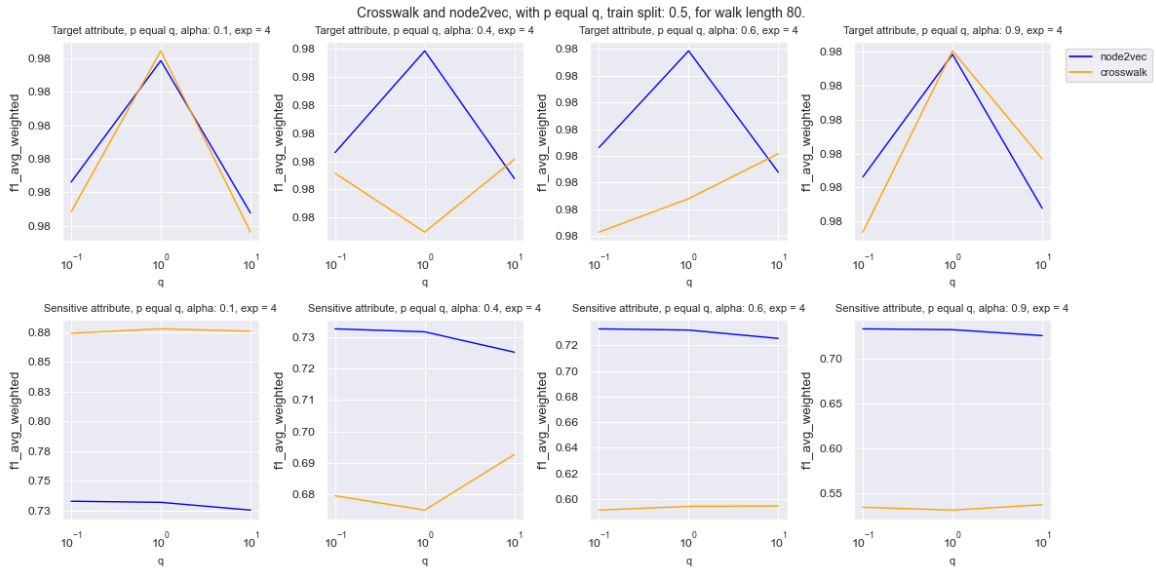


Figure 5.1: Results compared for *node2vec* and *CrossWalk* with varying parameter settings on the *distinct* sub-graph (a), each on the target attribute and the sensitive attribute for multiple parameters.

that *CrossWalk* achieves a higher invariance towards the sensitive attribute. In contrast, the embedding quality for the target attribute is not significantly affected.

Further, for the target attribute, it is recognizable that the mean values are relatively high, and the difference between the minimum and maximum values is slight. Therefore, the structure of the *distinct* sub-graph (a) makes it easy to predict the target attribute.

Taking into account figure 5.3, we see the previous observations confirmed as there beings only a relatively small difference between the mean values of the target attribute for both embeddings compared to the differences in the sensitive attribute. We can see that the parameter α significantly impacts the results, especially for the sensitive attribute of the minority class, group 2. For the sensitive attribute, it is observable that for small α values, the prediction quality is even better with *CrossWalk* than with *node2vec*, but reaches the invariance with growing α values. The most significant difference between both algorithms is seen in group 2, the minority class. The results seem relatively stable over the parameter settings on both attributes, for both algorithms on the target attribute and with *node2vec*.

In figure 5.1, we can see that for selected parameter settings, the results for *CrossWalk* and *node2vec* on the target variable do not differ much. However, for the sensitive attribute, the results for *CrossWalk* become better regarding invariance, meaning lower F1-scores, with increasing α values.

(a) *CrossWalk* sampling parameter: p: 1, q: 1, alpha: 0.05, exp: 2.5

(b) *CrossWalk* sampling parameter: p: 1, q: 1, alpha: 0.6, exp: 6

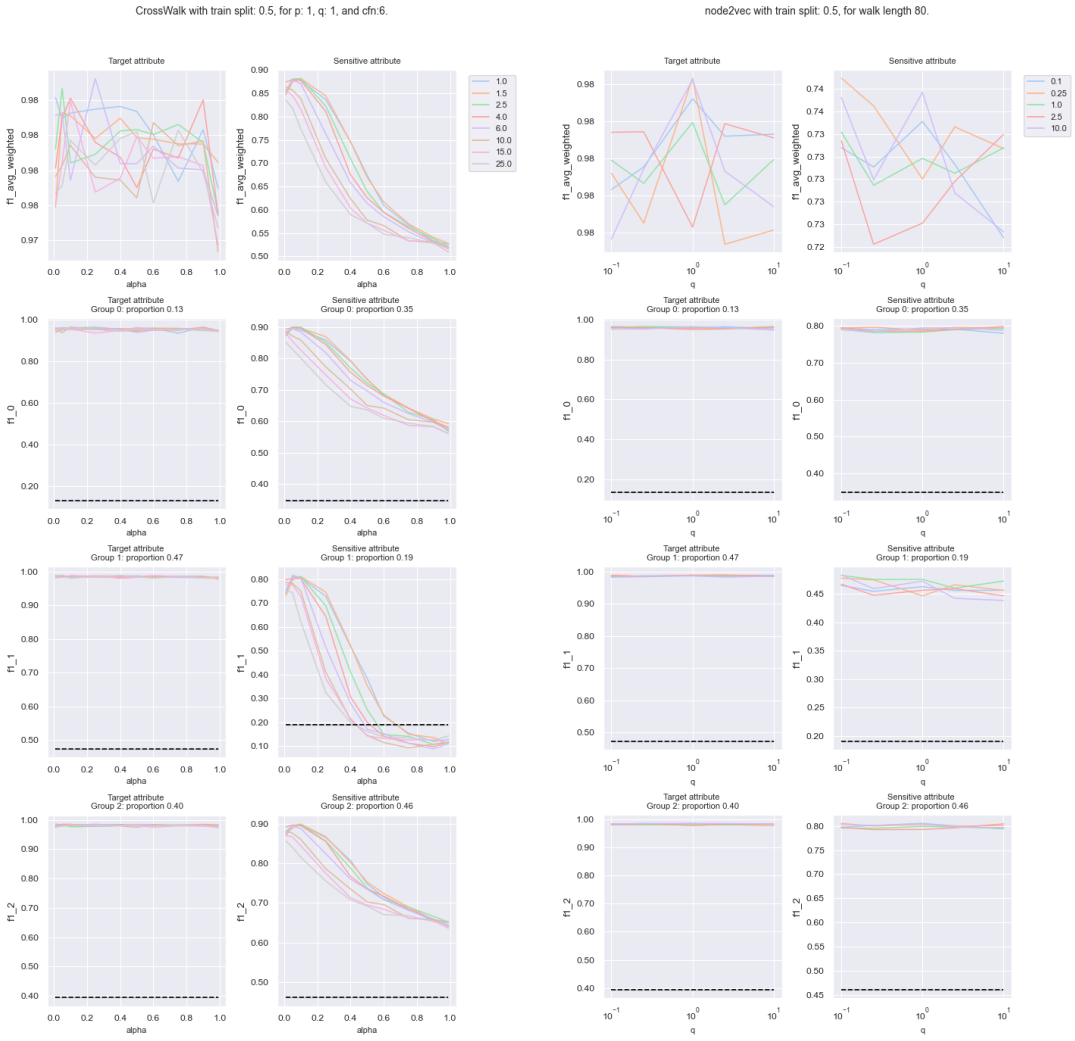
Figure 5.2: Node embedding colored according to the sensitive groups generated with *CrossWalk* for the *distinct* sub-graph (a), with reduced dimensionality using t-SNE.

Results Overview for the *Semi-distinct* Sub-Graph (b)

Table 5.2 shows the results on the *semi-distinct* sub-graph (b). The results show similar tendencies for the sensitive attribute and the embeddings generated by *CrossWalk*. However, a bit weaker as on *distinct* sub-graph (a) in table 5.1. Here also, we see lower mean values and different maximum and minimum values on the sensitive attribute for *CrossWalk* compared to *node2vec*. This effect is comparable to setting (a) most noticeable for group 2, the minority class. Regarding the target attribute, however, these results vary more than those for the embedding of the *distinct* sub-graph (a).

Node2vec, on the other hand, similar to the *distinct* setting, only shows relatively small deviations between the minimum and maximum values for both attributes. Looking at figure 5.1, for the *semi-distinct* sub-graph (b) setting, it can be seen that *CrossWalk* achieves significant invariance towards the sensitive attribute, especially for the minority group. In contrast, the embedding quality for the target attribute is more affected than in the previous setting (a). Like in the *distinct* setting, *Crosswalk* shows similar tendencies on the sensitive attribute. The predictions are again more accurate for lower α values and more invariant towards the sensitive attribute for higher α values. With this, *Crosswalk* reaches the group proportion on the sensitive attribute, the absolute invariance reference, for the minority group analogous to setting (a).

Altogether, the predictions on the *semi-distinct* setting are not as good as in the *distinct* setting. The classification of embeddings from both algorithms achieves significant results, but the prediction seems more affected by parameter settings. In figure 5.4, the direct comparison of the difference between the two algorithms on the target attribute is more significant. In



(a) *CrossWalk* (b) *node2vec*

Figure 5.3: Group related results compared for *node2vec* and *CrossWalk* on the *distinct* sub-graph (a), each on the target attribute and the sensitive attribute for multiple parameters for classification on a 50% training set.

	Mean		Median		Min		Max	
	CrossWalk	node2vec	CrossWalk	node2vec	CrossWalk	node2vec	CrossWalk	node2vec
	Sensitive attribute		Sensitive attribute		Sensitive attribute		Sensitive attribute	
$F1_{weighted}$	0.6979	0.7562	0.6609	0.7568	0.5465	0.7463	0.8787	0.7652
$F1_{macro}$	0.6185	0.6922	0.5656	0.6937	0.4227	0.6782	0.8538	0.7039
$F1_{micro}$	0.7197	0.7611	0.6885	0.7613	0.5906	0.7519	0.8796	0.7696
Acc.	0.7197	0.7611	0.6885	0.7613	0.5906	0.7519	0.8796	0.7696
Var.	0.0019	0.0004	0.0018	0.0003	0.0001	0.0002	0.0061	0.0006
$F1_{group0}$	0.6396	0.7296	0.6116	0.7302	0.4247	0.7115	0.8614	0.7410
$F1_{group1}$	0.4036	0.5025	0.2986	0.5057	0.0946	0.4759	0.7968	0.5261
$F1_{group2}$	0.8123	0.8445	0.7919	0.8447	0.7237	0.8386	0.9160	0.8496
	Target attribute		Target attribute		Target attribute		Target attribute	
$F1_{weighted}$	0.7247	0.7633	0.7242	0.7636	0.6851	0.7556	0.7598	0.7689
$F1_{macro}$	0.7503	0.7891	0.7510	0.7896	0.7045	0.7814	0.7840	0.7960
$F1_{micro}$	0.7247	0.7633	0.7241	0.7637	0.6856	0.7556	0.7596	0.7688
Acc.	0.7247	0.7633	0.7241	0.7637	0.6856	0.7556	0.7596	0.7688
Var.	16.6528	15.9056	16.5221	15.8073	7.4199	12.5562	30.2604	21.2014
$F1_{group0}$	0.6400	0.6886	0.6370	0.6893	0.5952	0.6759	0.6889	0.6991
$F1_{group1}$	0.6299	0.6729	0.6293	0.6731	0.5873	0.6573	0.6727	0.6866
$F1_{group2}$	0.8099	0.8558	0.8157	0.8569	0.7239	0.8400	0.8562	0.8689
$F1_{group3}$	0.9214	0.9392	0.9232	0.9395	0.8876	0.9310	0.9400	0.9437

Table 5.2: Results of a classification on a 50% training-set and five-fold cross-validation on the *semi-distinct* sub-graph (b).

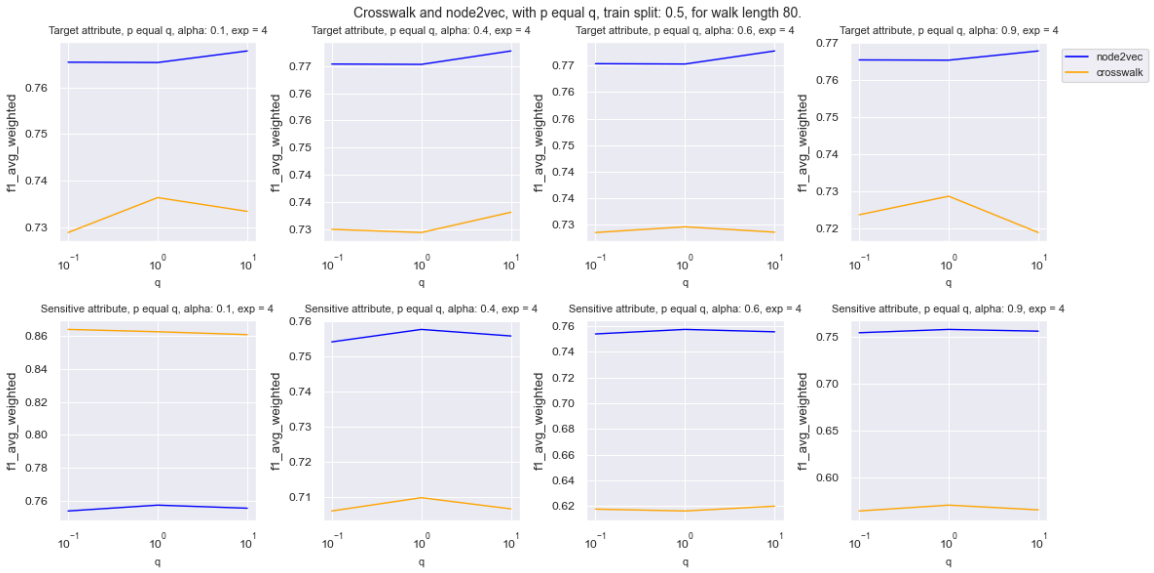
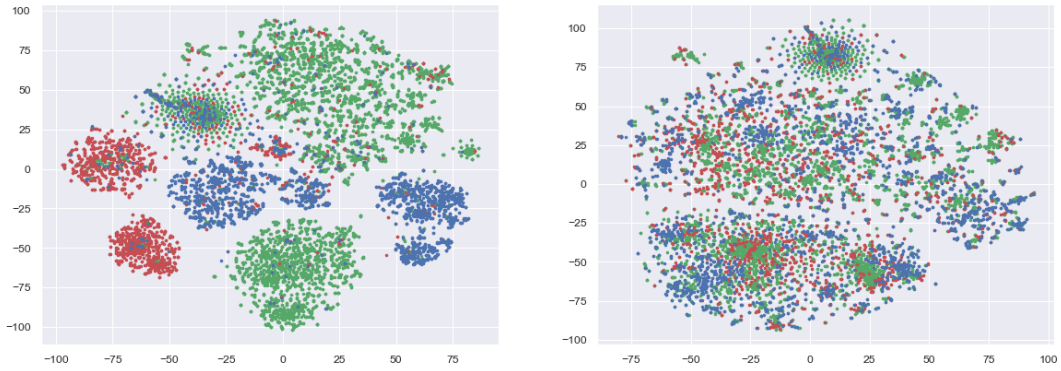


Figure 5.4: Results compared for *node2vec* and *CrossWalk* with varying parameter settings on the *semi-distinct* sub-graph (b), each on the target attribute and the sensitive attribute for multiple parameters.



(a) *CrossWalk* sampling parameter: p: 1, q: 10, alpha: 0.01, exp: 1.5

(b) *CrossWalk* sampling parameter: p: 1, q: 1, alpha: 0.5, exp: 6

Figure 5.5: Node embedding according to the sensitive groups generated with *CrossWalk* for the *semi-distinct* sub-graph (b), with reduced dimensionality using t-SNE.

contrast, the difference on the sensitive attribute is less between both than on the *distinct* set, given the same parameter settings.

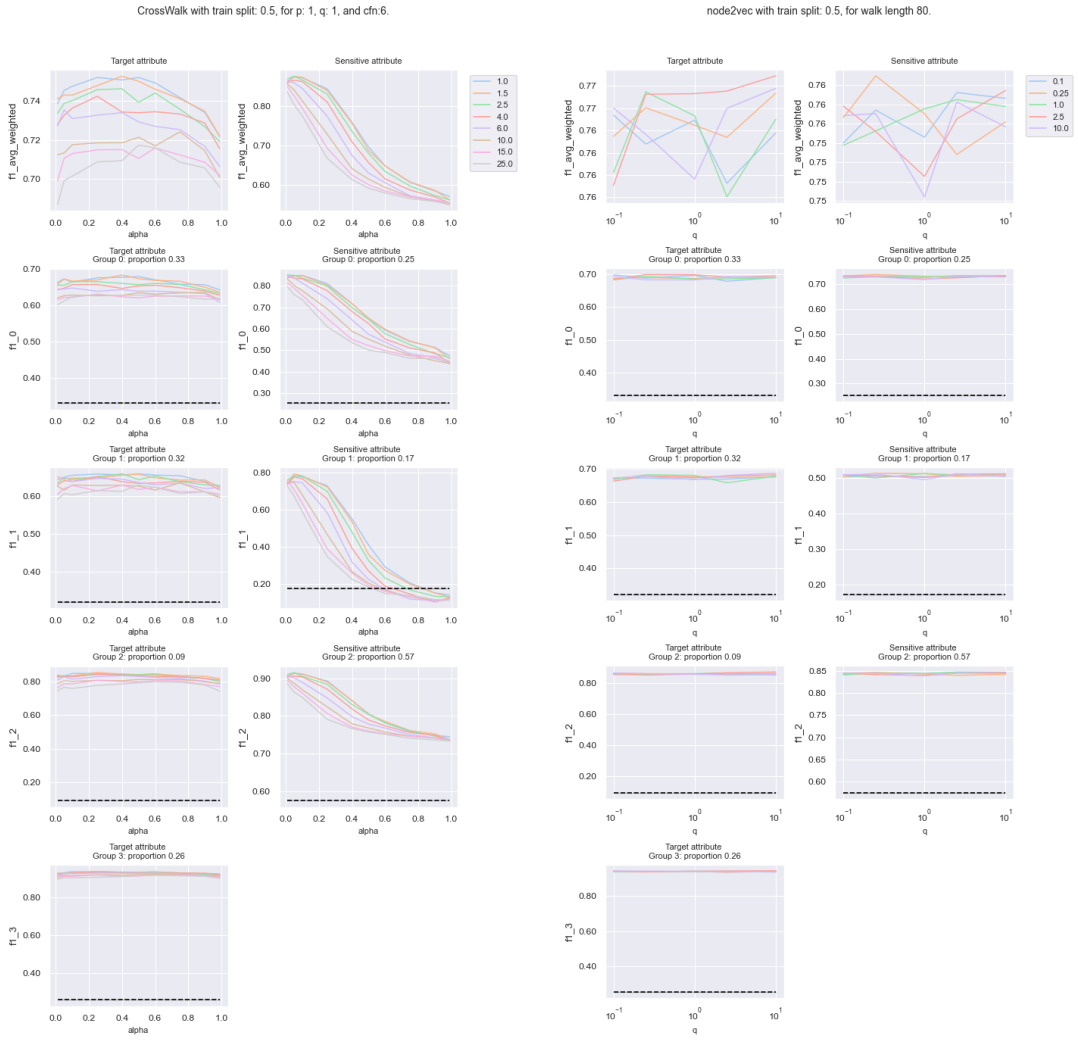
Results Overview for the *Mixed* Sub-Graph (c)

Table 5.3 shows the results on *Mixed* sub-graph (c). The results have similar tendencies for both attributes and the embeddings generated by the varying parameters on both algorithms. The *Mixed* setting, the adjacent neighborhoods of a big city, see section 4.1 seem to make it harder to predict the regions as the target attribute. Altogether the results are not as clear as in the previous settings (a) and (b) but clearly show similar tendencies.

5.2 Parameter Sensitivity

In the following, we provide a sensitivity analysis for the parameters of the two algorithms. For this, we illustrate the sensitivity of the parameters walk-length and q for *node2vec* and the parameters pre-walk length and α for *CrossWalk*. We plotted each setting for the three sub-graph settings as follows: The y-axis of the plots shows the F1-score for the classification of both attributes, colored orange for the sensitive attribute and blue for the target attribute. The x-axis represents the respective parameter values as categories, and the columns represent the relative training set size.

Figure 5.10 shows the sensitivity of the walk-length parameter and figure 5.11 shows the sensitivity of the parameter q for *node2vec*. Both figures represent boxplots varying over all other parameter settings for *node2vec*.



(a) *CrossWalk* (b) *node2vec*

Figure 5.6: Group related results compared for *node2vec* and *CrossWalk* on the *semi-distinct* sub-graph (b), each on the target attribute and the sensitive attribute for multiple parameters.

	Mean		Median		Min		Max	
	CrossWalk	node2vec	CrossWalk	node2vec	CrossWalk	node2vec	CrossWalk	node2vec
$F1_{weighted}$	0.7588	0.7904	0.7461	0.7906	0.6792	0.7819	0.8759	0.7945
$F1_{macro}$	0.5956	0.6533	0.5736	0.6532	0.4532	0.6433	0.8028	0.6611
$F1_{micro}$	0.7845	0.8033	0.7696	0.8034	0.7267	0.7907	0.8813	0.8076
Acc.	0.7845	0.8033	0.7696	0.8034	0.7267	0.7907	0.8813	0.8076
Var.	0.0011	0.0004	0.0010	0.0004	0.0001	0.0002	0.0027	0.0008
$F1_{group0}$	0.5847	0.6844	0.5738	0.6842	0.3839	0.6672	0.8080	0.6965
$F1_{group1}$	0.3230	0.3833	0.2685	0.3833	0.1189	0.3693	0.6706	0.4091
$F1_{group2}$	0.8790	0.8921	0.8724	0.8924	0.8441	0.8843	0.9299	0.8954
	Target attribute		Target attribute		Target attribute		Target attribute	
$F1_{weighted}$	0.4032	0.4529	0.3996	0.4530	0.3670	0.4453	0.4479	0.4610
$F1_{macro}$	0.3482	0.4055	0.3443	0.4057	0.3006	0.3967	0.4010	0.4129
$F1_{micro}$	0.4094	0.4586	0.4058	0.4586	0.3749	0.4509	0.4523	0.4673
Acc.	0.4094	0.4586	0.4058	0.4586	0.3749	0.4509	0.4523	0.4673
Var.	0.0038	0.0053	0.0037	0.0053	0.0007	0.0043	0.0075	0.0065
$F1_{group0}$	0.5005	0.5469	0.4968	0.5472	0.4708	0.5380	0.5420	0.5596
$F1_{group1}$	0.3589	0.3972	0.3574	0.3973	0.3182	0.3815	0.3994	0.4089
$F1_{group2}$	0.3511	0.4504	0.3438	0.4510	0.2721	0.4330	0.4462	0.4642
$F1_{group3}$	0.3484	0.3905	0.3470	0.3902	0.2988	0.3809	0.3922	0.4010
$F1_{group4}$	0.1821	0.2423	0.1790	0.2415	0.0943	0.2137	0.2622	0.2744

Table 5.3: Results of a classification on a 50% training-set and five-fold cross-validation on the *Mixed* sub-graph (c).

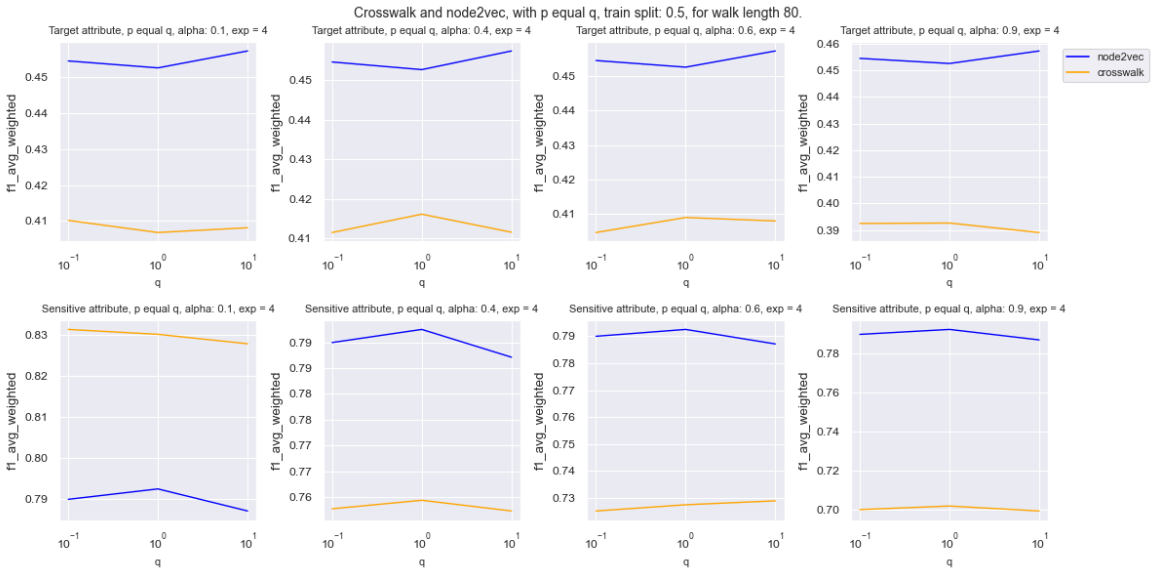


Figure 5.7: Results compared for *node2vec* and *CrossWalk* with varying parameter settings on the *Mixed* sub-graph (c), each on the target attribute and the sensitive attribute for multiple parameters.

(a) *CrossWalk* sampling parameter: p: 2.5, q: 2.5

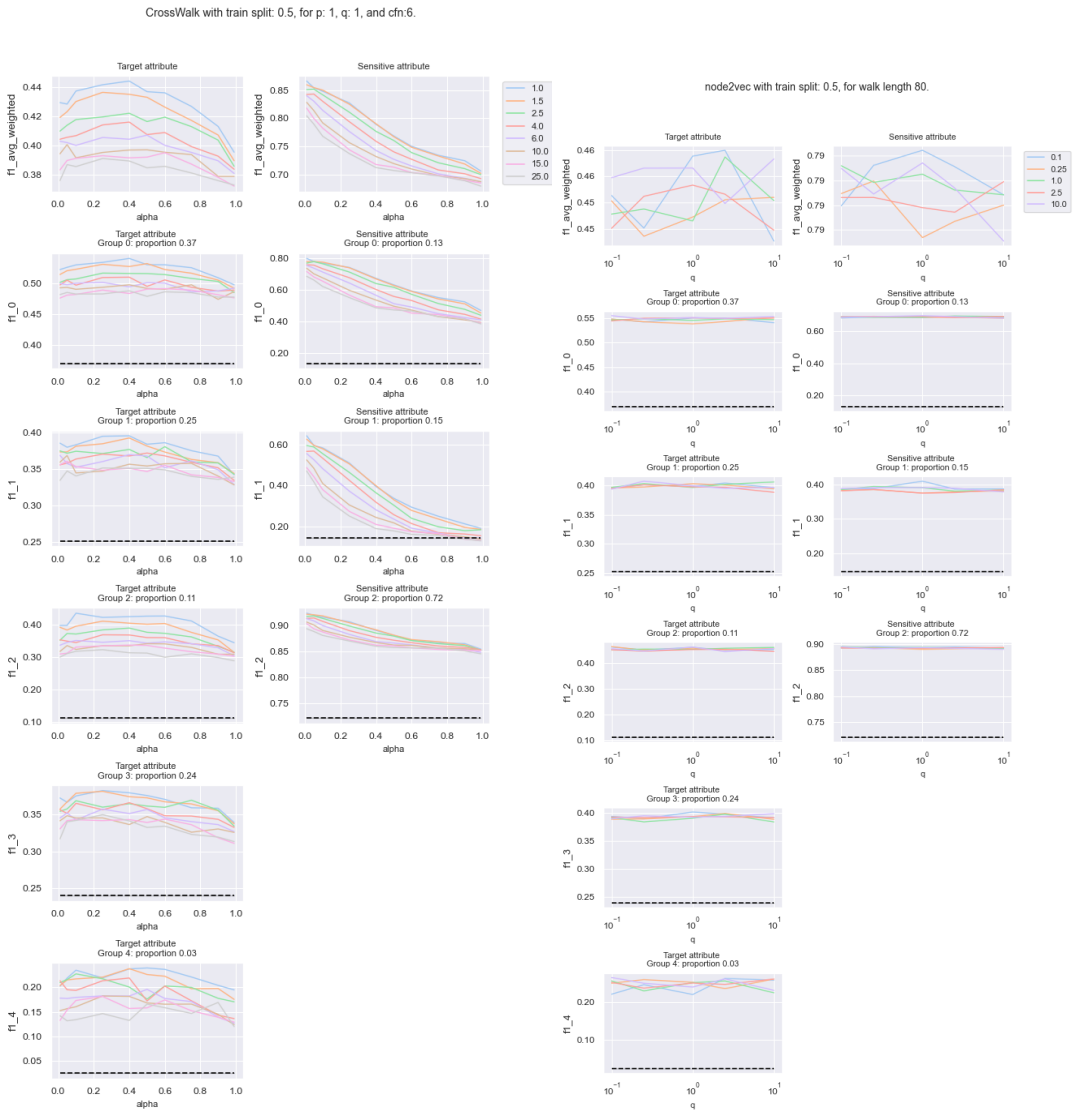
(b) *CrossWalk* sampling parameter: p: 1, q: 1, alpha: 0.75, exp: 6

Figure 5.8: Node embedding according to the sensitive groups generated with *CrossWalk* for the *Mixed* sub-graph (c), with reduced dimensionality using t-SNE.

Figure 5.12 shows the sensitivity of the pre-walk length parameter as a scatterplot, with colored attributes, various sizes for exp, and various signs for α for *CrossWalk*. Figure 5.13 shows the sensitivity of the parameter α for *CrossWalk* as boxplot.

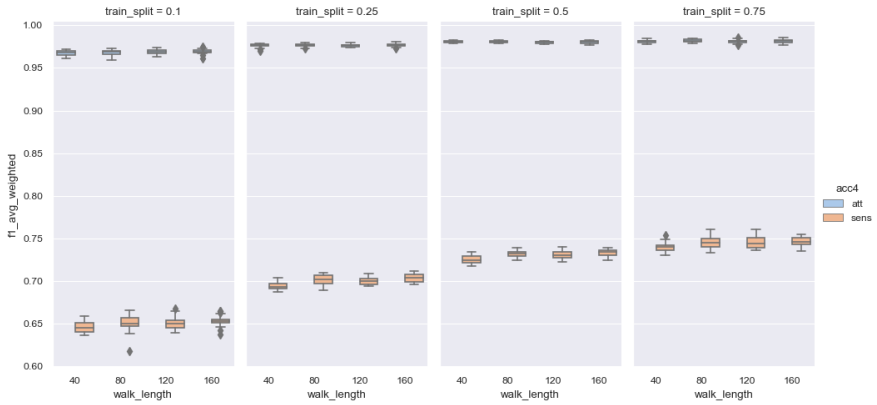
The most considerable Effect on the classification results is caused by the training set size and the graph structure. The walklenght for *node2vec* seems not to have a significant effect for a longer walk than 80, which fit the observation the authors made in [GL16]. The parameter q for *node2vec*, as well the other *node2vec* parameters, seem to have only a slight effect on the results in the evaluated settings. The same is observable for the Effect of the pre-walk length to calculate the cfn parameter for *CrossWalk*.

Hence, the results for the parameter α for *CrossWalk* show an apparent effect on the results. In each classification setting and on each graph structure, *CrossWalk* embeddings outperform *node2vec* in both directions, with the higher prediction quality and the higher invariance, both regarding the sensitive attribute, with alpha as the significant parameter. Concerning the target attribute, there is a trade-off between accuracy and invariance, which is also observable in the results of [KKB+22].

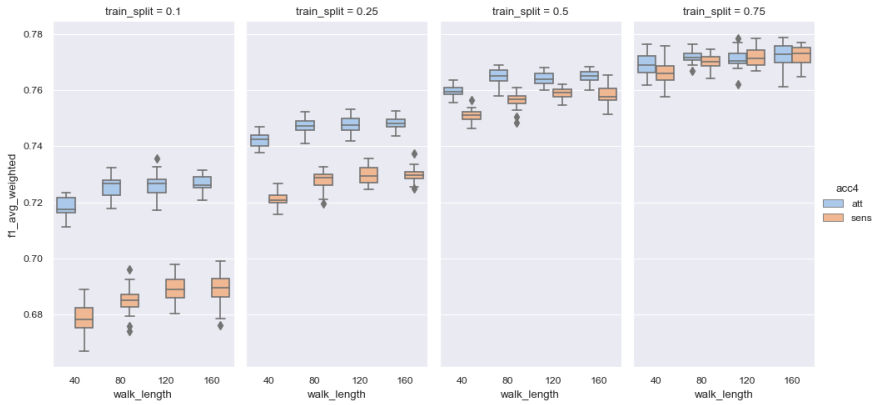


(a) *CrossWalk* (b) *node2vec*

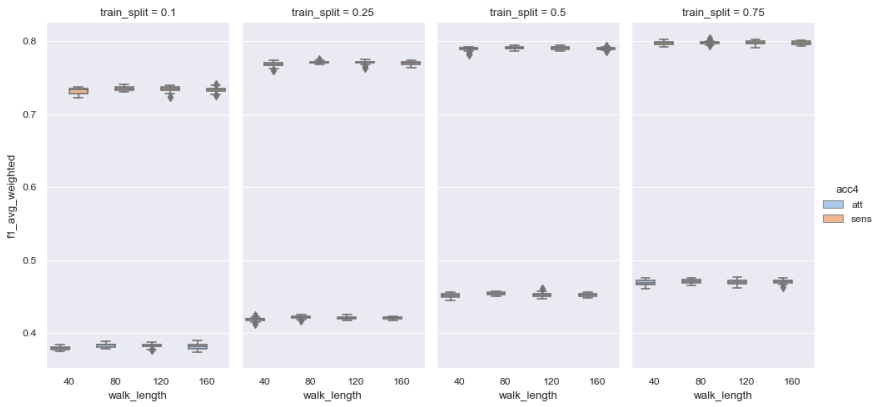
Figure 5.9: Group related results compared for *node2vec* and *CrossWalk* on the *Mixed* sub-graph (c), each on the target attribute and the sensitive attribute for multiple parameters.



(a) Effect of walk length on *node2veck* samplings for the *distibc* sub-graph (a).

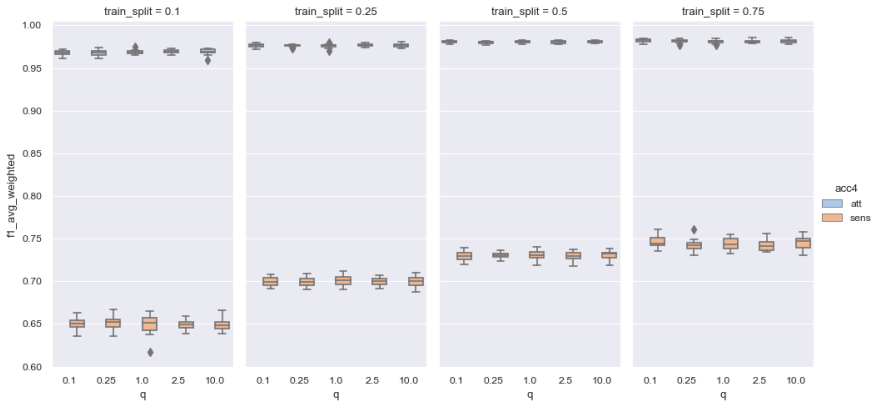


(b) Effect of walk length on *node2veck* samplings for the *semi-distibc* sub-graph (b).

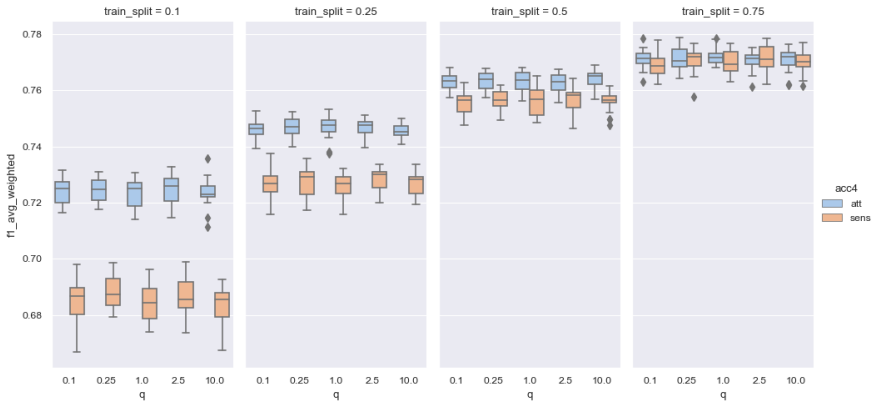


(c) Effect of walk length on *node2veck* samplings for the *mixed* sub-graph (c).

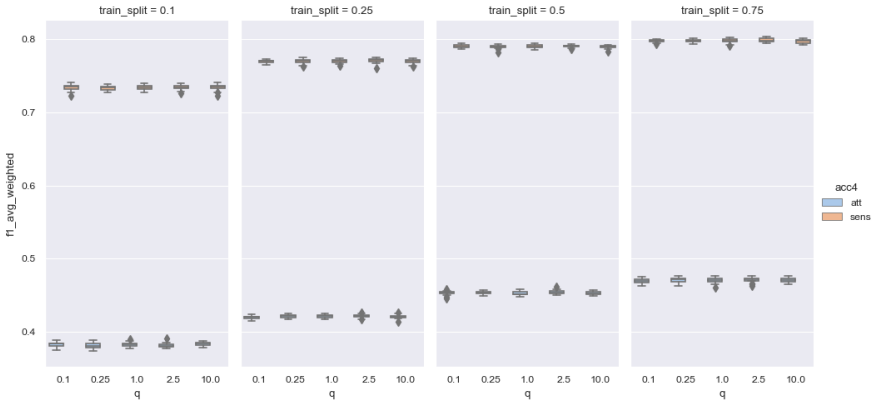
Figure 5.10: Effect of wavelength on *node2veck* samplings compared for the target and sensitive attributes.



(a) Effect of parameter q on *node2veck* samplings for the *distinct* sub-graph (a)

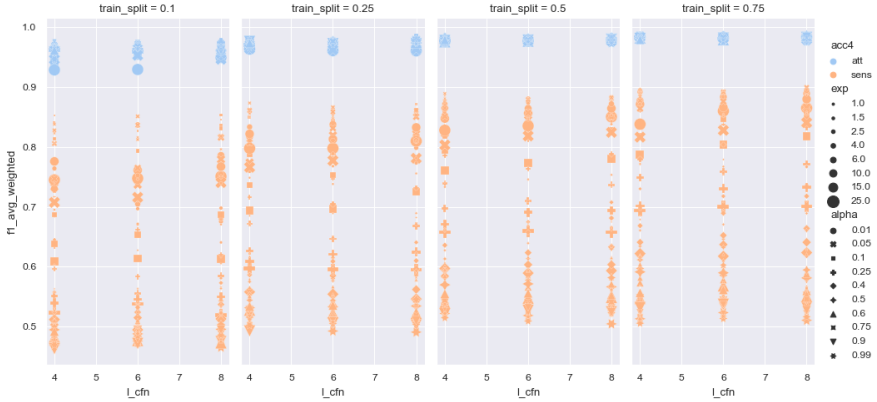


(b) Effect of parameter q on *node2veck* samplings for the *semi-distibc* sub-graph (b).

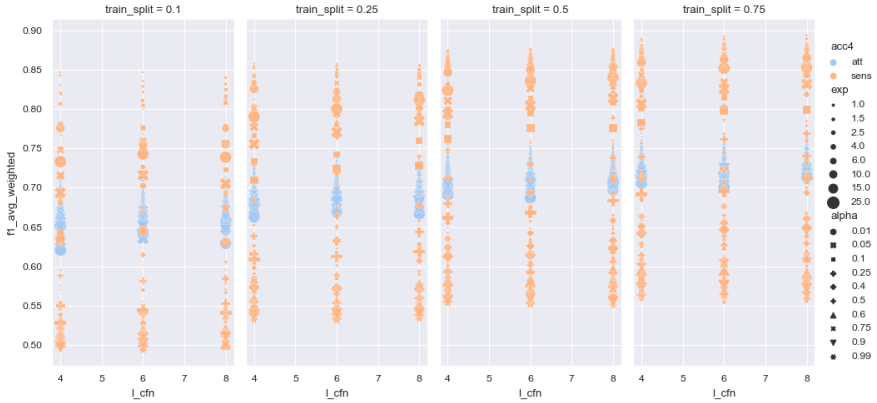


(c) Effect of parameter q on *node2veck* samplings for the *mixed* sub-graph (c).

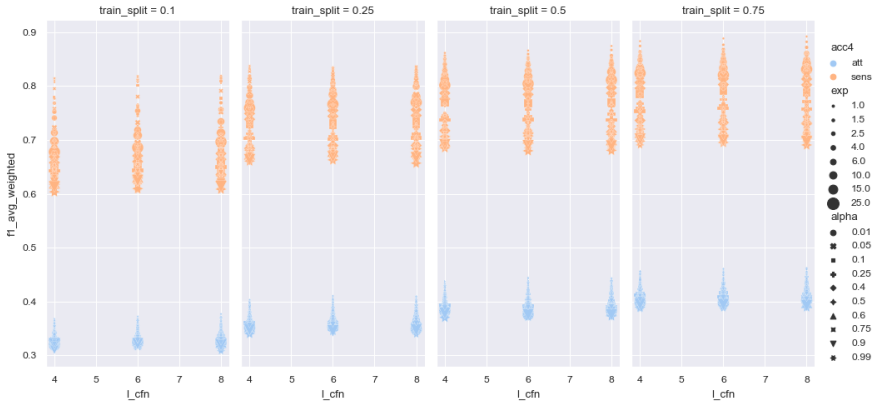
Figure 5.11: Effect of parameter q on *node2veck* samplings compared for the target and sensitive attributes.



(a) Effect of pre-walk length on *CrossWalk* samplings for the *distinct* sub-graph (a)

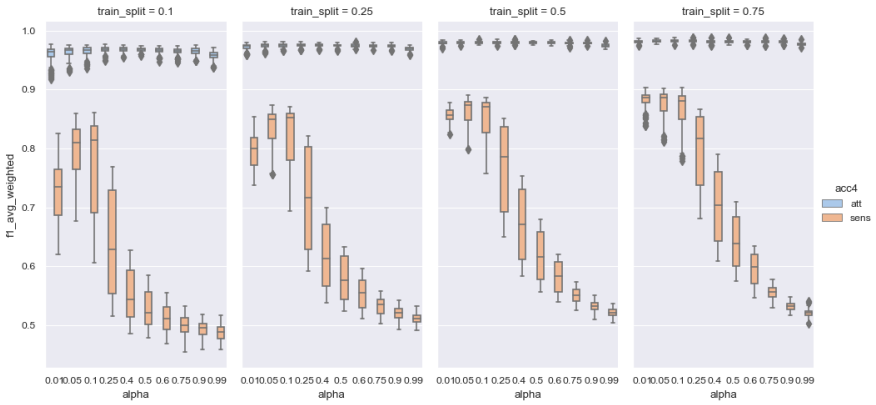


(b) Effect of pre-walk length on *CrossWalk* samplings for the *semi-distibc* sub-graph (b).

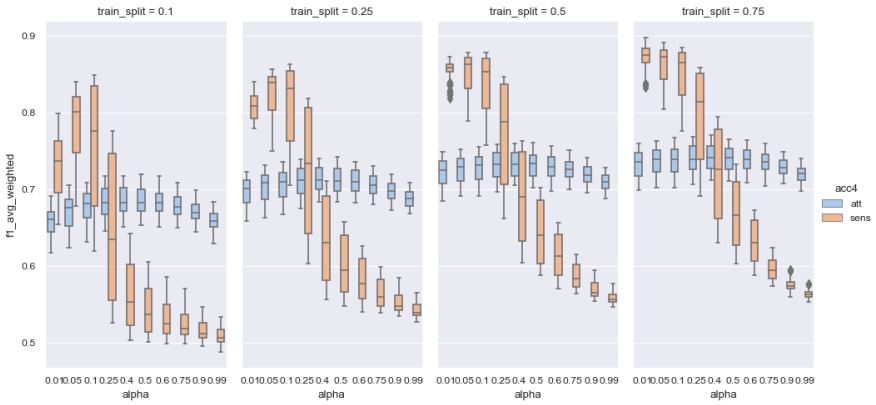


(c) Effect of pre-walk length on *CrossWalk* samplings for the *mixed* sub-graph (c).

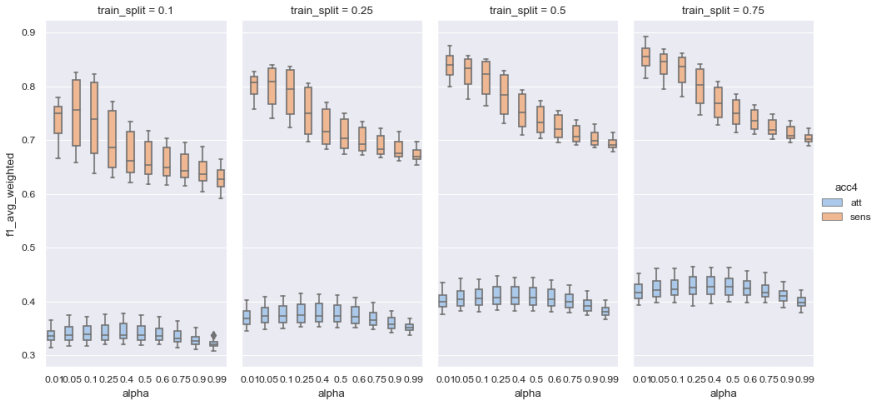
Figure 5.12: effect of pre-walk length on *CrossWalk* samplings compared for the target attribute and the sensitive attribute.



(a) Effect of parameter α on *CrossWalk* samplings for the *distinct* sub-graph (a)



(b) Effect of parameter α on *CrossWalk* samplings for the *semi-distibc* sub-graph (b).



(c) Effect of parameter α on *CrossWalk* samplings for the *mixed* sub-graph (c).

Figure 5.13: Effect of parameter α on *CrossWalk* samplings compared for the target attribute and the sensitive attribute.

6 Conclusion

This work examines how graph-sampling-based methods generate reasonable representations of real-world social networks. For this, we first outlined the theoretical foundations of the underlying areas to introduce the random walk-based graph sampling methods regarding the related work. After that, we reviewed our experimental setup.

Nevertheless, the results of our experiments suggest that the choice of the sampling method and parameter selection is crucial for the quality of the resulting graph embedding. Biased random walks significantly affect the outcome of node embeddings concerning node classification as a downstream task. Therefore, the biasing of such random walks is a good choice for sampling methods to control certain aspects of the graph representations regarding hiding or revealing graph structure-based information, such as targeted attributes and sensitive attributes.

The downstream task of classification of the underlying graph structure, as well as the training set size, affects the outcome of predictions significantly and, therefore, represents a substantial factor besides the sampling method and the parameter selection. The tendencies of the hyperparameter effects on the embedding quality have shown to be consistent in all examined cases, and the effects of the *CrossWalk* sampling on specified parameters are significant. There is a clear trade-off between the accuracy and the fairness of the resulting embeddings. Still, the difference in the prediction quality regarding the target attribute between *CrossWalk* and *node2vec* is relatively minor compared to the difference in the invariance tendencies of the embeddings.

Hence, taking into account that the prediction of the sensitive attribute for smaller alpha values becomes significantly more accurate with *CrossWalk* generated graph representations compared to the *node2vec*, we can conclude that the *CrossWalk* random walk biasing is a good approach to achieve not only invariant embeddings but also better prediction accuracy. This holds particularly for minority classes, depending on the parameter settings. Our evaluation showed that the effect of random walk biasing on graph embeddings is significant in several real-world-based social network structures and controllable through hyperparameter settings.

Further research on the topic of node embeddings and fairness is needed to find best-case approximations for the trade-off between fairness and accuracy and to figure out further if the *CrossWalk* random walk biasing may be suitable to achieve invariant embeddings of

social networks and to maximize prediction accuracy, even on graphs from other domains. Additionally, it would be interesting to see how the *CrossWalk* random walk interacts with certain specific graph structures and how the interaction with parameter settings influences the outcome.

Regarding further development, it seems worth looking into the extension of *CrossWalk* to graphs with mostly unlabeled nodes by predicting the unknown related groups for the up following group-based random walk biasing. This is particularly important against exploiting social network user attributes that were never given by the user but are hidden in the graph structure. Finally, considering the results of the *CrossWalk* random walk biasing in favor of fairness, we can conclude that the effect of random walk biasing on embedding is significant and controllable through hyperparameter settings.

7 Appendix

	Mean Sensitive attribute		Median Sensitive attribute		Min Sensitive attribute		Max Sensitive attribute	
	CrossWalk	node2vec	CrossWalk	node2vec	CrossWalk	node2vec	CrossWalk	node2vec
$F1_{weighted}$	0.6039	0.6499	0.5383	0.6501	0.4550	0.6175	0.8610	0.6674
$F1_{macro}$	0.5406	0.5838	0.4571	0.5838	0.3793	0.5525	0.8463	0.6097
$F1_{micro}$	0.6355	0.6706	0.5786	0.6707	0.4933	0.6354	0.8627	0.6931
Acc.	0.6355	0.6706	0.5786	0.6707	0.4933	0.6354	0.8627	0.6931
Var.	0.0044	0.0041	0.0017	0.0038	0.0001	0.0009	0.0258	0.0115
$Precision_{weighted}$	0.6143	0.6493	0.5495	0.6490	0.4517	0.6187	0.8625	0.6701
$Precision_{micro}$	0.6355	0.6706	0.5786	0.6707	0.4933	0.6354	0.8627	0.6931
$Precision_{macro}$	0.5860	0.6038	0.5191	0.6034	0.4064	0.5683	0.8586	0.6355
$Recall_{weighted}$	0.6355	0.6706	0.5786	0.6707	0.4933	0.6354	0.8627	0.6931
$Recall_{micro}$	0.6355	0.6706	0.5786	0.6707	0.4933	0.6354	0.8627	0.6931
$Recall_{macro}$	0.5583	0.5934	0.4840	0.5931	0.4114	0.5635	0.8486	0.6167
$F1_{group0}$	0.6570	0.7316	0.6041	0.7321	0.4662	0.6856	0.8833	0.7504
$F1_{group1}$	0.2578	0.2775	0.1031	0.2781	0.0274	0.2235	0.7885	0.3456
$F1_{group2}$	0.7070	0.7423	0.6694	0.7426	0.5646	0.7155	0.8824	0.7660
Rel. $Support_{group0}$	0.3478	0.3478	0.3478	0.3478	0.3478	0.3478	0.3478	0.3478
Rel. $Support_{group1}$	0.1908	0.1908	0.1908	0.1908	0.1908	0.1908	0.1908	0.1908
Rel. $Support_{group2}$	0.4615	0.4615	0.4615	0.4615	0.4615	0.4615	0.4615	0.4615
	Mean Target attribute		Median Target attribute		Min Target attribute		Max Target attribute	
	CrossWalk	node2vec	CrossWalk	node2vec	CrossWalk	node2vec	CrossWalk	node2vec
$F1_{weighted}$	0.9647	0.9683	0.9665	0.9688	0.9175	0.9596	0.9777	0.9752
$F1_{macro}$	0.9517	0.9562	0.9539	0.9563	0.8811	0.9445	0.9702	0.9652
$F1_{micro}$	0.9642	0.9680	0.9663	0.9685	0.9136	0.9592	0.9775	0.9749
Acc.	0.9642	0.9680	0.9663	0.9685	0.9136	0.9592	0.9775	0.9749
Var.	0.0003	0.0001	0.0002	0.0001	0.0000	0.0000	0.0052	0.0004
$Precision_{weighted}$	0.9665	0.9694	0.9681	0.9698	0.9279	0.9614	0.9784	0.9762
$Precision_{micro}$	0.9642	0.9680	0.9663	0.9685	0.9136	0.9592	0.9775	0.9749
$Precision_{macro}$	0.9473	0.9518	0.9494	0.9515	0.8663	0.9343	0.9766	0.9670
$Recall_{weighted}$	0.9642	0.9680	0.9663	0.9685	0.9136	0.9592	0.9775	0.9749
$Recall_{micro}$	0.9642	0.9680	0.9663	0.9685	0.9136	0.9592	0.9775	0.9749
$Recall_{macro}$	0.9586	0.9620	0.9599	0.9619	0.9084	0.9539	0.9767	0.9716
$F1_{group0}$	0.9098	0.9168	0.9142	0.9177	0.7622	0.8920	0.9521	0.9351
$F1_{group1}$	0.9767	0.9792	0.9780	0.9798	0.9463	0.9693	0.9865	0.9836
$F1_{group2}$	0.9686	0.9725	0.9703	0.9729	0.9077	0.9620	0.9842	0.9807
Rel. $Support_{group0}$	0.1316	0.1316	0.1316	0.1316	0.1316	0.1316	0.1316	0.1316
Rel. $Support_{group1}$	0.4731	0.4731	0.4731	0.4731	0.4731	0.4731	0.4731	0.4731
Rel. $Support_{group2}$	0.3953	0.3953	0.3953	0.3953	0.3953	0.3953	0.3953	0.3953

Table 7.1: Results of a classification with a 10% training-set and five-fold cross-validation on the *Distinct* sub-graph (a)

	Mean Sensitive attribute		Median Sensitive attribute		Min Sensitive attribute		Max Sensitive attribute	
	CrossWalk	node2vec	CrossWalk	node2vec	CrossWalk	node2vec	CrossWalk	node2vec
$F1_{weighted}$	0.6521	0.6998	0.5954	0.7000	0.4911	0.6877	0.8734	0.7118
$F1_{macro}$	0.5957	0.6448	0.5215	0.6441	0.4178	0.6293	0.8608	0.6604
$F1_{micro}$	0.6771	0.7119	0.6350	0.7121	0.5257	0.7014	0.8740	0.7221
Acc.	0.6771	0.7119	0.6350	0.7121	0.5257	0.7014	0.8740	0.7221
Var.	0.0029	0.0030	0.0014	0.0028	0.0001	0.0015	0.0156	0.0051
$Precision_{weighted}$	0.6560	0.6981	0.6107	0.6981	0.4894	0.6859	0.8739	0.7108
$Precision_{micro}$	0.6771	0.7119	0.6350	0.7121	0.5257	0.7014	0.8740	0.7221
$Precision_{macro}$	0.6290	0.6589	0.5749	0.6592	0.4444	0.6457	0.8635	0.6736
$Recall_{weighted}$	0.6771	0.7119	0.6350	0.7121	0.5257	0.7014	0.8740	0.7221
$Recall_{micro}$	0.6771	0.7119	0.6350	0.7121	0.5257	0.7014	0.8740	0.7221
$Recall_{macro}$	0.6064	0.6461	0.5373	0.6458	0.4402	0.6320	0.8607	0.6608
$F1_{group0}$	0.7034	0.7667	0.6725	0.7672	0.5266	0.7529	0.8932	0.7807
$F1_{group1}$	0.3422	0.3905	0.1842	0.3898	0.0623	0.3516	0.8044	0.4234
$F1_{group2}$	0.7415	0.7772	0.7140	0.7764	0.6071	0.7631	0.8910	0.7900
Rel. $Support_{group0}$	0.3480	0.3480	0.3480	0.3480	0.3480	0.3480	0.3480	0.3480
Rel. $Support_{group1}$	0.1907	0.1907	0.1907	0.1907	0.1907	0.1907	0.1907	0.1907
Rel. $Support_{group2}$	0.4613	0.4613	0.4613	0.4613	0.4613	0.4613	0.4613	0.4613
	Mean Target attribute		Median Target attribute		Min Target attribute		Max Target attribute	
	CrossWalk	node2vec	CrossWalk	node2vec	CrossWalk	node2vec	CrossWalk	node2vec
$F1_{weighted}$	0.9741	0.9766	0.9747	0.9768	0.9587	0.9703	0.9815	0.9806
$F1_{macro}$	0.9660	0.9690	0.9670	0.9691	0.9419	0.9601	0.9768	0.9749
$F1_{micro}$	0.9740	0.9766	0.9747	0.9768	0.9581	0.9701	0.9814	0.9806
Acc.	0.9740	0.9766	0.9747	0.9768	0.9581	0.9701	0.9814	0.9806
Var.	0.0001	0.0001	0.0001	0.0000	0.0000	0.0000	0.0011	0.0002
$Precision_{weighted}$	0.9747	0.9770	0.9752	0.9773	0.9604	0.9710	0.9816	0.9809
$Precision_{micro}$	0.9740	0.9766	0.9747	0.9768	0.9581	0.9701	0.9814	0.9806
$Precision_{macro}$	0.9675	0.9717	0.9686	0.9723	0.9320	0.9591	0.9851	0.9820
$Recall_{weighted}$	0.9740	0.9766	0.9747	0.9768	0.9581	0.9701	0.9814	0.9806
$Recall_{micro}$	0.9740	0.9766	0.9747	0.9768	0.9581	0.9701	0.9814	0.9806
$Recall_{macro}$	0.9654	0.9672	0.9656	0.9672	0.9468	0.9592	0.9792	0.9751
$F1_{group0}$	0.9399	0.9448	0.9418	0.9453	0.8845	0.9284	0.9658	0.9579
$F1_{group1}$	0.9815	0.9836	0.9819	0.9834	0.9647	0.9780	0.9892	0.9887
$F1_{group2}$	0.9767	0.9788	0.9772	0.9792	0.9560	0.9695	0.9853	0.9856
Rel. $Support_{group0}$	0.1318	0.1318	0.1318	0.1318	0.1318	0.1318	0.1318	0.1318
Rel. $Support_{group1}$	0.4732	0.4732	0.4732	0.4732	0.4732	0.4732	0.4732	0.4732
Rel. $Support_{group2}$	0.3950	0.3950	0.3950	0.3950	0.3950	0.3950	0.3950	0.3950

Table 7.2: Results of a classification with a 25% training-set and five-fold cross-validation on the *Distinct* sub-graph (a)

	Mean Sensitive attribute		Median Sensitive attribute		Min Sensitive attribute		Max Sensitive attribute	
	CrossWalk	node2vec	CrossWalk	node2vec	CrossWalk	node2vec	CrossWalk	node2vec
$F1_{weighted}$	0.6846	0.7302	0.6464	0.7305	0.5046	0.7181	0.8901	0.7406
$F1_{macro}$	0.6343	0.6818	0.5766	0.6824	0.4301	0.6649	0.8795	0.6954
$F1_{micro}$	0.7048	0.7387	0.6738	0.7392	0.5398	0.7269	0.8905	0.7490
Acc.	0.7048	0.7387	0.6738	0.7392	0.5398	0.7269	0.8905	0.7490
Var.	0.0026	0.0032	0.0013	0.0031	0.0000	0.0015	0.0111	0.0059
$Precision_{weighted}$	0.6850	0.7291	0.6536	0.7297	0.4989	0.7157	0.8918	0.7401
$Precision_{micro}$	0.7048	0.7387	0.6738	0.7392	0.5398	0.7269	0.8905	0.7490
$Precision_{macro}$	0.6580	0.6938	0.6217	0.6941	0.4454	0.6791	0.8834	0.7079
$Recall_{weighted}$	0.7048	0.7387	0.6738	0.7392	0.5398	0.7269	0.8905	0.7490
$Recall_{micro}$	0.7048	0.7387	0.6738	0.7392	0.5398	0.7269	0.8905	0.7490
$Recall_{macro}$	0.6420	0.6807	0.5863	0.6811	0.4532	0.6659	0.8797	0.6946
$F1_{group0}$	0.7329	0.7896	0.7176	0.7899	0.5505	0.7750	0.9059	0.8022
$F1_{group1}$	0.4070	0.4576	0.2605	0.4583	0.0799	0.4143	0.8310	0.4880
$F1_{group2}$	0.7630	0.7980	0.7435	0.7980	0.6241	0.7867	0.9060	0.8094
Rel. $Support_{group0}$	0.3479	0.3479	0.3479	0.3479	0.3479	0.3479	0.3479	0.3479
Rel. $Support_{group1}$	0.1907	0.1907	0.1907	0.1907	0.1907	0.1907	0.1907	0.1907
Rel. $Support_{group2}$	0.4613	0.4613	0.4613	0.4613	0.4613	0.4613	0.4613	0.4613
	Mean Target attribute		Median Target attribute		Min Target attribute		Max Target attribute	
	CrossWalk	node2vec	CrossWalk	node2vec	CrossWalk	node2vec	CrossWalk	node2vec
$F1_{weighted}$	0.9792	0.9806	0.9795	0.9807	0.9691	0.9774	0.9858	0.9832
$F1_{macro}$	0.9730	0.9748	0.9734	0.9751	0.9567	0.9692	0.9826	0.9789
$F1_{micro}$	0.9792	0.9806	0.9795	0.9807	0.9687	0.9773	0.9858	0.9831
Acc.	0.9792	0.9806	0.9795	0.9807	0.9687	0.9773	0.9858	0.9831
Var.	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0002	0.0001
$Precision_{weighted}$	0.9795	0.9809	0.9797	0.9810	0.9700	0.9776	0.9860	0.9835
$Precision_{micro}$	0.9792	0.9806	0.9795	0.9807	0.9687	0.9773	0.9858	0.9831
$Precision_{macro}$	0.9752	0.9778	0.9761	0.9780	0.9470	0.9666	0.9881	0.9852
$Recall_{weighted}$	0.9792	0.9806	0.9795	0.9807	0.9687	0.9773	0.9858	0.9831
$Recall_{micro}$	0.9792	0.9806	0.9795	0.9807	0.9687	0.9773	0.9858	0.9831
$Recall_{macro}$	0.9713	0.9723	0.9714	0.9717	0.9555	0.9661	0.9822	0.9798
$F1_{group0}$	0.9527	0.9560	0.9540	0.9562	0.9157	0.9421	0.9745	0.9658
$F1_{group1}$	0.9851	0.9862	0.9854	0.9862	0.9747	0.9819	0.9914	0.9903
$F1_{group2}$	0.9810	0.9822	0.9813	0.9822	0.9681	0.9770	0.9881	0.9871
Rel. $Support_{group0}$	0.1318	0.1318	0.1318	0.1318	0.1318	0.1318	0.1318	0.1318
Rel. $Support_{group1}$	0.4731	0.4731	0.4731	0.4731	0.4731	0.4731	0.4731	0.4731
Rel. $Support_{group2}$	0.3951	0.3951	0.3951	0.3951	0.3951	0.3951	0.3951	0.3951

Table 7.3: Results of a classification with a 50% training-set and five-fold cross-validation on the *Distinct* sub-graph (a)

	Mean Sensitive attribute		Median Sensitive attribute		Min Sensitive attribute		Max Sensitive attribute	
	CrossWalk	node2vec	CrossWalk	node2vec	CrossWalk	node2vec	CrossWalk	node2vec
$F1_{weighted}$	0.6999	0.7440	0.6735	0.7436	0.5029	0.7303	0.9034	0.7609
$F1_{macro}$	0.6530	0.6991	0.6127	0.6997	0.4310	0.6792	0.8938	0.7201
$F1_{micro}$	0.7173	0.7507	0.6961	0.7498	0.5280	0.7389	0.9034	0.7665
Acc.	0.7173	0.7507	0.6961	0.7498	0.5280	0.7389	0.9034	0.7665
Var.	0.0029	0.0041	0.0019	0.0040	0.0001	0.0017	0.0122	0.0080
$Precision_{weighted}$	0.6986	0.7440	0.6832	0.7437	0.4879	0.7295	0.9046	0.7612
$Precision_{micro}$	0.7173	0.7507	0.6961	0.7498	0.5280	0.7389	0.9034	0.7665
$Precision_{macro}$	0.6704	0.7093	0.6553	0.7084	0.4292	0.6930	0.9038	0.7285
$Recall_{weighted}$	0.7173	0.7507	0.6961	0.7498	0.5280	0.7389	0.9034	0.7665
$Recall_{micro}$	0.7173	0.7507	0.6961	0.7498	0.5280	0.7389	0.9034	0.7665
$Recall_{macro}$	0.6602	0.6980	0.6163	0.6988	0.4473	0.6792	0.8971	0.7180
$F1_{group0}$	0.7460	0.7988	0.7400	0.7984	0.5415	0.7798	0.9193	0.8215
$F1_{group1}$	0.4406	0.4910	0.3287	0.4896	0.0656	0.4399	0.8577	0.5352
$F1_{group2}$	0.7724	0.8073	0.7596	0.8065	0.6165	0.7955	0.9189	0.8213
Rel. $Support_{group0}$	0.3478	0.3478	0.3478	0.3478	0.3478	0.3478	0.3478	0.3478
Rel. $Support_{group1}$	0.1909	0.1909	0.1909	0.1909	0.1909	0.1909	0.1909	0.1909
Rel. $Support_{group2}$	0.4613	0.4613	0.4613	0.4613	0.4613	0.4613	0.4613	0.4613
	Mean Target attribute		Median Target attribute		Min Target attribute		Max Target attribute	
	CrossWalk	node2vec	CrossWalk	node2vec	CrossWalk	node2vec	CrossWalk	node2vec
$F1_{weighted}$	0.9812	0.9815	0.9814	0.9815	0.9710	0.9765	0.9886	0.9858
$F1_{macro}$	0.9755	0.9759	0.9758	0.9757	0.9585	0.9677	0.9858	0.9829
$F1_{micro}$	0.9812	0.9815	0.9815	0.9815	0.9706	0.9764	0.9886	0.9858
Acc.	0.9812	0.9815	0.9815	0.9815	0.9706	0.9764	0.9886	0.9858
Var.	0.0001	0.0001	0.0001	0.0001	0.0000	0.0000	0.0002	0.0002
$Precision_{weighted}$	0.9815	0.9817	0.9816	0.9817	0.9718	0.9768	0.9887	0.9859
$Precision_{micro}$	0.9812	0.9815	0.9815	0.9815	0.9706	0.9764	0.9886	0.9858
$Precision_{macro}$	0.9773	0.9780	0.9781	0.9782	0.9495	0.9651	0.9906	0.9878
$Recall_{weighted}$	0.9812	0.9815	0.9815	0.9815	0.9706	0.9764	0.9886	0.9858
$Recall_{micro}$	0.9812	0.9815	0.9815	0.9815	0.9706	0.9764	0.9886	0.9858
$Recall_{macro}$	0.9741	0.9743	0.9744	0.9744	0.9575	0.9667	0.9864	0.9818
$F1_{group0}$	0.9571	0.9581	0.9579	0.9575	0.9170	0.9393	0.9804	0.9735
$F1_{group1}$	0.9867	0.9869	0.9869	0.9872	0.9768	0.9820	0.9941	0.9933
$F1_{group2}$	0.9827	0.9828	0.9829	0.9829	0.9709	0.9748	0.9910	0.9885
Rel. $Support_{group0}$	0.1318	0.1318	0.1318	0.1318	0.1318	0.1318	0.1318	0.1318
Rel. $Support_{group1}$	0.4728	0.4728	0.4728	0.4728	0.4728	0.4728	0.4728	0.4728
Rel. $Support_{group2}$	0.3954	0.3954	0.3954	0.3954	0.3954	0.3954	0.3954	0.3954

Table 7.4: Results of a classification with a 75% training-set and five-fold cross-validation on the *Distinct* sub-graph (a)

	Mean Sensitive attribute		Median Sensitive attribute		Min Sensitive attribute		Max Sensitive attribute	
	CrossWalk	node2vec	CrossWalk	node2vec	CrossWalk	node2vec	CrossWalk	node2vec
$F1_{weighted}$	0.6105	0.6854	0.5519	0.6857	0.4881	0.6669	0.8492	0.6990
$F1_{macro}$	0.5092	0.6124	0.4271	0.6142	0.3639	0.5836	0.8204	0.6306
$F1_{micro}$	0.6495	0.6902	0.6053	0.6903	0.5134	0.6737	0.8502	0.7045
Acc.	0.6495	0.6902	0.6053	0.6903	0.5134	0.6737	0.8502	0.7045
Var.	0.0023	0.0013	0.0018	0.0011	0.0001	0.0003	0.0116	0.0041
$Precision_{weighted}$	0.6131	0.6856	0.5569	0.6869	0.4800	0.6657	0.8495	0.6989
$Precision_{micro}$	0.6495	0.6902	0.6053	0.6903	0.5134	0.6737	0.8502	0.7045
$Precision_{macro}$	0.5618	0.6191	0.4979	0.6196	0.3979	0.5961	0.8286	0.6361
$Recall_{weighted}$	0.6495	0.6902	0.6053	0.6903	0.5134	0.6737	0.8502	0.7045
$Recall_{micro}$	0.6495	0.6902	0.6053	0.6903	0.5134	0.6737	0.8502	0.7045
$Recall_{macro}$	0.5141	0.6137	0.4366	0.6156	0.3844	0.5874	0.8219	0.6350
$F1_{group0}$	0.5129	0.6493	0.4372	0.6500	0.3196	0.6224	0.8127	0.6686
$F1_{group1}$	0.2535	0.4008	0.1148	0.4041	0.0408	0.3269	0.7629	0.4429
$F1_{group2}$	0.7611	0.7871	0.7379	0.7869	0.6498	0.7665	0.8940	0.7990
Rel. $Support_{group0}$	0.2521	0.2521	0.2521	0.2521	0.2521	0.2521	0.2521	0.2521
Rel. $Support_{group1}$	0.1735	0.1735	0.1735	0.1735	0.1735	0.1735	0.1735	0.1735
Rel. $Support_{group2}$	0.5745	0.5745	0.5745	0.5745	0.5745	0.5745	0.5745	0.5745

Table 7.5: Results of a classification with a 10% training-set and five-fold cross-validation on the *Semi-distinct* sub-graph (b)

	Mean Sensitive attribute		Median Sensitive attribute		Min Sensitive attribute		Max Sensitive attribute	
	CrossWalk	node2vec	CrossWalk	node2vec	CrossWalk	node2vec	CrossWalk	node2vec
$F1_{weighted}$	0.6641	0.7271	0.6165	0.7284	0.5273	0.7157	0.8630	0.7375
$F1_{macro}$	0.5754	0.6584	0.5098	0.6605	0.4003	0.6422	0.8359	0.6706
$F1_{micro}$	0.6927	0.7320	0.6483	0.7328	0.5679	0.7218	0.8637	0.7427
Acc.	0.6927	0.7320	0.6483	0.7328	0.5679	0.7218	0.8637	0.7427
Var.	0.0018	0.0006	0.0015	0.0006	0.0001	0.0003	0.0059	0.0010
$Precision_{weighted}$	0.6636	0.7250	0.6152	0.7266	0.5160	0.7124	0.8626	0.7350
$Precision_{micro}$	0.6927	0.7320	0.6483	0.7328	0.5679	0.7218	0.8637	0.7427
$Precision_{macro}$	0.6177	0.6657	0.5685	0.6663	0.4352	0.6512	0.8450	0.6787
$Recall_{weighted}$	0.6927	0.7320	0.6483	0.7328	0.5679	0.7218	0.8637	0.7427
$Recall_{micro}$	0.6927	0.7320	0.6483	0.7328	0.5679	0.7218	0.8637	0.7427
$Recall_{macro}$	0.5730	0.6558	0.5078	0.6586	0.4136	0.6379	0.8304	0.6680
$F1_{group0}$	0.5928	0.6970	0.5472	0.6988	0.3884	0.6806	0.8301	0.7153
$F1_{group1}$	0.3405	0.4560	0.2140	0.4582	0.0728	0.4114	0.7772	0.4791
$F1_{group2}$	0.7930	0.8221	0.7673	0.8221	0.7046	0.8135	0.9045	0.8294
Rel. $Support_{group0}$	0.2522	0.2522	0.2522	0.2522	0.2522	0.2522	0.2522	0.2522
Rel. $Support_{group1}$	0.1734	0.1734	0.1734	0.1734	0.1734	0.1734	0.1734	0.1734
Rel. $Support_{group2}$	0.5744	0.5744	0.5744	0.5744	0.5744	0.5744	0.5744	0.5744

Table 7.6: Results of a classification with a 25% training-set and five-fold cross-validation on the *Semi-distinct* sub-graph (b)

	Mean Sensitive attribute		Median Sensitive attribute		Min Sensitive attribute		Max Sensitive attribute	
	CrossWalk	node2vec	CrossWalk	node2vec	CrossWalk	node2vec	CrossWalk	node2vec
$F1_{weighted}$	0.6979	0.7562	0.6609	0.7568	0.5465	0.7463	0.8787	0.7652
$F1_{macro}$	0.6185	0.6922	0.5656	0.6937	0.4227	0.6782	0.8538	0.7039
$F1_{micro}$	0.7197	0.7611	0.6885	0.7613	0.5906	0.7519	0.8796	0.7696
Acc.	0.7197	0.7611	0.6885	0.7613	0.5906	0.7519	0.8796	0.7696
Var.	0.0019	0.0004	0.0018	0.0003	0.0001	0.0002	0.0061	0.0006
$Precision_{weighted}$	0.6954	0.7540	0.6658	0.7548	0.5340	0.7433	0.8787	0.7630
$Precision_{micro}$	0.7197	0.7611	0.6885	0.7613	0.5906	0.7519	0.8796	0.7696
$Precision_{macro}$	0.6511	0.7016	0.6223	0.7018	0.4528	0.6887	0.8830	0.7126
$Recall_{weighted}$	0.7197	0.7611	0.6885	0.7613	0.5906	0.7519	0.8796	0.7696
$Recall_{micro}$	0.7197	0.7611	0.6885	0.7613	0.5906	0.7519	0.8796	0.7696
$Recall_{macro}$	0.6138	0.6874	0.5580	0.6896	0.4331	0.6709	0.8474	0.7011
$F1_{group0}$	0.6396	0.7296	0.6116	0.7302	0.4247	0.7115	0.8614	0.7410
$F1_{group1}$	0.4036	0.5025	0.2986	0.5057	0.0946	0.4759	0.7968	0.5261
$F1_{group2}$	0.8123	0.8445	0.7919	0.8447	0.7237	0.8386	0.9160	0.8496
Rel. $Support_{group0}$	0.2521	0.2521	0.2521	0.2521	0.2521	0.2521	0.2521	0.2521
Rel. $Support_{group1}$	0.1735	0.1735	0.1735	0.1735	0.1735	0.1735	0.1735	0.1735
Rel. $Support_{group2}$	0.5744	0.5744	0.5744	0.5744	0.5744	0.5744	0.5744	0.5744

Table 7.7: Results of a classification with a 50% training-set and five-fold cross-validation on the *Semi-distinct* sub-graph (b)

	Mean Sensitive attribute		Median Sensitive attribute		Min Sensitive attribute		Max Sensitive attribute	
	CrossWalk	node2vec	CrossWalk	node2vec	CrossWalk	node2vec	CrossWalk	node2vec
$F1_{weighted}$	0.7135	0.7701	0.6917	0.7699	0.5533	0.7577	0.8980	0.7785
$F1_{macro}$	0.6392	0.7094	0.6100	0.7093	0.4312	0.6931	0.8765	0.7204
$F1_{micro}$	0.7310	0.7741	0.7105	0.7741	0.5797	0.7643	0.9000	0.7825
Acc.	0.7310	0.7741	0.7105	0.7741	0.5797	0.7643	0.9000	0.7825
Var.	0.0022	0.0004	0.0020	0.0004	0.0001	0.0002	0.0091	0.0009
$Precision_{weighted}$	0.7097	0.7687	0.6936	0.7685	0.5374	0.7551	0.9005	0.7770
$Precision_{micro}$	0.7310	0.7741	0.7105	0.7741	0.5797	0.7643	0.9000	0.7825
$Precision_{macro}$	0.6636	0.7171	0.6529	0.7174	0.4428	0.7068	0.9034	0.7277
$Recall_{weighted}$	0.7310	0.7741	0.7105	0.7741	0.5797	0.7643	0.9000	0.7825
$Recall_{micro}$	0.7310	0.7741	0.7105	0.7741	0.5797	0.7643	0.9000	0.7825
$Recall_{macro}$	0.6354	0.7060	0.5989	0.7064	0.4417	0.6845	0.8619	0.7186
$F1_{group0}$	0.6618	0.7447	0.6487	0.7442	0.4448	0.7292	0.8795	0.7574
$F1_{group1}$	0.4358	0.5295	0.3750	0.5305	0.0981	0.4995	0.8249	0.5579
$F1_{group2}$	0.8200	0.8539	0.8065	0.8540	0.7198	0.8470	0.9315	0.8600
Rel. $Support_{group0}$	0.2519	0.2519	0.2519	0.2519	0.2519	0.2519	0.2519	0.2519
Rel. $Support_{group1}$	0.1735	0.1735	0.1735	0.1735	0.1735	0.1735	0.1735	0.1735
Rel. $Support_{group2}$	0.5745	0.5745	0.5745	0.5745	0.5745	0.5745	0.5745	0.5745
NaN	NaN							

Table 7.8: Results of a classification with a 75% training-set and five-fold cross-validation on the *Semi-distinct* sub-graph (b)

	Mean		Median		Min		Max	
	Sensitive attribute		Sensitive attribute		Sensitive attribute		Sensitive attribute	
	CrossWalk	node2vec	CrossWalk	node2vec	CrossWalk	node2vec	CrossWalk	node2vec
$F1_{weighted}$	0.6842	0.7340	0.6682	0.7345	0.5911	0.7233	0.8254	0.7414
$F1_{macro}$	0.4809	0.5660	0.4493	0.5667	0.3705	0.5526	0.7189	0.5765
$F1_{micro}$	0.7167	0.7509	0.7088	0.7515	0.5970	0.7374	0.8327	0.7592
Acc.	0.7167	0.7509	0.7088	0.7515	0.5970	0.7374	0.8327	0.7592
Var.	0.0012	0.0006	0.0013	0.0006	0.0001	0.0003	0.0028	0.0011
$Precision_{weighted}$	0.6758	0.7272	0.6595	0.7272	0.5952	0.7198	0.8234	0.7343
$Precision_{micro}$	0.7167	0.7509	0.7088	0.7515	0.5970	0.7374	0.8327	0.7592
$Precision_{macro}$	0.5258	0.5826	0.5074	0.5835	0.3881	0.5661	0.7462	0.5963
$Recall_{weighted}$	0.7167	0.7509	0.7088	0.7515	0.5970	0.7374	0.8327	0.7592
$Recall_{micro}$	0.7167	0.7509	0.7088	0.7515	0.5970	0.7374	0.8327	0.7592
$Recall_{macro}$	0.4806	0.5701	0.4442	0.5701	0.3827	0.5568	0.7035	0.5801
$F1_{group0}$	0.4095	0.5792	0.3772	0.5805	0.2286	0.5538	0.7151	0.5955
$F1_{group1}$	0.2007	0.2606	0.1396	0.2601	0.0783	0.2388	0.5377	0.2841
$F1_{group2}$	0.8326	0.8582	0.8290	0.8588	0.7467	0.8476	0.9040	0.8645
Rel. $Support_{group0}$	0.1337	0.1337	0.1337	0.1337	0.1337	0.1337	0.1337	0.1337
Rel. $Support_{group1}$	0.1453	0.1453	0.1453	0.1453	0.1453	0.1453	0.1453	0.1453
Rel. $Support_{group2}$	0.7210	0.7210	0.7210	0.7210	0.7210	0.7210	0.7210	0.7210

Table 7.9: Results of a classification with a 10% training-set and five-fold cross-validation on the *Mixed* sub-graph (c)

	Mean		Median		Min		Max	
	Sensitive attribute		Sensitive attribute		Sensitive attribute		Sensitive attribute	
	CrossWalk	node2vec	CrossWalk	node2vec	CrossWalk	node2vec	CrossWalk	node2vec
$F1_{weighted}$	0.7323	0.7703	0.7184	0.7706	0.6542	0.7608	0.8408	0.7758
$F1_{macro}$	0.5518	0.6231	0.5260	0.6233	0.4231	0.6098	0.7445	0.6306
$F1_{micro}$	0.7629	0.7830	0.7494	0.7835	0.6916	0.7703	0.8473	0.7890
Acc.	0.7629	0.7830	0.7494	0.7835	0.6916	0.7703	0.8473	0.7890
Var.	0.0012	0.0005	0.0013	0.0005	0.0001	0.0002	0.0029	0.0009
$Precision_{weighted}$	0.7237	0.7633	0.7075	0.7634	0.6356	0.7555	0.8387	0.7686
$Precision_{micro}$	0.7629	0.7830	0.7494	0.7835	0.6916	0.7703	0.8473	0.7890
$Precision_{macro}$	0.6055	0.6417	0.5803	0.6421	0.4615	0.6267	0.7780	0.6532
$Recall_{weighted}$	0.7629	0.7830	0.7494	0.7835	0.6916	0.7703	0.8473	0.7890
$Recall_{micro}$	0.7629	0.7830	0.7494	0.7835	0.6916	0.7703	0.8473	0.7890
$Recall_{macro}$	0.5369	0.6153	0.5085	0.6157	0.4210	0.6051	0.7275	0.6218
$F1_{group0}$	0.5190	0.6466	0.5019	0.6470	0.3163	0.6217	0.7545	0.6593
$F1_{group1}$	0.2716	0.3434	0.2141	0.3436	0.1023	0.3232	0.5698	0.3567
$F1_{group2}$	0.8648	0.8793	0.8590	0.8796	0.8189	0.8702	0.9126	0.8834
Rel. $Support_{group0}$	0.1337	0.1337	0.1337	0.1337	0.1337	0.1337	0.1337	0.1337
Rel. $Support_{group1}$	0.1453	0.1453	0.1453	0.1453	0.1453	0.1453	0.1453	0.1453
Rel. $Support_{group2}$	0.7210	0.7210	0.7210	0.7210	0.7210	0.7210	0.7210	0.7210

Table 7.10: Results of a classification with a 25% training-set and five-fold cross-validation on the *Mixed* sub-graph (c)

	Mean		Median		Min		Max	
	Sensitive attribute		Sensitive attribute		Sensitive attribute		Sensitive attribute	
	CrossWalk	node2vec	CrossWalk	node2vec	CrossWalk	node2vec	CrossWalk	node2vec
$F1_{weighted}$	0.7588	0.7904	0.7461	0.7906	0.6792	0.7819	0.8759	0.7945
$F1_{macro}$	0.5956	0.6533	0.5736	0.6532	0.4532	0.6433	0.8028	0.6611
$F1_{micro}$	0.7845	0.8033	0.7696	0.8034	0.7267	0.7907	0.8813	0.8076
Acc.	0.7845	0.8033	0.7696	0.8034	0.7267	0.7907	0.8813	0.8076
Var.	0.0011	0.0004	0.0010	0.0004	0.0001	0.0002	0.0027	0.0008
$Precision_{weighted}$	0.7516	0.7841	0.7375	0.7843	0.6627	0.7763	0.8767	0.7886
$Precision_{micro}$	0.7845	0.8033	0.7696	0.8034	0.7267	0.7907	0.8813	0.8076
$Precision_{macro}$	0.6479	0.6799	0.6209	0.6805	0.5137	0.6607	0.8501	0.6879
$Recall_{weighted}$	0.7845	0.8033	0.7696	0.8034	0.7267	0.7907	0.8813	0.8076
$Recall_{micro}$	0.7845	0.8033	0.7696	0.8034	0.7267	0.7907	0.8813	0.8076
$Recall_{macro}$	0.5759	0.6388	0.5523	0.6385	0.4443	0.6321	0.7693	0.6451
$F1_{group0}$	0.5847	0.6844	0.5738	0.6842	0.3839	0.6672	0.8080	0.6965
$F1_{group1}$	0.3230	0.3833	0.2685	0.3833	0.1189	0.3693	0.6706	0.4091
$F1_{group2}$	0.8790	0.8921	0.8724	0.8924	0.8441	0.8843	0.9299	0.8954
Rel. $Support_{group0}$	0.1337	0.1337	0.1337	0.1337	0.1337	0.1337	0.1337	0.1337
Rel. $Support_{group1}$	0.1454	0.1454	0.1454	0.1454	0.1454	0.1454	0.1454	0.1454
Rel. $Support_{group2}$	0.7209	0.7209	0.7209	0.7209	0.7209	0.7209	0.7209	0.7209

Table 7.11: Results of a classification with a 50% training-set and five-fold cross-validation on the *Mixed* sub-graph (c)

	Mean		Median		Min		Max	
	Sensitive attribute		Sensitive attribute		Sensitive attribute		Sensitive attribute	
	CrossWalk	node2vec	CrossWalk	node2vec	CrossWalk	node2vec	CrossWalk	node2vec
$F1_{weighted}$	0.7725	0.7981	0.7599	0.7981	0.6898	0.7917	0.8929	0.8040
$F1_{macro}$	0.6205	0.6668	0.6007	0.6668	0.4700	0.6571	0.8300	0.6757
$F1_{micro}$	0.7937	0.8095	0.7791	0.8095	0.7332	0.7983	0.8969	0.8150
Acc.	0.7937	0.8095	0.7791	0.8095	0.7332	0.7983	0.8969	0.8150
Var.	0.0011	0.0005	0.0010	0.0004	0.0001	0.0001	0.0040	0.0013
$Precision_{weighted}$	0.7652	0.7925	0.7515	0.7924	0.6749	0.7860	0.8938	0.7987
$Precision_{micro}$	0.7937	0.8095	0.7791	0.8095	0.7332	0.7983	0.8969	0.8150
$Precision_{macro}$	0.6628	0.6903	0.6386	0.6904	0.5263	0.6766	0.8709	0.7014
$Recall_{weighted}$	0.7937	0.8095	0.7791	0.8095	0.7332	0.7983	0.8969	0.8150
$Recall_{micro}$	0.7937	0.8095	0.7791	0.8095	0.7332	0.7983	0.8969	0.8150
$Recall_{macro}$	0.6026	0.6540	0.5842	0.6538	0.4601	0.6448	0.8003	0.6609
$F1_{group0}$	0.6204	0.6999	0.6160	0.7001	0.4184	0.6799	0.8275	0.7121
$F1_{group1}$	0.3564	0.4049	0.3071	0.4045	0.1291	0.3877	0.7324	0.4278
$F1_{group2}$	0.8847	0.8957	0.8779	0.8956	0.8500	0.8875	0.9392	0.9000
Rel. $Support_{group0}$	0.1337	0.1337	0.1337	0.1337	0.1337	0.1337	0.1337	0.1337
Rel. $Support_{group1}$	0.1454	0.1454	0.1454	0.1454	0.1454	0.1454	0.1454	0.1454
Rel. $Support_{group2}$	0.7209	0.7209	0.7209	0.7209	0.7209	0.7209	0.7209	0.7209

Table 7.12: Results of a classification with a 75% training-set and five-fold cross-validation on the *Mixed* sub-graph (c)

Bibliography

[BH19] Avishek Bose and William Hamilton. Compositional Fairness Constraints for Graph Embeddings. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 715–724. PMLR, 09–15 Jun 2019. URL: https://proceedings.mlr.press/v97/bose19a.html.

[CH20] Simon Caton and Christian Haas. Fairness in Machine Learning: A Survey, 2020. URL: https://arxiv.org/abs/2010.04053, doi:10.48550/ARXIV.2010.04053.

[CLL22] Manvi Choudhary, Charlotte Laclau, and Christine Largeron. A Survey on Fairness for Machine Learning on [graphs], 2022. URL: https://arxiv.org/abs/2205.05396, doi:10.48550/ARXIV.2205.05396.

[CN06] Gabor Csardi and Tamas Nepusz. The igraph software package for complex network research. *InterJournal*, Complex Systems:1695, 2006. URL: https://igraph.org.

[Coh18] Elior Cohen. node2vec: Python3 implementation of the node2vec algorithm based on node2vec: Scalable Feature Learning for Networks. A. Grover, J. Leskovec. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2016. https://github.com/eliorc/node2vec, 2018.

[Fac04] Facebook Inc. Facebook, 2004. URL: https://www.facebook.com/.

[GL16] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016. doi:10.1145/2939672.2939754.

[HGER+12] Keith Henderson, Brian Gallagher, Tina Eliassi-Rad, Hanghang Tong, Sugato Basu, Leman Akoglu, Danai Koutra, Christos Faloutsos, and Lei Li. Rolx: structural role extraction & mining in large graphs. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1231–1239, 2012. URL: http://briangallagher.net/pubs/henderson-etal-kdd2012.pdf.

[HL13] Pili Hu and Wing Cheong Lau. A Survey and Taxonomy of Graph Sampling, 2013. URL: https://arxiv.org/abs/1308.5865, doi:10.48550/ARXIV.1308.5865.

[HM15] Mohammad Hossin and Sulaiman M.N. A Review on Evaluation Metrics for Data Classification Evaluations. *International Journal of Data Mining & Knowledge Management Process*, 5:01–11, 03 2015. doi:10.5121/ijdkp.2015.5201.

[HYL17] William L. Hamilton, Rex Ying, and Jure Leskovec. Representation Learning on Graphs: Methods and Applications. *CoRR*, abs/1709.05584, 2017. URL: http://arxiv.org/abs/1709.05584, arXiv:1709.05584, doi:10.48550/ARXIV.1709.05584.

[ITAC19] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum. Label Propagation for Deep Semi-Supervised Learning. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. URL: https://openaccess.thecvf.com/content_CVPR_2019/html/Iscen_Label_Propagation_for_Deep_Semi-Supervised_Learning_CVPR_2019_paper.html.

[JWHT13] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning*, volume 112. Springer, 2013.

[KBŠBŠ20] Damir Krstinić, Maja Braović, Ljiljana Šerić, and Dunja Božić-Štulić. Multi-label classifier performance evaluation with confusion matrix. *Comput Sci Inf Technol*, 10:1–14, 2020. doi:DOI:10.5121/csit.2020.100801.

[Kha21] Ahmad Khajehnejad. CrossWalk: Source code and data used for the CrossWalk paper. https://github.com/ahmadkhajehnejad/CrossWalk, 2021.

[KKB+22] Ahmad Khajehnejad, Moein Khajehnejad, Mahmoudreza Babaei, Krishna P. Gummadi, Adrian Weller, and Baharan Mirzasoleiman. CrossWalk: Fairness-Enhanced Node Representation Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(11):11963–11970, June 2022. URL: https://ojs.aaai.org/index.php/AAAI/article/view/21454, doi:10.1609/aaai.v36i11.21454.

[KKBK21] Matthias Kuppler, Christoph Kern, Ruben L. Bach, and Frauke Kreuter. Distributive Justice and Fairness Metrics in Automated Decision-making: How Much Overlap Is There?, 2021. URL: https://arxiv.org/abs/2105.01441, doi:10.48550/ARXIV.2105.01441.

[KLPS12] Gang Kou, YANQUN LU, Yi Peng, and Yong Shi. Evaluation of classification algorithms using MCDM and rank correlation. *International Journal of Information Technology & Decision Making*, 11, 04 2012. doi:10.1142/S0219622012500095.

[LK14] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data, Jun 2014.

[LL10] Gregory F Lawler and Vlada Limic. *Random walk: a modern introduction*, volume 123. Cambridge University Press, 2010.

[LRKS18] Joshua R. Loftus, Chris Russell, Matt J. Kusner, and Ricardo Silva. Causal Reasoning for Algorithmic Fairness, 2018. URL: https://arxiv.org/abs/1805.05859, doi:10.48550/ARXIV.1805.05859.

[MBW+19] Pankaj Mehta, Marin Bukov, Ching-Hao Wang, Alexandre GR Day, Clint Richardson, Charles K Fisher, and David J Schwab. A high-bias, low-variance introduction to machine learning for physicists. *Physics reports*, 810:1–124, 2019. doi:https://doi.org/10.1016/j.physrep.2019.03.001.

[MCC+03] Peter R Monge, Noshir S Contractor, Peter S Contractor, R Peter, S Noshir, et al. *Theories of Communication Networks*. Oxford University Press, USA, 2003.

[MCCD13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space, 2013. URL: https://arxiv.org/abs/1301.3781, doi:10.48550/ARXIV.1301.3781.

[PARS14] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014. doi:https://doi.org/10.1145/2623330.2623732.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. URL: https://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf.

[RSBZ19] Tahleen Rahman, Bartlomiej Surma, Michael Backes, and Yang Zhang. Fairwalk: Towards fair graph embedding. *International Joint Conference on Artificial Intelligence (IJCAI)*, 2019. URL: https://publications.cispa.saarland/

2933/.

[SSBD14] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[TMKM18] Anton Tsitsulin, Davide Mottin, Panagiotis Karras, and Emmanuel Müller. Verse: Versatile graph embeddings from similarity measures. In *Proceedings of the 2018 world wide web conference*, pages 539–548, 2018. doi:10.1145/3178876.3186120.

[TZ12] Lubos Takac and Michal Zabovsky. Data analysis in public social networks. In *International scientific conference and international workshop present day trends of innovations*, volume 1. Present Day Trends of Innovations Lamza Poland, 2012. URL: http://snap.stanford.edu/data/soc-pokec.pdf.

[VR18] Sahil Verma and Julia Rubin. Fairness Definitions Explained. In *2018 IEEE/ACM International Workshop on Software Fairness (FairWare)*, pages 1–7. IEEE, 2018. doi:10.1145/3194770.3194776.

[WCA+16] Yanhong Wu, Nan Cao, Daniel Archambault, Qiaomu Shen, Huamin Qu, and Weiwei Cui. Evaluation of graph sampling: A visualization perspective. *IEEE transactions on visualization and computer graphics*, 23(1):401–410, 2016. doi:10.1109/TVCG.2016.2598867.

[Wig19] Avi Wigderson. *Mathematics and Computation*. Princeton University Press, 2019.

[Xu21] Mengjia Xu. Understanding graph embedding methods and their applications. *SIAM Review*, 63(4):825–853, 2021. doi:10.1137/20M1386062.

[ZG02] Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation, 2002. URL: http://mlg.eng.cam.ac.uk/zoubin/papers/propagate.ps.gz.

[ZYZZ18] Daokun Zhang, Jie Yin, Xingquan Zhu, and Chengqi Zhang. Network representation learning: A survey. *IEEE transactions on Big Data*, 6(1):3–28, 2018. doi:10.1109/TBDATA.2018.2850013.