Network Embedding with Attribute Refinement

Tong Xiao
Zhejiang University, China
xiaot@zju.edu.cn

Yuan Fang
Singapore Management University, Singapore
yfang@smu.edu.sg

Hongxia Yang
Alibaba Group, China
yang.yhx@alibaba-inc.com

Vincent W. Zheng
WeBank, China
vincent.zheng@adsc-create.edu.sg

ABSTRACT

Network embedding has been an active research area given its effectiveness in mapping nodes to low-dimensional representations. While previous studies mostly focus on network topology, recent advances have shown that rich node-level information, known as attributes, often exist and can substantially benefit embedding based on the assumption of homophily: nodes often connect to other nodes similar to themselves. However, we find that inconsistencies often occur in node attributes in real-world data, which can undermine the homophily assumption and thus degrade the performance of attributed network embedding. To address this drawback, in this paper, we present a novel framework for unsupervised network embedding with attribute refinement. In particular, we propose a learnable filter to automatically refine the individual attributes of every node. To overcome the challenge of no supervision, we leverage homophily to guide the refinement—attributes should be fine-tuned in a way to reinforce the correlation with topology. Finally, we conduct extensive experiments on three benchmark real-world datasets, which show that our model significantly outperforms state-of-the-art methods on both node classification and link prediction tasks. Furthermore, we perform model analysis to demonstrate that our framework can effectively refine attributes.

CCS CONCEPTS

• **Computing methodologies** → **Learning latent representations**;
• **Information systems** → *Data mining*; *Social networks*.

KEYWORDS

Network embedding, attribute refinement, homophily

ACM Reference Format:
Tong Xiao, Yuan Fang, Hongxia Yang, and Vincent W. Zheng. 2020. Network Embedding with Attribute Refinement. In *The 16th International Workshop on Mining and Learning with Graphs.* ACM, New York, NY, USA, 8 pages.

MLG '20, Aug 24, 2020, Virtual Event, San Diego, CA, USA

ACM ISBN . . . $

1 INTRODUCTION

Many real-world problems involve a network or a graph, a powerful structure for modeling not only individual entities as its nodes, but also complex interplay among these nodes. Prominent examples include social networks that encompass inter-user relationships, and biological networks that capture protein-protein interactions. Such networks enable many crucial applications, for instance, personalized recommendation for users [13], query intent classification [6] and disease gene identification [2], which can often cast as instances of node classification, link prediction or network clustering. Solving these problems typically requires effective representations for each node in the network. While earlier studies relied on manual feature engineering to derive node representations, current research has achieved considerable success through network embedding [9, 22, 23, 25], often with significantly reduced manual efforts and increased performances.

In this paper, we study the problem of unsupervised network embedding with attributes. It aims to learn a mapping from each node to a low-dimensional and continuous vector, which can be subsequently used to support downstream applications. In particular, we focus on the *unsupervised* setting, where application-specific labels are not required. In other words, no manual effort is involved to learn the mappings, and the resulting vectors can be universally used by different applications. Traditionally, unsupervised embedding techniques [9, 22, 23, 25] only leverage network topology (*i.e.*, the relationships between nodes), missing out on the rich information associated with individual nodes. Such information almost always exist, such as the age or location of users in social networks, as well as the keywords or texts describing proteins. These node-level information are usually encoded as node *attributes* in a structured way. For example, categorical values may employ one-hot encoding, whereas texts may adopt the vector space model. These attributes can be used in conjunction with network topology to improve the embedding. In particular, node attributes are fundamentally important to sparse networks, which are widespread in real-world applications.

Several recent developments [11, 18, 29] have accounted for node attributes in an unsupervised setting, which generally outperform topology-only approaches. Their key insight boils down to the assumption of *homophily* [19, 20]—the tendency for nodes to relate to or interact with other nodes that are similar to themselves. This implies that node attributes constitute an important source of information to complement network topology. We illustrate the homophily assumption on three benchmark attributed networks, namely, Cora, Citeseer and Wiki in Figure 1. Homophily implies that neighbor pairs (*i.e.*, nodes directly linked to each other in the network) should

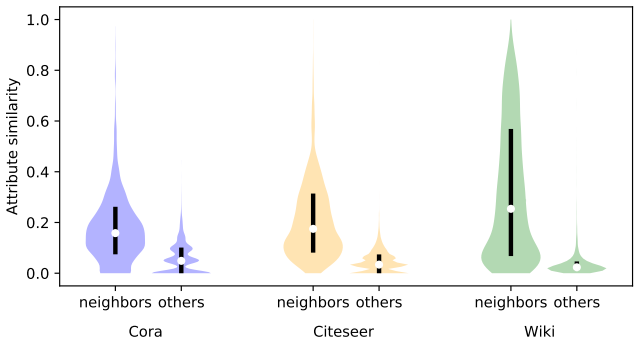

Figure 1: Homophily on three attributed networks (Cora, Citeseer and Wiki). On each network, two groups of node pairs are studied: neighbors and other (non-neighbor) pairs. For each group, the distribution of their attribute similarities is plotted and marked with the median, 20^{th} and 80^{th} percentiles.

have significantly more similar attributes than other non-neighbor pairs. We measure the attribute similarity of a node pair using the cosine similarity of their attribute vectors. The homophily effect clearly exists on all three networks, as a neighbor pair is far more likely to have higher attribute similarity than a non-neighbor pair.

However, the homophily assumption could be severely undermined when inconsistencies appear in node attributes, when the observed attribute values deviate from their presumably normal values to different extents. Such inconsistencies remain a prevalent issue in real-world data, although their interpretations often vary in different contexts. For instance, in social networks, user reported demographic attributes are often outdated, under-reported or deliberately misreported. Previous studies on Twitter [5, 16] show that only a small fraction of users (as few as 16%) report city-level locations, while most leave non-informative or generic locations (*e.g.*, "my home", "CA" or "Earth"). On the other hand, in protein interaction networks, while the keywords describing the proteins are often curated by experts [4], not all keywords are of equal relevance and importance. The varying degrees of relevance or importance can be deemed as a form of inconsistency. A similar interpretation exists on citation networks where articles have keywords as attributes.

Towards remediating the adverse impact of inconsistencies, we present a novel, unsupervised algorithm called Network Embedding with Attribute Refinement (NEAR). However, detecting and mitigating attribute inconsistencies are non-trivial, posing two major challenges. First, inconsistencies in attribute can occur in an ad-hoc and non-systematic manner. There is a need for fine-grained refinement designed for individual attributes and nodes. In particular, we propose to fine-tune each attribute of every node using a learnable filter, with the ability to refine attributes on a per-attribute, per-node basis. Second, as we assume an unsupervised setting, there exists no explicit guidance on modeling attribute inconsistencies. Note that nodes with attribute inconsistencies are not necessarily outliers, as they may have irregularities in only one or a few attributes. Furthermore, attributes can be extremely high dimensional, rendering traditional unsupervised algorithms for outlier detection ineffective. To compensate for the lack of supervision, we propose to refine attributes in a manner consistent with the homophily assumption.

To summarize, in this paper, we make the following contributions to attributed network embedding.

- We observe the need to address inconsistencies in node attributes for more effective and robust network embedding.
- We propose a novel, unsupervised framework NEAR for network embedding with attribute refinement, through a fully learnable filter guided by the homophily assumption.
- We conduct extensive experiments to demonstrate that NEAR can significantly outperform the state-of-the-art in both link prediction and node classification, and present an in-depth analysis on the underlying mechanism of NEAR.

2 RELATED WORK

Early research in network embedding has been inspired by word embedding [21] in natural language processing. In DeepWalk [22], Perozzi *et al.* observed that the degrees of nodes in a network follow a power law, similar to the frequencies of words occurring in natural languages. Leveraging on this property, DeepWalk performs random walks on the network to sample sequences of nodes, which are similar to sentences in language. Such "sentences" of nodes in fact leverage network topology: nodes closely connected to each other should have similar representations. Subsequently, node2vec [9] generalizes DeepWalk by adopting an improved random walk strategy. The basic idea is to explore the network with both breadth and depth-first search, and thus captures both local and global topology. For a similar purpose, several studies [23, 25, 27] design and model first and second-order proximity to preserve topology both locally and globally, based on various techniques such as deep autoencoders and matrix factorization. In particular, in community preserving network embedding [27], the proposed model incorporates not only microscopic topological structures such as first and second-order proximity, but also mesoscopic community structures. More recently, the macroscopic scale-free property exhibited by many real-world networks has been investigated [7], where a penalty on the degrees of nodes is enforced to follow the power law.

All of the above studies only deal with network topology. However, in real-world networks, nodes are often associated with rich information, which can be encoded into attributes. The underlying principle is hinged on the assumption of homophily [19, 20]: nodes tend to connect with other nodes similar to themselves. This assumption implies that node attributes form a valuable source of information to complement network topology. Recent studies [11, 18, 29, 31] have demonstrated many times that integrating node attributes are indeed beneficial, for both text information [29] and general attributes [11, 18, 32]. Some utilize matrix factorization [3, 11, 29] and some use deep neural networks [8, 18, 26, 32], but the essence boils down to integrating the two aspects of topology and attributes, so that nodes similar to each other in the two aspects share similar representations. Sometimes, this essence is materialized as explicit constraints. For instance, Li *et al.* [15] proposed a property-preserving model called PPNE, which employs inequality and numeric constraints to force node pairs with similar attributes to become closer in the learned representations.

However, in the above works, inconsistencies in node attributes are not considered and addressed. In a recent work [30], the node embeddings are directly built upon attributes. Their model projects attributes into low-dimensional representations, with parameters jointly learned with topology. Presumably, inconsistencies can be refined by a projection with right parameters, as opposed to models that train node embeddings directly. However, it does not allow for fine-grained refinement for each node and each attribute. In contrast, our proposed model NEAR leverages a fully learnable filter, which is able to refine node attributes on a per-attribute, per-node basis. Separately, Zhou *et al.* [33] observed the partial correlation problem between topological and attribute similarities, *i.e.*, the homophily effect is not always true. While not necessarily caused by inconsistencies (although they could be a significant factor), they proposed a relation ranking framework to prioritize node pairs that do satisfy the homophily assumption. In our approach, we learn a filter to directly refine attributes so that the homophily effect is reinforced. Finally, there also exist the related problem of partial observation [28], where we can only observe the topology of some nodes, or the attributes of some others. The issue of missing information is beyond the scope of this paper.

While the above models all focus on the unsupervised setting, there also exist semi-supervised or supervised approaches, where task-specific supervision is required. Examples include label informed attributed network embedding (LANE) [12], as well as recent graph neural networks such as graph convolutional networks [14], graph attention networks [24] and GraphSAGE [10]. While these studies do not assume any attribute inconsistencies, Liang *et al.* [17] has proposed a model called SEANO to deal with attribute noises in a semi-supervised manner.

3 PROPOSED APPROACH

In this section, we introduce our model NEAR, or Network Embedding with Attribute Refinement. We first include some preliminaries on the notations and problem definition, before presenting our proposed approach.

3.1 Preliminaries

Formally, we denote a network with attributes as $G = (V, E, A)$. Herein $V = \{v_1, v_2, ..., v_n\}$ represents the set of nodes, and $E \subseteq V \times V$ represents the set of edges, such that $(v_i, v_j) \in E$ if and only if v_i is linked to v_j. In addition, $A \in \mathbb{R}^{n \times m}$ is the attribute matrix, where n is the number of nodes and m is the number of attributes. More specifically, $A_{i,j}$ encodes the j^{th} attribute of node v_i, which can be binary or real-valued. Typically, categorical attributes can be converted into one hot vectors, and texts can be converted into a bag of words with binary, tf-idf or other weightings. The main notations used in this paper are summarized in Table 1.

As motivated in Section 1, node attributes often present inconsistencies. The j^{th} attribute of node v_i is said to be *inconsistent*, if $A_{i,j}$ is non-zero and deviates from its (latent) normal value. Note that when $A_{i,j}$ is zero, it is considered as a potential *missing* value, which is beyond the scope of this paper.

Table 1: Summary of main notations.

Notation	Description
G, V, E	network G, node set V and edge set E
$v_i \in V$	a node
$\mathcal{N}_i$	the context of node v_i
n, m	number of nodes and attributes, resp.
d	number of embedding dimensions
$A, \tilde{A} \in \mathbb{R}^{n \times m}$	attribute matrices, original or filtered, resp.
$U, U' \in \mathbb{R}^{n \times d}$	center or context node embeddings, resp.
$F \in \mathbb{R}^{n \times m}$	filter matrix
$W \in \mathbb{R}^{m \times d}$	weight matrix for attributes
$B \in \mathbb{R}^{n \times d}$	bias matrix for topology variations
$\mathbf{a}_i, \tilde{\mathbf{a}}_i, \mathbf{f}_i \in \mathbb{R}^m$	the i^{th} row vector in A, $\tilde{A}$ or F, resp.
$\mathbf{u}_i, \mathbf{u}'_i, \mathbf{w}_i, \mathbf{b}_i \in \mathbb{R}^d$	the i^{th} row vector in U or U', resp.

3.2 Node representation

Embedding with attributes. To begin with, we integrate attributes directly into each node's representation, without assuming any inconsistency in the attributes at the moment. Specifically, the representation matrix for nodes, $U \in \mathbb{R}^{n \times d}$, is defined as

$$U = AW + B, \tag{1}$$

where $W \in \mathbb{R}^{m \times d}$ and $B \in \mathbb{R}^{n \times d}$ are the weight and bias matrices, respectively, and d is the embedding dimension. Even under the homophily assumption, it is not expected that topology and attributes align perfectly [33]. In order to account for topology variations from attributes, the bias is node-specific rather than global to all nodes. Thus, B can be understood as capturing the topological variations from attributes. Note that the i^{th}-row in U corresponds to $\mathbf{u}_i$, the representation for the node v_i. Similarly, the i^{th}-row in B, or $\mathbf{b}_i$, is the bias for v_i.

Attribute filter. Next, we consider potential inconsistencies in the attributes. Arguably, the weight matrix W is able to mitigate inconsistencies in the j^{th} attribute by setting its j^{th} row to near-zero values, but this mechanism is far from ideal. In particular, only when the j^{th} attribute is inconsistent across all of the nodes in V, it makes sense to have $\mathbf{w}_j \approx \mathbf{0}$. Although there could be such attributes, in a more general scenario, inconsistencies can present in a more ad-hoc and less systematic manner.

Thus, we propose a fully learnable filter $F \in \mathbb{R}^{n \times m}$, which has the same shape as the attribute matrix A. Therefore, F can be applied to A in an element-wise manner, achieving fine-grained refinement on a per-node, per-attribute basis. More specifically, we extend Equation (1) to the following:

$$U = (A \circ \sigma(F))\, W + B, \tag{2}$$

where $\circ$ denotes the Hadamard product, and σ denotes the sigmoid function. Note that A is typically very sparse. Therefore, F is also sparse: if $A_{i,j} = 0$, $F_{i,j}$ needs not be materialized. Note that zero-value attributes present a different problem of handling potentially missing values, which we leave to future work.

3.3 Overall framework and loss

The overall framework is summarized in Figure 2. We design a loss function that not only incorporates node attributes and network

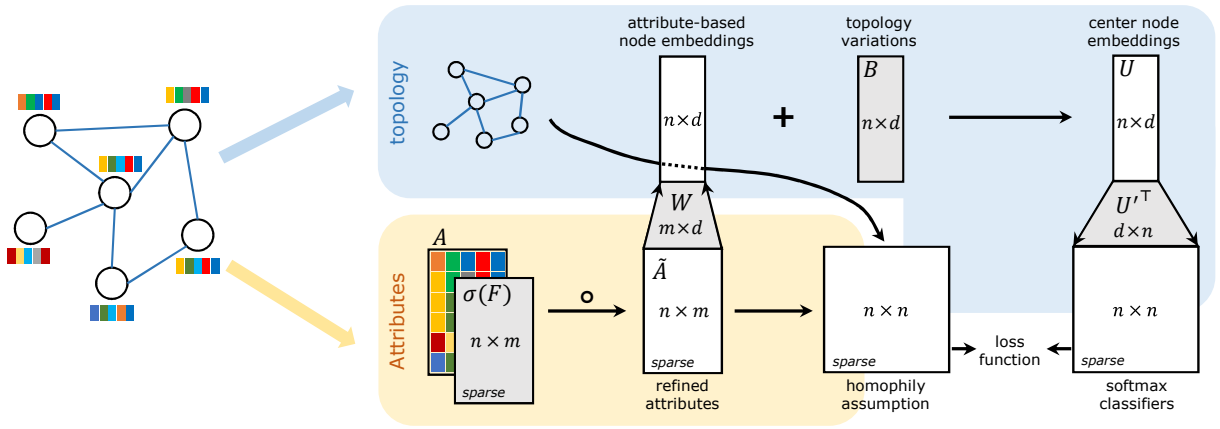

Figure 2: Overall framework. Node embedding matrix U is derived jointly from both topology and (filtered) attributes.

topology, but also performs fine-grained attribute refinement guided by the homophily effect.

Topology and attributes. Like previous work, we also assume that a node v_i is associated with a set of *context* nodes $\mathcal{N}_i$. These context nodes are closely related to v_i based on network topology, such as the neighbors of v_i, or nodes sampled by random walks (*i.e.*, the skip-gram model [22]). We call v_i as the *center* of its contexts. Subsequently, we aim to predict the contexts of a given center v_i:

$$P(\mathcal{N}_i|v_i) = \prod_{v_j \in \mathcal{N}_i} P(v_j|v_i) = \prod_{v_j \in \mathcal{N}_i} \frac{\exp\left(\mathbf{u}'_j \cdot \mathbf{u}_i\right)}{\sum_{k=1}^{n} \exp\left(\mathbf{u}'_k \cdot \mathbf{u}_i\right)}, \quad (3)$$

where $\mathbf{u}'_j$ is the embedding of v_j served as the context, and $\mathbf{u}_i$ is the representation of v_i served as the center. In particular, $\mathbf{u}'_j$ is the j^{th} row of the context embedding matrix $U' \in \mathbb{R}^{n \times d}$, a matrix to be fitted as illustrated in Figure 2. In practice, negative sampling could be adopted for the softmax function to accelerate training. Finally, we minimize the negative log-likelihood of all nodes on the network:

$$L_\alpha = -\sum_{v_i \in V} \sum_{v_j \in \mathcal{N}_i} \log P(v_j|v_i). \quad (4)$$

Since the embedding $\mathbf{u}_i$ depends on the attributes of node v_i, the above loss incorporates both topology and attributes.

Reinforcement of homophily. As motivated earlier, to guide the learning of the filter F without supervision, we exploit the homophily assumption. Specifically, inconsistencies should be identified and refined in a way to reinforce homophily. That is, after attribute refinement using the filter, the homophily effect should ideally become stronger—given a pair of nodes, their similarities in terms of topology and in terms of attributes should correlate better with each other. Let $\text{sim}_\text{t}(v_i, v_j) \in [0, 1]$ measure the topological similarity between v_i and v_j, and $\text{sim}_\text{a}(v_i, v_j) \in [0, 1]$ measure the attribute similarity between v_i and v_j. Therefore, we can optimize the reinforcement of homophily by minimizing the following loss:

$$L_\beta = -\sum_{v_i \in V} \sum_{v_j \in \mathcal{N}_i} \text{sim}_\text{t}(v_i, v_j) \, \text{sim}_\text{a}(v_i, v_j). \quad (5)$$

Note that the $n \times n$ matrix for homophily reinforcement in Figure 2 is sparse, as only the cells corresponding to the center-context pairs are non-zero.

Since the network topology remains fixed in our model, $\text{sim}_\text{t}(v_i, v_j)$ can be treated as a constant and precomputed. One choice for sim_t is the Adamic/Adar index [1]: $\text{sim}_\text{t}(v_i, v_j) = \sum_{v_k \in N_i \cap N_j} \frac{1}{\log |N_k|}$, where N_i is the set of neighbors of v_i. The intuition is that, a common neighboring node of v_i and v_j is less significant when that common node itself has a large neighborhood.

On the other hand, $\text{sim}_\text{a}(v_i, v_j)$ is a function of the refined attributes of v_i and v_j, which is ultimately a function of the learnable filter F. We adopt the common cosine similarity: $\text{sim}_\text{a}(v_i, v_j) = \tilde{\mathbf{a}}_i \cdot \tilde{\mathbf{a}}_j / \|\tilde{\mathbf{a}}_i\| \|\tilde{\mathbf{a}}_j\|$, where $\tilde{\mathbf{a}}_i \in \mathbb{R}^m$ is the refined attribute vector of v_i, or the i^{th} row of the refined attribute matrix. Correspondingly, $\mathbf{a}_i \in \mathbb{R}^m$ is the raw attribute vector.

Overall loss. Accounting for both loss components in Equation (4) and (5), we minimize the following with respect to model parameters $\Theta = (F, W, B, U')$:

$$L = L_\alpha + \lambda L_\beta, \quad (6)$$

where $\lambda \geq 0$ is a hyperparameter to control the trade-off between the two loss components. While $U = (A \circ \sigma(F))W + B$ and U' are the embeddings for nodes serving as the center and context respectively, we use $U + U'$ as the final embedding following SNE [18], given its generally better empirical performance.

3.4 Optimization

We use the Adam optimizer to solve the loss function L in Equation (6). The partial derivatives of L_α and L_β with respect to each parameter can be computed as follows. Note that L_β is only a function of F, and is independent of other parameters in Θ.

$$\frac{\partial L_\alpha}{\partial W} = -\sum_{v_i \in V} \sum_{v_j \in \mathcal{N}_i} \tilde{\mathbf{a}}_i^\top \left(\mathbf{u}'_j - \sum_{k \in V} P(v_k|v_i) \mathbf{u}'_k \right), \quad (7)$$

$$\frac{\partial L_\alpha}{\partial \mathbf{b}_i} = -\sum_{v_j \in \mathcal{N}_i} \left(\mathbf{u}'_j - \sum_{v_k \in V} P(v_k|v_i) \mathbf{u}'_k \right), \quad (8)$$

$$\frac{\partial L_\alpha}{\partial \mathbf{f}_i} = -\left(\sum_{v_j \in \mathcal{N}_i} \mathbf{u}'_j - |\mathcal{N}_i| \sum_{v_k \in V} P(v_k|v_i)\mathbf{u}'_k\right) W^\top \circ \mathbf{a}_i \frac{\partial \sigma(\mathbf{f}_i)}{\partial \mathbf{f}_i}, \quad (9)$$

$$\frac{\partial L_\alpha}{\partial \mathbf{u}'_i} = -\sum_{v_j \in \mathcal{N}_i^{-1}} \mathbf{u}_j + \sum_{v_k \in V} |\mathcal{N}_k| P(v_i|v_k)\mathbf{u}_k, \quad (10)$$

$$\frac{\partial L_\beta}{\partial \mathbf{f}_i} = -\sum_{v_j \in \mathcal{N}_i \cup \mathcal{N}_i^{-1}} \mathrm{sim_t}(v_i, v_j) \frac{\partial \mathrm{sim_a}(v_i, v_j)}{\partial \sigma(\mathbf{f}_i)} \frac{\partial \sigma(\mathbf{f}_i)}{\partial \mathbf{f}_i}, \quad (11)$$

where $\mathcal{N}_i^{-1}$ denotes the inverse of context—the set of centers of v_i when v_i is the context. When $\mathrm{sim_a}$ is the cosine similarity, it is differentiable with respect to $\sigma(F)$, and we omit it here.

To accelerate the training process, we use negative sampling to approximate the softmax function. Instead of considering all nodes in V as the negative examples, we only sample a fixed number of nodes with the log-uniform sampler $P_{\mathrm{neg}}(v_i) = \frac{\log(i+1)-\log(i)}{\log(n+1)}$, where the nodes $v_1, v_2, ..., v_n$ are sorted in descending order of their degrees. In this way, a node with a larger degree has a higher probability to be sampled. Note that when negative sampling is used, the $n \times n$ matrix for softmax classifiers in Figure 2 becomes sparse.

Lastly, we analyze the time complexity of our optimization during one training epoch. Assume a total of ℓ context-center pairs, *i.e.*, $\sum_{i \in V} \sum_{j \in \mathcal{N}_i} 1 = \ell$. By Equation (2), the node representation matrix U can be computed in $O(nmd)$ time. Given a computed U, the softmax function for a single pair can be computed in $O(d)$ time with negative sampling, where a constant number of context nodes are sampled for each node. Subsequently, the loss function and its gradients can be computed with the time complexity of $O(nmd + \ell md)$. Typically $\ell \geq n$, and thus the overall complexity becomes $O(\ell md)$, *i.e.*, our algorithm scales linearly in the number of pairs, attributes and embedding dimensions.

4 EXPERIMENTS

In this section, we conduct extensive experiments on three public datasets. In particular, we evaluate the performance of NEAR and state-of-the-art baselines using two common tasks on network data: node classification and link prediction. We further investigate the underlying mechanism of NEAR by analyzing the impact of inconsistencies, the learnable filter, and the selection of parameter.

4.1 Experimental setup

Datasets. We consider three public network datasets[1] with node attributes, as summarized in Table 2. Each node represents a document, and the links represent the citations between the documents. A node is associated with some attributes, where each attribute represents a word feature. In particular, on the Cora and Citeseer datasets, the attributes are binary-valued to indicate the presence or absence of the words in their titles and abstracts. On the Wiki dataset, the attributes are real-valued as the tf-idf of the words derived from the entire document. Nodes without any attribute is removed in a preprocessing step. On each dataset, a node is further labeled as one of the classes (*e.g.*, the topics of the documents).

Tasks and evaluation. We evaluate the learned embedding on two common tasks, namely, node classification and link prediction. We held out 10% of the edges in each network for validation and another 10% for testing, while the remaining 80% are used for training node embeddings. Nodes not found in the training edges are disregarded.

[1] https://github.com/thunlp/TADW

Table 2: Description of datasets.

Dataset	# Nodes	# Edges	# Attributes	Attr. type	# Classes
Cora	2 708	5 429	1 433	binary	7
Citeseer	3 312	4 732	3 703	binary	6
Wiki	2 405	17 981	4 973	real	19

For node classification, each node is associated with one of the several classes as shown in Table 2. Therefore, this is a multi-class classification problem. Note that the classes are often imbalanced, and thus we evaluated the results with micro- and macro-F scores in addition to accuracy. Each node's final embedding is used as the feature vector.

For link prediction, while using the held-out 10% test edges as positive instances, we further sampled an equal number of unlinked node pairs as negative instances. In particular, half of the negative instances were randomly sampled from nodes that are two-hop away from each other, and the other half were randomly chosen from all unlinked pairs. The task is thus cast as a binary classification problem, and can be evaluated with the standard metrics of F-score, AUC-ROC and accuracy. As each instance contains two nodes, we used the Hadamard product of the two nodes' final embeddings as the feature vector.

For both tasks, we used an SVM classifier. The instances for each task were further split into 80% for training and 20% for testing of the SVM, and the classifier is tuned on the training set with five-fold cross validation. We repeated such splitting for 10 times and report the average results.

Algorithms under evaluation. We compare NEAR with the following baseline algorithms, as well as two variants of NEAR.

- DeepWalk [22]: A classic network embedding algorithm that exploits topology only.
- SNE [18]: A state-of-the-art method for attributed network embedding, which fuses node and attribute encodings and feeds them into a neural network.
- AANE [11]: A state-of-the-art method for attributed network embedding, which aims to recover attribute and toplogy-based similarity matrices jointly through matrix factorization.
- GraphSAGE [10]: A state-of-the-art graph neural network method, which derives node representations from attributes in a toplogy-aware manner. We adopt their unsupervised variant to match our setting.
- NEAR\F: The same as NEAR except that no filter is used, *i.e.*, in Equation (2) the filter matrix F is set to all one's without any learnable parameter. This variant is also similar in spirit to the UPP-SNE model [30].
- NEAR\H: The same as NEAR except that there is no reinforcement on the homophily effect, *i.e.*, in Equation (6) we set $\lambda = 0$.

In all methods, the number of embedding dimensions is set to 64. Other dimensions such as 128 give similar results. The hyperparamters of the baselines are tuned empirically based on guidance from their respective papers. In NEAR, the hyperparameter λ that

Table 3: Performance on node classification (multi-class classification).

	Cora			Citeseer			Wiki		
	Micro-F	Macro-F	Accuracy	Micro-F	Macro-F	Accuracy	Micro-F	Macro-F	Accuracy
DeepWalk	0.5673	0.5671	0.5928	0.4178	0.3679	0.4262	0.5846	0.4321	0.5867
SNE	0.6456	0.6414	0.6555	0.4411	0.4033	0.4623	0.5914	0.4585	0.5939
AANE	0.0000	0.0658	0.2990	0.2252	0.1056	0.2103	0.0076	0.0188	0.1682
GraphSAGE	0.3364	0.3499	0.4239	0.3392	0.2931	0.3454	0.3562	0.2491	0.3628
NEAR\F	0.7350	0.7326	0.7495	0.5567	0.5242	0.5865	0.6869	0.5625	0.6791
NEAR\H	0.7044	0.7054	0.7159	0.5170	0.4743	0.5363	0.6263	0.4868	0.6198
NEAR	**0.7604**	**0.7577**	**0.7707**	**0.6440**	**0.6241**	**0.6667**	**0.6885**	**0.5792**	**0.6912**

Table 4: Performance on link prediction (binary classification).

	Cora			Citeseer			Wiki		
	F	AUC-ROC	Accuracy	F	AUC-ROC	Accuracy	F	AUC-ROC	Accuracy
DeepWalk	0.6202	0.6682	0.6262	0.6211	0.6517	0.6120	0.8285	**0.9027**	0.8301
SNE	0.6868	0.7430	0.6854	0.6583	0.7172	0.6574	0.8007	0.8754	0.8043
AANE	0.6321	0.6675	0.6202	0.6222	0.6612	0.6457	0.7891	0.8387	0.7758
GraphSAGE	0.6611	0.7158	0.6624	0.6253	0.6555	0.6159	0.7710	0.8494	0.7674
NEAR\F	0.6957	0.7449	0.6903	0.6721	**0.7235**	0.6643	0.8240	0.8943	0.8190
NEAR\H	0.6829	0.7505	0.6804	0.6462	0.6842	0.6415	0.8243	0.8972	0.8179
NEAR	**0.7092**	**0.7626**	**0.7014**	**0.6736**	0.7161	**0.6756**	**0.8394**	0.9013	**0.8430**

controls the homophily effect is tuned using the softmax loss on the held-out validation edges.

4.2 Performance comparison

We compare the performance of the baselines and NEAR on both tasks of node classification and link prediction.

Node classification. We first report the results on the three dataset in Table 3. In summary, our NEAR obtains the best results consistently across all datasets and metrics. Among the baselines, DeepWalk only considers network topology. Thus, by integrating valuable node attributes, SNE is able to perform better than DeepWalk as expected. Note that, while AANE and GraphSAGE also considers attributes, they achieve unsatisfactory results as they do not handle class imbalance well. In particular, on the Cora and Wiki datasets, AANE predicts the largest class on almost all nodes, giving very low micro- and macro-F scores. In general, SNE represents the best baseline, but it is still inadequate as it assumes no inconsistency in the attributes. Thus, our proposed NEAR outperforms SNE often by a large margin, *e.g.*, by 10% or more in micro-F scores.

Not surprisingly, NEAR is also superior to its two variants. It is worth noting that NEAR\F is often better than NEAR\H, even though the former does not employ any filter whereas the latter does. The results imply that the noise filter is not learnable on its own in an unsupervised setting; without any guidance from the homophily assumption, the filter may eventually lead to worse outcomes given an increased number of parameters. To gain more insights, we will further examine the quality of the learned filters in Section 4.3.

Link prediction. We report the results in Table 4. Similarly, our method NEAR generally outperforms all the baselines. On the Cora and Citeseer datasets, the attributed embedding baselines including SNE, AANE and GraphSAGE often perform better than DeepWalk. On the Wiki dataset, as the attributes are extracted from the full document, more inconsistencies in the attributes can be expected (in contrast, on the Cora and Citeseer datasets, the attributes are only extracted from the titles and abstracts). This potentially explains why DeepWalk performs better on the Wiki dataset. However, since NEAR is able to filter out such inconsistencies, it still outperforms the baselines by a noticeable margin, *e.g.*, up to 3.3% in F-scores.

4.3 Analysis and discussion

To understand the underlying mechanism of NEAR, we further investigate the impact of inconsistencies, analyze the learned filters, and study parameter selection.

Impact of inconsistencies. While the datasets naturally contain inconsistencies in node attributes, to better analyze the refinement mechanism of NEAR, we further added artificial inconsistencies to the attributes. Specifically, for each attribute of each node whose original value is zero, there is a probability to flip to one if the attribute is binary, or a non-zero real value sampled from the distribution of that attribute. We varied this probability between 0 and 0.05 in our experiments.

We compare NEAR with its two simpler variants NEAR\F and NEAR\H, as introduced earlier. In Figures 3 and 4, we plot the impact of artificial inconsistencies on node classification and link prediction tasks, respectively. Note that a zero probability is equivalent to the original dataset. While the performance of NEAR and its variants all decreases as the probability of inconsistencies increases, NEAR is generally less affected due to its learnable filter and fine-grained refinement mechanism. Similar to the observation in Section 4.2, NEAR\F generally outperforms NEAR\H, which

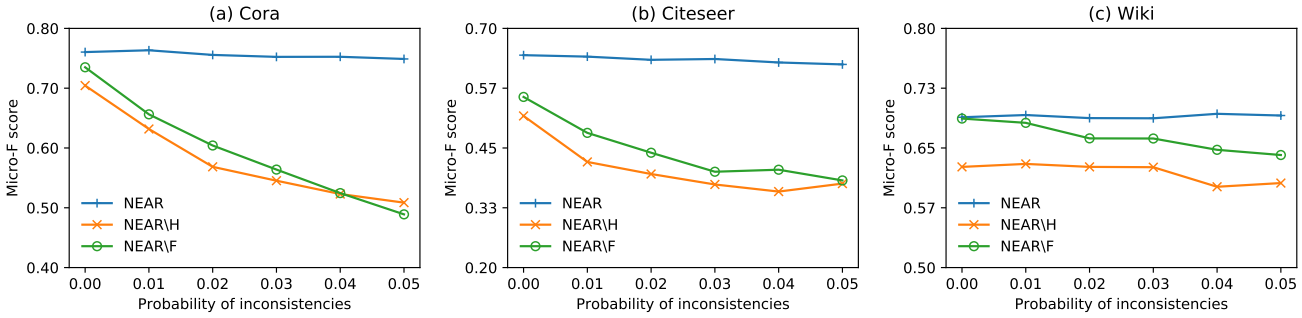

Figure 3: Impact of artificial inconsistencies on node classification.

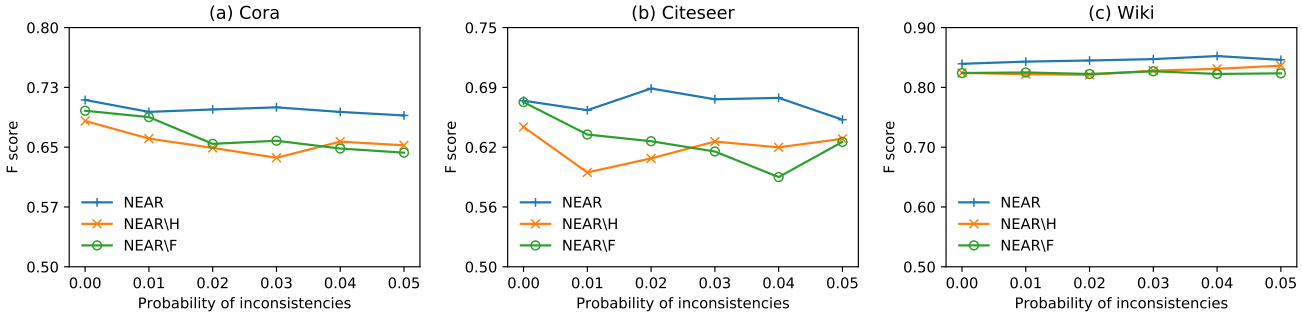

Figure 4: Impact of artificial inconsistencies on link prediction.

means the homophily reinforcement is necessary for learning the filter.

We also observe that NEAR\F and NEAR\H are more prune to inconsistencies in node classification than in link prediction. A potential explanation is that attributes are node-level information that could more directly influence node classification.

Evaluation of learned filter. As our model hinges on the learnable filter, we evaluate the learned filter to demonstrate that it aligns with the goal of attribute refinement. As there is no ground truth on attribute inconsistencies, we analyze the filters learned when we added artificial inconsistencies to the datasets with probability 0.05. Specifically, we examine the individual elements in the learned filter F, dividing them into two subsets: Φ containing the elements in F corresponding to the non-zero attribute values in the original dataset; and Φ' containing the elements in F corresponding to the artificial inconsistencies. Intuitively, a meaning filter is able to separate the original attribute values and the artificial inconsistencies, resulting in differently distributed subsets Φ and Φ'.

To evaluate how well Φ and Φ' can be separated (and hence how good the filter is), we compute several metrics defined below.

- Mean-ratio: the ratio between the mean values of Φ and Φ'. A random filter produces a ratio of 1, whereas a meaningful filter produces a significantly larger ratio;
- AUC-ROC: using the filter value as a predictor to classify whether a given element belongs to Φ or Φ', we assess the classification power by AUC-ROC. A random filter produces a score of 0.5, whereas a meaningful filter produces a score close to 1;
- p-value: we perform the nonparametric Kolmogorov-Smirnov test to compare if the two subsets Φ and Φ' are drawn from the same continuous distribution. For this purpose, we randomly sample 1 000 values from each subset for comparison. A sufficiently small p-value rejects the null hypothesis that the two samples are drawn from the same distribution.

We present the above metrics in Table 5 to compare the filters learned by NEAR and NEAR\H. We hypothesize that, NEAR\H will learn a near-random filter that does not separate Φ and Φ', as it employs no explicit guidance to learn the filter. On the contrary, NEAR guides the learning with the homophily assumption, and thus will learn a meaningful filter that better separates the two subsets. The results in Table 5 indeed establish consistent evidence that NEAR learns much more meaningful filters than NEAR\H does. The implication is that the homophily assumption is crucial to learning a meaningful filter, towards identifying and mitigating attribute inconsistencies effectively.

Selection of hyperparameter. Last but not the least, we study how the hyperparameter λ influences our model NEAR. Specifically, we vary λ between 0 and 10^4, and report the resulting performance in Figure 5. Note that NEAR degenerates to NEAR\H when $\lambda = 0$. The datasets marked with # indicate that they contain artificial inconsistencies with probability 0.05.

First, consider the node classification task in Figure 5(a). As we increase λ, model performance generally improves until eventually

Table 5: Comparison of filters learned by NEAR and NEAR\H.

	Cora		Citeseer		Wiki	
	NEAR\H	NEAR	NEAR\H	NEAR	NEAR\H	NEAR
mean-ratio	1.01	6.74	1.00	6.07	1.01	10.3
AUC-ROC	.504	.649	.501	.651	.503	.912
p-value	.459	$< .001$	.493	$< .001$	.603	$< .001$

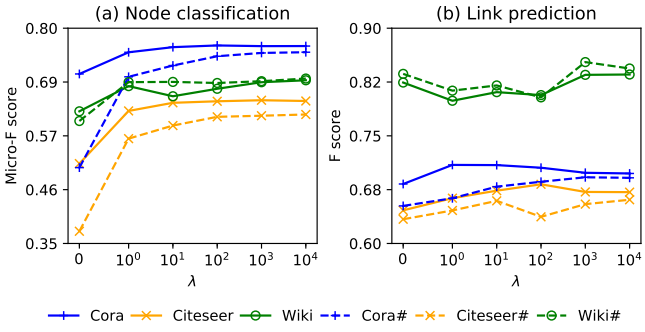

Figure 5: Effects of λ on both original datasets and their counterparts with artificial inconsistencies (marked with #).

reaching a plateau. Moreover, the plateau is often reached earlier on the original datasets than on their counterparts with artificial inconsistencies. Such an observation is intuitive: when inconsistencies are more severe, stronger homophily assumption (*i.e.*, a larger λ) is required to properly guide the learning of the filter.

Second, on the link prediction task in Figure 5(b), a similar general pattern still exists, although the trend is weaker than that on node classification. Again, a potential explanation is that attribute inconsistencies affect node classification more, given that attributes are node-level information.

5 CONCLUSION

In this paper, we studied the problem of network embedding with attribute inconsistencies. To exploit both network topology and node attributes, the homophily assumption is leveraged: nodes tend to link with other nodes similar to themselves. However, in real-world networks, node attributes often present inconsistencies in various ways, undermining the homophily assumption. Towards robust network embedding with attributes, we proposed a novel model NEAR hinging on a learnable filter, to automatically refine attributes in an unsupervised and fine-grained manner under the guidance of the homophily assumption. In particular, attributes should be refined in a way that reinforces the correlation between attributes and topology. Lastly, we conducted extensive experiments on three datasets, and obtained promising results on node classification and link prediction tasks. We further investigated the learned filter, which demonstrates its ability to effectively refine attributes.

REFERENCES

[1] Adamic, L. A., and Adar, E. Friends and neighbors on the web. *Social Networks 25*, 3 (2003), 211–230.
[2] Ata, S. K., Fang, Y., Wu, M., Li, X.-L., and Xiao, X. Disease gene classification with metagraph representations. *Methods 131* (2017), 83–92.
[3] Bandyopadhyay, S., Lokesh, N., and Murty, M. N. Outlier aware network embedding for attributed networks. In *AAAI* (2019), vol. 33, pp. 12–19.
[4] Consortium, U. UniProt: a hub for protein information. *Nucleic acids research 43*, D1 (2015), D204–D212.
[5] Fang, Y., and Chang, M.-W. Entity linking on microblogs with spatial and temporal signals. *TACL 2* (2014), 259–272.
[6] Fang, Y., Hsu, B. P., and Chang, K. C. Confidence-aware graph regularization with heterogeneous pairwise features. In *SIGIR* (2012), pp. 951–960.
[7] Feng, R., Yang, Y., Hu, W., Wu, F., and Zhang, Y. Representation learning for scale-free networks. In *AAAI* (2018).
[8] Gao, H., and Huang, H. Deep attributed network embedding. In *IJCAI* (2018), pp. 3364–3370.
[9] Grover, A., and Leskovec, J. Node2vec: Scalable feature learning for networks. In *KDD* (2016), pp. 855–864.
[10] Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *NeurIPS* (2017), pp. 1024–1034.
[11] Huang, X., Li, J., and Hu, X. Accelerated attributed network embedding. In *SDM* (2017), pp. 633–641.
[12] Huang, X., Li, J., and Hu, X. Label informed attributed network embedding. In *WSDM* (2017), pp. 731–739.
[13] Huang, Z., Chung, W., and Chen, H. A graph model for e-commerce recommender systems. *JASIST 55*, 3 (Feb. 2004), 259–274.
[14] Kipf, T. N., and Welling, M. Semi-supervised classification with graph convolutional networks. In *ICLR* (2017).
[15] Li, C., Wang, S., Yang, D., Li, Z., Yang, Y., Zhang, X., and Zhou, J. PPNE: Property preserving network embedding. In *DASFAA* (2017), pp. 163–179.
[16] Li, R., Wang, S., and Chang, K. C.-C. Multiple location profiling for users and relationships from social network and content. *PVLDB 5*, 11 (2012), 1603–1614.
[17] Liang, J., Jacobs, P., Sun, J., and Parthasarathy, S. Semi-supervised embedding in attributed networks with outliers. In *SDM* (2018).
[18] Liao, L., He, X., Zhang, H., and Chua, T.-S. Attributed social network embedding. *TKDE 30*, 12 (2018), 2257–2270.
[19] Marsden, P. V. Homogeneity in confiding relations. *Social Networks 10*, 1 (1988), 57–76.
[20] McPherson, M., Smith-Lovin, L., and Cook, J. M. Birds of a feather: Homophily in social networks. *Annual Review of Sociology 27*, 1 (2001), 415–444.
[21] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. Distributed representations of words and phrases and their compositionality. In *NeurIPS* (2013), pp. 3111–3119.
[22] Perozzi, B., Al-Rfou, R., and Skiena, S. DeepWalk: Online learning of social representations. In *KDD* (2014), pp. 701–710.
[23] Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., and Mei, Q. LINE: Large-scale information network embedding. In *WWW* (2015), pp. 1067–1077.
[24] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. In *ICLR* (2017).
[25] Wang, D., Cui, P., and Zhu, W. Structural deep network embedding. In *KDD* (2016), pp. 1225–1234.
[26] Wang, H., Chen, E., Liu, Q., Xu, T., Du, D., Su, W., and Zhang, X. A united approach to learning sparse attributed network embedding. In *ICDM* (2018), pp. 557–566.
[27] Wang, X., Cui, P., Wang, J., Pei, J., Zhu, W., and Yang, S. Community preserving network embedding. In *AAAI* (2017), pp. 203–209.
[28] Wei, X., Xu, L., Cao, B., and Yu, P. S. Cross view link prediction by learning noise-resilient representation consensus. In *WWW* (2017), pp. 1611–1619.
[29] Yang, C., Liu, Z., Zhao, D., Sun, M., and Chang, E. Y. Network representation learning with rich text information. In *IJCAI* (2015), pp. 2111–2117.
[30] Zhang, D., Yin, J., Zhu, X., and Zhang, C. User profile preserving social network embedding. In *IJCAI* (2017), pp. 3378–3384.
[31] Zhang, D., Yin, J., Zhu, X., and Zhang, C. SINE: Scalable incomplete network embedding. In *ICDM* (2018), pp. 737–746.
[32] Zhang, Z., Yang, H., Bu, J., Zhou, S., Yu, P., Zhang, J., Ester, M., and Wang, C. ANRL: Attributed network representation learning via deep neural networks. In *IJCAI* (2018), pp. 3155–3161.
[33] Zhou, S., Yang, H., Wang, X., Bu, J., Ester, M., Yu, P., Zhang, J., and Wang, C. PRRE: personalized relation ranking embedding for attributed networks. In *CIKM* (2018), pp. 823–832.