Retrofitting structural graph embeddings with node attribute information

Piotr Bielak[1][0000−0002−1487−2569], Daria Puchalska[1], and Tomasz Kajdanowicz[1][0000−0002−8417−1012]

Department of Artificial Intelligence,
Wrocław University of Science and Technology,
Wrocław, Poland
piotr.bielak@pwr.edu.pl

Abstract. Representation learning for graphs has attracted increasing attention in recent years. In this paper, we define and study a new problem of learning attributed graph embeddings. Our setting considers how to update existing node representations from structural graph embedding methods when some additional node attributes are given. To this end, we propose Graph Embedding RetroFitting (GERF), a method that delivers a compound node embedding that follows both the graph structure and attribute space similarity. Unlike other attributed graph embedding methods, GERF is a novel representation learning method that does not require recalculation of the embedding from scratch but rather uses existing ones and retrofits the embedding according to neighborhoods defined by the graph structure and the node attributes space. Moreover, our approach keeps the same embedding space all the time and allows comparing the positions of embedding vectors and quantifying the impact of attributes on the representation update. Our GERF method updates embedding vectors by optimizing the invariance loss, graph neighbor loss, and attribute the neighbor loss to obtain high-quality embeddings. Experiments on WikiCS, Amazon-CS, Amazon-Photo, and Coauthor-CS datasets demonstrate that our proposed algorithm receives similar results compared to other state-of-the-art attributed graph embedding models despite working in retrofitting manner.

Keywords: graph embedding · attributed graphs · graph embedding retrofitting.

1 Introduction

Machine learning methods have been studied in a variety of applications and data types, including images and video (computer vision), text (natural language processing), audio or time-series data, among many others. Since most downstream ML models expect a vector from a continuous space as input, representation learning methods have been developed to create those representation vectors (embeddings) automatically. While there are many embedding methods traditional data types, such as word2vec [11] and FastText [4] for text, or ResNet

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_13

[7] and EfficientNet [14] for images, this task is much more difficult for graph-structured data. A simple concatenation of unimodal representations (graph structure and node attributes) is often not sufficient, as it does not consider the mutual relationships between modalities. Therefore, the main challenge for such methods is discovering the interrelationship between multiple modalities to create a coherent representation that will integrate the multimodal information.

Problem statement Consider a situation in which data changing over time is analyzed on an ongoing basis. In the first case, the structure of the network remains unchanged, but the attributes of the nodes are constantly changing – an example may be a network of connected weather sensors. Conversely, the values of the node attributes can be constant, but the structure of the graph changes, e.g., in a telephone network, where the edge denotes the currently ongoing call. In both situations, graph embedding models that consider both the network structure and node attributes can be used, however, if one of these modalities does not change over time, this may not be the best solution. Especially, in the first of the above-mentioned situations, it may be more advantageous to generate the structural graph embeddings once, and then use a method that would modify them depending on the current values of the attributes, somehow incorporating information from the attribute space into the structural embedding space. The simplest solution would be to simply concatenate both vectors, but the resulting representation would be neither consistent nor low-dimensional.

Goal The aim of this work is to develop an algorithm that will enhance (retrofit) existing structural node embeddings by incorporating information from the attribute space. That is, based on the node attributes, it will appropriately modify the embedding vectors derived from the structural graph representation learning methods. The new embedding vectors returned by such method should provide better performance in downstream tasks than by using naive approaches (like concatenation of structural embeddings with node attribute vectors).

Contributions We summarize our contributions as follows: (1) we introduce a new problem in the area of graph representation learning, in which a structural network embedding is updated (retrofitted) according to node attributes, (2) we propose a novel method (GERF) for unsupervised representation learning on graphs which combines information from the space of structural embeddings and node attributes while maintaining low dimensionality of the representation vectors, (3) we perform experiments demonstrating competitive quality of the proposed GERF method compared to other approaches, (4) we make our code and experimental pipeline publicly available to ensure reproducibility: `https://github.com/pbielak/gerf/`.

2 Related Work

The problem of graph representation learning (GRL) has received a lot of attention in recent years in the machine learning community. The main goal is to learn

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_13

low-dimensional continuous vector representations (embeddings), which can be later used for specific downstream prediction tasks such as node classification or link prediction. We can distinguish two groups of embeddings in GRL methods: (i) structural embeddings that take into account information extracted from the network structure only, such as the neighborhood of nodes proximities, and (ii) attributed embeddings which, apart from the relationships in the network structure, also reflect the similarity in the node feature space.

2.1 Structural representation learning methods

Early GRL methods built low-dimensional node embeddings that reflect the structure of the network. Among them, the most frequently referenced and used are: DeepWalk [12], Node2vec [6], LINE [15] and SDNE [16]. **DeepWalk** [12] samples node sequences using random walks and passes them into the Skip-gram model [11] (a word embedding method). **Node2vec** [6] extends DeepWalk by developing a biased random walk procedure to explore diverse neighborhoods by interpolating between a breadth-first (BFS) and depth-first (DFS) graph search algorithms. **LINE** [15] is a scalable method that learns node representations by preserving the first-order (similar embeddings of neighbor nodes) and second-order graph proximities (similar embeddings of nodes sharing the same neighborhood). **SDNE** [16] also focuses on preserving the first-order and second-order proximities. However, it uses an autoencoder approach to map the highly non-linear underlying network structure to latent space.

2.2 Attributed graph embedding methods

The structure of the network is given by connections between objects. However, there are many other possible sources of information. Additional node attributes can be given in the form of a vector representation of their content, which in the case of classic methods such as *bag-of-words* model or *tf-idf* is an additional challenge because these vectors are usually sparse. Methods designed to learn representations in attributed networks include TADW [17], FSCNMF [3], DANE [5] and ANRL [18]. **TADW** [17] (**T**ext-**A**ssociated **D**eep**W**alk) shows that DeepWalk is equivalent to matrix factorization and proposes its text-associated version. **FSCNMF** [3] is based on non-negative matrix factorization and produces node embeddings that are consistent with the graph structure and nodes' attributes. The structure-based embedding matrix serves as a regularizer when optimizing the attribute-based embedding matrix and vice-versa.

3 Notations and problem definition

Graph A graph G is a pair $G = (V, E)$, where $V = \{v_1, \ldots, v_{|V|}\}$ is a set of nodes and $E \subseteq V \times V$ is the set of edges that connect node pairs, i.e., each edge e_{ij} is a pair (v_i, v_j). The graph connectivity can be represented as an adjacency matrix $\mathbf{A} \in \{0, 1\}^{|V| \times |V|}$ with element A_{ij} indicating the existence of an edge (v_i, v_j).

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_13

Attributed graph Apart from the structure of connections between objects, this kind of graph has additional information about each of the nodes, i.e., each node has an assigned feature vector (also called the attribute vector). An attributed graph is a 3-tuple $G = (V, E, \mathbf{X})$, where V and E follow the previous definition. $\mathbf{X} \in \mathbb{R}^{|V| \times d_X}$ is a matrix that encodes all node attributes information, and $\mathbf{x}_i$ describes the attributes associated with node v_i.

Attribute proximity We can analyze the similarity between nodes not only based on the network structure but also in the attribute space. Given a network $G = \{V, E, \mathbf{X}\}$, the attribute proximity of two nodes v_i and v_j is determined by the similarity of $\mathbf{x}_i$ and $\mathbf{x}_j$. Note that these are two separate spaces to analyze. The similarity of two nodes in the graph structure does not imply their similarity in the attribute space and vice versa. Thus, the representation learning methods for attributed graphs should take into account dependencies in both spaces and coherently combine them.

Node representation learning Given a network $G = (V, E)$ (or $G = (V, E, \mathbf{X})$ in the case of an attributed network), the goal is to represent every graph node $v_i \in V$ as a low-dimensional vector $\mathbf{z}_i$ (called node embedding) by learning a mapping function $f : v_i \rightarrow \mathbf{z}_i \in \mathbb{R}^{d_Z}$, where $d_Z << |V|$, such that important network properties are preserved in the embedding space (e.g. structural and semantic graph information). Overall, the node embeddings are stored as a node embedding matrix $\mathbf{Z} \in \mathbb{R}^{V \times d_Z}$. If two nodes are similar in the graph structure (they are connected or share neighbors), or have similar attribute values, their learned embeddings should also be similar.

Attribute-based neighborhood We can easily define the neighborhood of a node in the network as the set of other nodes that are connected to it by an edge. However, there are no clear relationships between objects within the attribute space itself. To combine information from the attribute space (which objects are closer to each other in this space and which are further) with structural relationships, it is necessary to first define the so-called *attribute-based neighborhood*. Based on the attribute matrix $\mathbf{X}$, for each node in the network G, its k nearest neighbors in this space were found based on the Euclidean distance metric, with k being equal to the number of neighbors of this node in the network G. The neighborhood of the node v_i in the attribute space, defined in this way, will be denoted by $\mathcal{N}_{\mathbf{X}}(v_i)$. Therefore, $\forall_i \ |\mathcal{N}(v_i)| = |\mathcal{N}_{\mathbf{X}}(v_i)|$.

4 Graph Embedding RetroFitting (GERF)

In this section, we describe our proposed Graph Embedding RetroFitting model that allows to update existing structural node embeddings $\mathbf{Z}$ with the node attribute information $\mathbf{X}$, resulting in the retrofitted node embeddings $\mathbf{Z}^*$. Figure 1 shows the overall processing pipeline of our method.

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_13

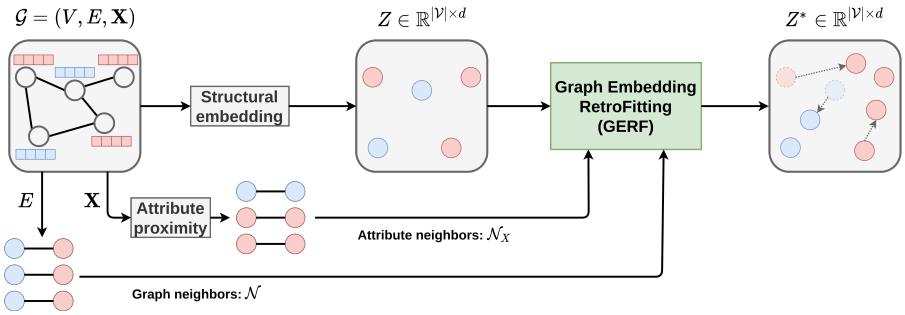

Fig. 1. Our proposed graph embedding retrofitting (GERF) method uses information about the structure of the graph (graph neighbors) and the node attribute space (attribute neighbors) to retrofit a structural embedding Z into one that incorporates node attribute information Z^*.

4.1 Objective function

Our model is based on the optimization of the $\mathbf{Z}^* \in \mathbb{R}^{V \times d}$ matrix. The objective function of our proposed GERF model takes the following form:

$$\begin{aligned}\mathcal{L}(\mathbf{Z}^*) = (1 - \lambda_G - \lambda_X) \sum_{i=1}^{n} \|\mathbf{z}_i^* - \mathbf{z}_i\|^2 \\ + \lambda_G \sum_{i=1}^{n} \sum_{j:\, v_j \in \mathcal{N}(v_i)} \frac{\|\mathbf{z}_i^* - \mathbf{z}_j^*\|^2}{|\mathcal{N}(v_i)|} + \lambda_X \sum_{i=1}^{n} \sum_{j:\, v_j \in \mathcal{N}_{\mathbf{X}}(v_i)} \frac{\|\mathbf{z}_i^* - \mathbf{z}_j^*\|^2}{|\mathcal{N}_{\mathbf{X}}(v_i)|},\end{aligned} \tag{1}$$

where $\mathbf{Z} = (\mathbf{z}_1, \ldots, \mathbf{z}_n)$ are the pre-trained structural embeddings for each node, $\mathbf{Z}^* = (\mathbf{z}_1^*, \ldots, \mathbf{z}_n^*)$ are the new embeddings combining multimodal information (from both spaces), and $\lambda_G > 0$ and $\lambda_X > 0$ are non-negative method hyperparameters that control the importance of the structural and attribute similarity, respectively.

With the purpose of the work in mind, one can easily explain the intuition behind each component in the objective function and why it should be included there. Since the method is intended to enhance the space of structural embeddings by incorporating information from the attribute space, it is necessary to include an a component in the objective function that will "keep the embeddings in place". That is, make sure that the new embeddings do not deviate significantly from their original values because this would lead to a complete loss of information from this space. Hence, what is needed is a component which is later referred to as **invariance loss**:

$$\mathcal{L}_I(\mathbf{Z}^*) = \sum_{i=1}^{n} \|\mathbf{z}_i^* - \mathbf{z}_i\|^2. \tag{2}$$

Further, in order to combine information from the network structure and node attributes, for each node its neighborhood in both of these spaces is considered, as defined earlier. In order for the representation of each node to be

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_13

similar to the representation of objects close to it in both spaces, it is necessary to define the loss related to the distance between the embeddings in the network, called **graph neighbor loss**:

$$\mathcal{L}_G(\mathbf{Z}^*) = \sum_{i=1}^{n} \sum_{j:\, v_j \in \mathcal{N}(v_i)} \frac{1}{|\mathcal{N}(v_i)|} \|\mathbf{z}_i^* - \mathbf{z}_j^*\|^2, \tag{3}$$

as well as an analogous component that controls the distances between node embeddings based on their attributes, called **attribute neighbor loss**:

$$\mathcal{L}_X(\mathbf{Z}^*) = \sum_{i=1}^{n} \sum_{j:\, v_j \in \mathcal{N}_{\mathbf{X}}(v_i)} \frac{1}{|\mathcal{N}(v_i)|} \|\mathbf{z}_i^* - \mathbf{z}_j^*\|^2. \tag{4}$$

By combining equations 2-4, it is possible to write the formula in Equation 1 in a different, simpler form:

$$\mathcal{L}(\mathbf{Z}^*) = (1 - \lambda_G - \lambda_X)\, \mathcal{L}_I(\mathbf{Z}^*) + \lambda_G\, \mathcal{L}_G(\mathbf{Z}^*) + \lambda_X\, \mathcal{L}_X(\mathbf{Z}^*). \tag{5}$$

4.2 Optimization

The Adam optimizer [8] was used to minimize the objective function from Equation 1. One can easily derive the formula for the first derivative of the function $\mathcal{L}$ with respect to one vector $\mathbf{z}_i^*$ as follows:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \mathbf{z}_i^*} &= 2\,(1 - \lambda_G - \lambda_X)\,(\mathbf{z}_i^* - \mathbf{z}_i) \\
&+ 2\,\lambda_G \sum_{j:\, v_j \in \mathcal{N}(v_i)} \frac{\mathbf{z}_i^* - \mathbf{z}_j^*}{|\mathcal{N}(v_i)|} - 2\,\lambda_G \sum_{j:\, v_i \in \mathcal{N}(v_j)} \frac{\mathbf{z}_j^* - \mathbf{z}_i^*}{|\mathcal{N}(v_j)|} \\
&+ 2\,\lambda_X \sum_{j:\, v_j \in \mathcal{N}_{\mathbf{X}}(v_i)} \frac{\mathbf{z}_i^* - \mathbf{z}_j^*}{|\mathcal{N}_{\mathbf{X}}(v_i)|} - 2\,\lambda_X \sum_{j:\, v_i \in \mathcal{N}_{\mathbf{X}}(v_j)} \frac{\mathbf{z}_j^* - \mathbf{z}_i^*}{|\mathcal{N}_{\mathbf{X}}(v_j)|}.
\end{aligned}$$

The matrix $\mathbf{Z}^*$ is initialized with the values of $\mathbf{Z}$. Currently, the values of λ_G and λ_X hyperparameters are determined based on grid search and simultaneous analysis of the model results in downstream tasks. However, more advanced techniques could be proposed for this purpose, e.g. based on the properties of the network structure and attribute space, which is planned for future work.

4.3 Summary

The proposed method allows the creation of a coherent representation for nodes in the network based on their attribute values and pre-trained structural embeddings. It assumes that there are dependencies between objects in the attribute space. Objects are considered adjacent if the distance between their attribute vectors is sufficiently small compared to others. The main advantages of this method are intuitive assumptions and simplicity of operation, as it allows for easy integration of multimodal information from the network structure and node attributes in the form of a low-dimensional representation vector.

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_13

5 Experimental setup

We perform an analysis of selected graph representation learning methods in the node classification downstream task. We compare attributed graph embedding models (TADW, FSCNMF, DGI), structural embeddings (node2vec, LINE, SDNE), and the ones modified by the proposed GERF method and a few other baselines with each other. Additionally, a simple approach is tested that completely ignores the network structure and uses only node attributes for the prediction. Four real-world benchmark datasets are employed.

5.1 Datasets

We employ four real-wold benchmark datasets from the *PyTorch-Geometric* [2] library. The statistics are provided in Table 1.

- **WikiCS** [10] is a network of Computer Science-related Wikipedia articles with edges denoting references between those articles. Each article belongs to one of 10 subfields (classes) and has features computed as averaged GloVe embeddings of the article content. We use the first provided train/val/test data splits without any modifications (we recompute the embeddings 10 times).
- **Amazon Computers** (Amazon-CS), **Amazon Photos** [9] are two networks extracted from Amazon's co-purchase data. Nodes are products and edges denote that these products were often bought together. Based on the reviews, each product is described using a Bag-of-Words representation (node features). There are 10 and 8 product categories (node classes), respectively. There are no data splits available for those datasets, so we generate a random train/val/test split (10%/10%/80%) for each one.
- **Coauthor-CS** is a network extracted from the Microsoft Academic Graph [13]. Nodes are authors, and edges denote a collaboration of two authors. Each author is described by the keywords used in their articles (Bag-of-Words representation; node features). There are 15 author research fields (node classes). Similar to the Amazon datasets, there is no data split provided, so we generate a random train/val/test split (10%/10%/80%).

Table 1. Datasets statistics.

Name	Nodes	Edges	Features	Classes
WikiCS	11,701	216,123	300	10
Amazon Computers	13,752	245,861	767	10
Amazon Photos	7,650	119,081	745	8
Coauthor-CS	18,333	81,894	6,805	15

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_13

5.2 Embedding methods

To be able to make a qualitative comparison of embedding methods and their ability to compress highly non-linear dependencies in a low-dimensional space, it was concluded that the same size of embedding would be assumed for each of the methods. Thus, each of the algorithms produces 128-dimensional representation vectors. The exact settings for the structural embedding methods are listed below:

- **node2vec** – the same default parameters were adopted for each of the datasets because the quality of the representation vectors obtained in this way was satisfactory (note that the priority of the work is not to achieve the highest possible results of the compared algorithms, but to show that the method proposed in Section 4 is able to improve them, therefore little importance was given to the search for the best set of hyperparameters). We use the following settings: batch size – 128, learning rate – 0.01, walk length – 20, number of walks per node – 10, context size – 10, and number of negative samples – 1. Besides, the algorithm parameters p and q have been set to 1, which means that it is effectively DeepWalk.
- **LINE** – the number of epochs was set to 10 for all datasets, and the mini-batch size was set to 128 (WikiCS, Coauthor-CS), 4096 (Amazon-CS), 256 (Amazon-Photos). Such settings were required as otherwise, the training resulted in collapsed embeddings or NaN values in the embedding vectors. The number of samples in the negative sampling procedure was set to 1 for all datasets.
- **SDNE** – these embeddings showed the worst quality in the downstream task, therefore a hyperparameter grid search was performed, searching for the optimal values of the α, β, ν parameters, as well as the number of epochs and the size of the autoencoder hidden layer. Based on preliminary experiments, the number of epochs was set to 50 and the hidden layer size to 256 for all datasets. The values of method parameters α was chosen to be 10^{-4}, β was left at default 5 and ν_1, ν_2 were set to 10^{-5}, 10^{-4}, respectively.

As the proposed GERF method combines information from the network structure and the attribute space, it can be compared with attributed representation learning methods for graphs. We choose the following:

- **TADW** – the learning rate was set to 0.01 and the number of epochs to 20, the other parameters were taken as defaults from the *Karate Club* implementation [1].
- **FSCNMF** – the number of epochs was set to 500, other settings are also default from the *Karate Club* implementation.
- **DGI** – we use a single layer Graph Convolutional (GCN) encoder network with PReLU activation and train it using the Adam optimizer with a learning rate of 0.001.

In addition to the above-mentioned representation learning methods, an approach in which the graph structure is completely ignored and only node attributes are used for prediction is additionally tested. Such a representation of

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_13

nodes is high-dimensional and extremely sparse, and in experiments, it will be referred to as **features**.

5.3 Baselines

To check the quality of the proposed method, which allows for modifying the existing structural embeddings based on node attributes, it is compared with several baseline methods:

- **Concat** – concatenation of the structural embedding and the attribute vector for each node (note the large dimension size of such a representation),
- **ConcatPCA** – in this method, the dimensionality of the concatenated structural embedding and the feature vector is reduced to the size of the embedding only, using *Principal Component Analysis*,
- **MLP** – a simple autoencoder architecture, with an encoder consisting of three linear layers: the first one with the size of $d_X + d_Z$ neurons with the ReLU activation function, another with the size of $(d_X + d_Z)//2$ also with the ReLU activation function, and the last one with the size of d_Z with the Tanh activation. The decoder is a single linear layer that takes a vector from a hidden space with dimension d_Z and returns a vector of size $d_X + d_Z$, which should be the best reconstruction of the input vector to the encoder. The autoencoder was trained for 20 epochs with the mini-batch size of 128, and Adam with learning rate of 0.001 was used as the optimizer.

5.4 GERF hyperparameters

The hyperparameter values (λ_G, λ_X) of the proposed GERF method were determined by performing a grid search (see Table 2). For each of the hyperparameters, we checked the following values: $0, 0.1, \ldots, 1.0$, while preserving the overall hyperparameter constraints ($\lambda_G + \lambda_X \leq 1$). Moreover, the learning rate was set to 10^{-1}, while the number of epochs was set to 100.

Table 2. Best found GERF hyperparameter values (λ_G, λ_X) for all datasets.

Dataset	node2vec (λ_G, λ_X)	LINE (λ_G, λ_X)	SDNE (λ_G, λ_X)
WikiCS	(0.1, 0.4)	(0.4, 0.4)	(0.3, 0.4)
Amazon-CS	(0.2, 0.3)	(0.3, 0.2)	(0.8, 0.0)
Amazon-Photo	(0.2, 0.4)	(0.3, 0.3)	(0.5, 0.2)
Coauthor-CS	(0.2, 0.5)	(0.6, 0.3)	(0.9, 0.0)

While performing the grid search, we collected the node classification performance for each hyperparameter setting (not only the best one). We present

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_13

the influence of each hyperparameter on the overall node classification performance in Figure 2 (for the WikiCS dataset). We notice that the results for both hyperparameters form a convex function with a single value that maximizes the downstream task performance. We also note that the proposed GERF method is robust in terms of hyperparameters. However, it is worth optimizing them as we observed up to 5 pp dispersion in the hyperparameter combinations impact on AUC (depending on the structural embeddings – LINE, node2vec, SDNE).

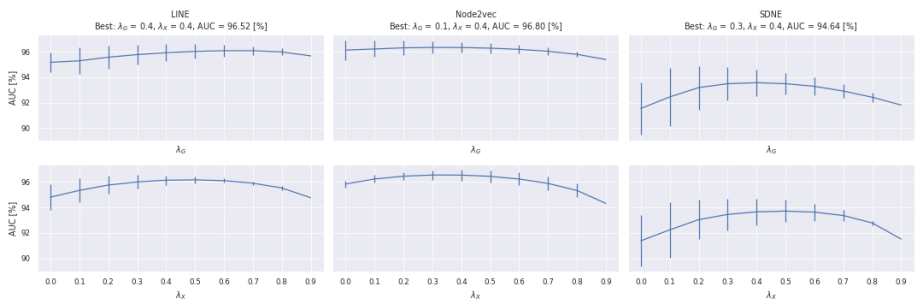

Fig. 2. Evaluation of different hyperparameter (λ_G, λ_X) values of the proposed GERF method in the node classification task on the WikiCS dataset. We present the mean and standard deviation of AUC (validation split) for each of the hyperparameter values. Note that while keeping one parameter fixed, we compute the AUC statistics over all possible values of the other hyperparameter.

6 Node classification

Setup The embeddings returned by the attributed representation learning methods, structural embeddings (themselves and enhanced by the proposed GERF method and baselines), as well as node attribute vectors, were compared in the node classification task. We compute the embeddings of both datasets 10 times to mitigate the random nature of the methods and their optimization procedure for both the structural and attributed graph representation learning methods. Each of those 10 embeddings is processed by the baselines and the proposed GERF model. We use a L_2 regularized logistic regression (from the *scikit-learn* package) trained on the embedding vectors (input) and the class information (output). The maximum number of iterations was set to 250, other parameters were left with their default values.

The classification results in terms of the AUC metric are shown in Table 3 For each of the three structural node embedding methods – node2vec, LINE and SDNE – as well as the embeddings updated by the baselines and our proposed GERF method – the best result is marked in bold. We report both the mean and standard deviation over 10 embedding recalculations.

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_13

Table 3. Node classification results in terms of the mean and standard deviation of the AUC metric over 10 recomputations of embeddings. For each structural embedding method (node2vec, LINE and SDNE) and their updated versions (by baselines and our proposed GERF method) we mark the best result in bold.

Method	WikiCS	Amazon-CS	Amazon-Photo	Coauthor-CS
features	94.79 ± 0.00	90.10 ± 0.00	94.58 ± 0.00	98.16 ± 0.00
node2vec	93.97 ± 0.15	98.22 ± 0.04	98.64 ± 0.04	98.33 ± 0.04
Concat (node2vec)	96.25 ± 0.10	98.23 ± 0.04	98.65 ± 0.04	98.38 ± 0.04
ConcatPCA (node2vec)	96.01 ± 0.11	98.22 ± 0.04	98.64 ± 0.04	98.34 ± 0.04
MLP (node2vec)	96.03 ± 0.08	98.29 ± 0.04	98.66 ± 0.04	98.41 ± 0.04
GERF (node2vec)	**96.28 ± 0.09**	**98.65 ± 0.04**	**99.18 ± 0.03**	**99.23 ± 0.02**
LINE	91.74 ± 0.20	97.63 ± 0.06	98.44 ± 0.08	93.13 ± 0.28
Concat (LINE)	95.02 ± 0.15	97.65 ± 0.06	98.45 ± 0.08	93.69 ± 0.26
ConcatPCA (LINE)	94.68 ± 0.14	97.64 ± 0.06	98.44 ± 0.08	93.16 ± 0.28
MLP (LINE)	94.90 ± 0.12	97.55 ± 0.07	98.45 ± 0.06	93.39 ± 0.24
GERF (LINE)	**96.18 ± 0.05**	**98.28 ± 0.05**	**99.06 ± 0.03**	**98.39 ± 0.05**
SDNE	74.94 ± 0.71	88.24 ± 0.41	90.89 ± 0.34	67.05 ± 1.05
Concat (SDNE)	**94.14 ± 0.27**	88.81 ± 0.41	91.34 ± 0.33	**93.86 ± 0.68**
ConcatPCA (SDNE)	93.63 ± 0.31	88.46 ± 0.43	91.16 ± 0.33	92.44 ± 0.73
MLP (SDNE)	93.75 ± 0.27	87.84 ± 0.42	90.55 ± 0.30	68.13 ± 0.86
GERF (SDNE)	92.97 ± 0.74	**97.49 ± 0.07**	**98.43 ± 0.08**	87.37 ± 2.86
TADW	90.65 ± 0.00	58.71 ± 0.00	55.91 ± 0.00	81.33 ± 0.00
FSCNMF	84.24 ± 0.00	49.93 ± 0.00	49.56 ± 0.00	50.14 ± 0.00
DGI	93.54 ± 0.17	78.20 ± 0.55	90.02 ± 0.54	98.48 ± 0.06

Discussion The first thing to note is the high score of the model that only uses node attributes to predict the class label and completely ignores information from the network structure despite the extremely large dimensionality of such a representation. In all cases, using only node attributes (features) for their classification gave better or similar results than the attributed representation learning methods. However, it should be noted that due to the long time of operation of these algorithms, the hyperparameter grid search was not performed, and their default values were adopted, which could have an impact on the results obtained.

Structural embedding vectors for WikiCS performed even worse than the attributed ones, particularly those learned using SDNE. However, the quality of the predictions increased significantly (in the case of SDNE, one could even say drastically) as they were processed by the baselines or the proposed GERF method, which can be seen in the increase of the AUC measure even by almost 18 percentage points (comparing SDNE and GERF (SDNE) embeddings for WikiCS)!

In general, in each case, the proposed GERF method allowed the incorporation of information from the attribute space into the structural embedding

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_13

space, improving the quality of prediction in this downstream task. Surprisingly good results were achieved with the use of this method on the node2vec and LINE embeddings, where it turned out to be not only better than the dedicated attributed methods, but also than all the proposed baselines, and in a significant way.

In the case of the SDNE embeddings group and the WikiCS and Coauthor-CS datasets, the baselines (*Concat*, *ConcatPCA* and *MLP*) methods turned out to be better than the proposed GERF method, but it is worth noting that both have some disadvantages. The concatenation of the attribute vector and structural embedding, which has proven to be best is highly dimensional and inconsistent with the structural embedding part. On the other hand, the *MLP* refiner, based on a simple autoencoder architecture, can exhibit problems when reconstructing sparse attribute vectors. All things considered, the results obtained by the proposed GERF method are satisfactory. While maintaining a low-dimensional representation, which allows saving memory, it achieves results similar or better to other methods, which depends on the quality of the underlying structural embeddings.

7 Conclusions

In this paper, we introduced a new graph representation learning problem setting, where given already precomputed structural node embeddings, we want to update them accordingly to node attributes, in such a way that the resulting embedding will preserve the information from both the structure and attributes. We proposed a novel graph embedding retrofitting model (GERF), which solves this problem by optimizing a compound loss function, which includes an invariance loss (keeping the new embedding close to the structural one), a graph neighborhood loss (which pushes embedding of neighboring nodes closer together) and a attribute neighborhood loss (which decreases the distance of embeddings of nodes with similar attributes). We evaluate this method on four real-world benchmark datasets (WikiCS, Amazon-CS, Amazon-Photo and Coauthor-CS), comparing it to attributed graph representation learning methods and other baselines and find that our method allows to enhance structural embeddings and results in better downstream node classification performance. In all cases, our method achieves the best results compared to other attribute aware embedding methods as well as for all datasets. In future, we want to find a way to automatically determine the hyperparameters (λ_G and λ_X) of our method based on the available graph data.

References

1. Karate Club: An API Oriented Open-source Python Framework for Unsupervised Learning on Graphs. https://github.com/benedekrozemberczki/karateclub
2. PyTorch geometric main page. https://pytorch-geometric.readthedocs.io/en/latest/index.html

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_13

3. Bandyopadhyay, S., Kara, H., Kannan, A., Murty, M.: FSCNMF: Fusing Structure and Content via Non-negative Matrix Factorization for Embedding Information Networks (04 2018)
4. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics **5**, 135–146 (2017)
5. Gao, H., Huang, H.: Deep Attributed Network Embedding. In: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18. pp. 3364–3370 (2018). https://doi.org/10.24963/ijcai.2018/467
6. Grover, A., Leskovec, J.: Node2vec: Scalable Feature Learning for Networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. p. 855–864. KDD '16, Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2939672.2939754
7. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 770–778 (2016)
8. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015)
9. McAuley, J., Targett, C., Shi, Q., van den Hengel, A.: Image-based recommendations on styles and substitutes. In: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval. p. 43–52. Association for Computing Machinery (2015). https://doi.org/10.1145/2766462.2767755
10. Mernyei, P., Cangea, C.: Wiki-cs: A wikipedia-based benchmark for graph neural networks. arXiv preprint arXiv:2007.02901 (2020)
11. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient Estimation of Word Representations in Vector Space. In: Bengio, Y., LeCun, Y. (eds.) 1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings (2013)
12. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 701–710. KDD '14, ACM (2014)
13. Sinha, A., Shen, Z., Song, Y., Ma, H., Eide, D., Hsu, B.J.P., Wang, K.: An overview of microsoft academic service (mas) and applications. In: Proceedings of the 24th International Conference on World Wide Web. p. 243–246. WWW '15 Companion, Association for Computing Machinery, New York, NY, USA (2015). https://doi.org/10.1145/2740908.2742839, https://doi.org/10.1145/2740908.2742839
14. Tan, M., Le, Q.: EfficientNet: Rethinking model scaling for convolutional neural networks. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 97, pp. 6105–6114. PMLR (09–15 Jun 2019)
15. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: LINE: Large-scale Information Network Embedding. Proceedings of the 24th International Conference on World Wide Web (May 2015). https://doi.org/10.1145/2736277.2741093, http://dx.doi.org/10.1145/2736277.2741093
16. Wang, D., Cui, P., Zhu, W.: Structural Deep Network Embedding. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_13

Data Mining. p. 1225–1234. KDD '16, Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2939672.2939753
17. Yang, C., Liu, Z., Zhao, D., Sun, M., Chang, E.Y.: Network representation learning with rich text information. In: Proceedings of the 24th International Conference on Artificial Intelligence. p. 2111–2117. IJCAI'15, AAAI Press (2015)
18. Zhang, Z., Yang, H., Bu, J., Zhou, S., Yu, P., Zhang, J., Ester, M., Wang, C.: ANRL: Attributed Network Representation Learning via Deep Neural Networks. In: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18. pp. 3155–3161. International Joint Conferences on Artificial Intelligence Organization (7 2018). https://doi.org/10.24963/ijcai.2018/438

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_13