Providing Node-level Local Explanation for node2vec through Reinforcement Learning

Hyunju Kang
Sungkyunkwan University
Suwon, Republic of Korea
neutor@skku.edu

Hogun Park*
Sungkyunkwan University
Suwon, Republic of Korea
hogunpark@skku.edu

ABSTRACT

Node representation learning is a technique to encode a network into a low-dimensional vector space while preserving the inherent relational properties in an embedding space. This approach is widely used in downstream tasks such as node classification, graph classification, and link prediction. Despite recent success in node-representation learning, it remains challenging to explain learned representations. In particular, many existing methods have focused less on *unsupervised* node representation learning methods (e.g., node2vec [3]); this makes it difficult to determine which subgraph contributes the most to the embedding of each node. In this paper, we propose a reinforcement-learning-based post-hoc and local explanation method that can identify the subgraph as an explanation of a target node. Compared to existing works, the explanation of our method has a greater impact on local neighbors when we measure the number of top-k nearest neighbors in the embedding space that are changed after perturbing the found subgraph and re-learning the node2vec method. Experiments using two real-world networks show that our proposed method generates more important subgraphs than the existing baselines.

KEYWORDS

node embedding, node representation learning, explanation

ACM Reference Format:
Hyunju Kang and Hogun Park. 2022. Providing Node-level Local Explanation for node2vec through Reinforcement Learning. In *MLoG-WSDM '22, February, Virtual Event, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Machine learning with graphs has an important role in graph classification, node prediction, and link prediction tasks, and its importance is growing in applications such as vast sensor networks and social networks. For example, supervised node representation learning methods (e.g., [6, 22]) have been proposed, and research shows state-of-the-art performance in many supervised learning tasks. In addition, there have been many advances in the field of *unsupervised* node representation learning (e.g., [3, 13, 18, 19]). *Unsupervised* node representation learning finds low-dimensional representations of nodes in graphs so that it can be applied to various downstream tasks. An example of an *unsupervised* node representation learning algorithm is node2vec [3], which learns the embedding vectors of each node based on random walks; it is available to project nodes in Euclidean space by placing nodes that have similar neighbors close to each other.

Despite the increasing application of node representation learning models, it is still difficult to understand how the models derive the representations because of the characteristics of graph-structured data and the use of complex neural networks. Accordingly, there is a growing need to explain node representation learning models. Recently, explainable AI (XAI) algorithms for supervised learning with GNNs were proposed in [23, 24, 26]. In particular, in XGNN [24] and SubgraphX [26], reinforcement learning was used to generate a subgraph to explain the predictive models. However, they are still limited to supervised node representation models.

There are a few methods for unsupervised node representation learning models that explain how the models derive the results. Previously, to interpret the results, we often leverage predictive models with the labels available [3] or related taxonomy [8] to measure the quality of the learned representations. However, the approach is only limited to the specific downstream tasks, rather than the node representations themselves. To increase the trustworthiness of the results from unsupervised node representation learning, it is necessary to directly consider the underlying node representations.

In this paper, we propose a new method to provide post-hoc and local explanations for node representations of node2vec [3], which is a widely used unsupervised node representation learning model. We first define the notion of an important subgraph as an explanation of the target node. It can be measured by calculating the *importance score* of the subgraph using a perturbation method. By perturbating the edges of the input graph, which are equivalent to those in the candidate subgraph, we obtain a perturbed graph and re-learn the embedding vectors with the perturbed graph. To obtain the *importance score* of the subgraph, we measure how much two embedding vectors of the target node (i.e., with or without perturbation) are locally different. However, there exist numerous possibilities of subgraphs around a target node, and it is difficult to obtain the most important subgraph to the original embedding efficiently. To address this problem, we utilize the Monte Carlo tree search (MCTS) in reinforcement learning to search for the most important subgraph efficiently. In our evaluation with two real-world graph datasets, the explanation of our proposed method has a greater impact on local neighbors compared to those of the

*Corresponding author

existing baselines. In particular, when we measure the number of top-k nearest neighbors in the embedding space that are changed after perturbing the found subgraph and re-learning the node2vec method, our proposed method changes top-k local neighbors twice as good as the baselines.

2 RELATED WORK

2.1 Unsupervised node representation learning

Unsupervised node representation learning extracts feature vectors of nodes without label information, and to achieve this, random walk-based algorithms, such as DeepWalk [12] and node2vec [3], are widely used. Random walks are employed to transform graph-structured data into a sequence of nodes by traveling neighborhoods with biased or unbiased transition probabilities. In DeepWalk, the distribution of the transition probability is uniform, leading to the movement of nodes in an unbiased manner. The method is generalized in node2vec by varying the transition probability in an unbiased or biased way using the return parameter p and in–out parameter q. p and q can present the different strategies to determine whether it is closer to BFS or DFS-like exploration. Notably, DeepWalk can be regarded as a special case of node2vec when setting p and q as 1. When we generate walks from an input graph using an unbiased or biased random walk procedure, we can treat these sequence data as one document. Each walk can be the sentence as in the field of natural language processing enabling the application of these forms of data to skip-gram algorithms [10], which are commonly used for word embedding tasks. It has shown enhanced representations to capture the context of related nodes by optimizing a neighborhood-preserving likelihood objective. As a result, it is possible to encode the nodes of an input graph into low-dimensional vectors. These methods have been applied to many downstream tasks and show considerable success in improving the performance compared to previous baselines.

2.2 Explainable AI for node representation learning models

The provision of interpretable explanations is a crucial criterion when node representation learning models are applied in time-critical and cost-intensive areas such as biology [11] and medicine [27]. There is active research on the application of explainable AI in node representation learning models. XAI models can be categorized in terms of their scope [2]. First, global XAI models aim to understand the model from a broad perspective; conversely, local XAI models are used to explain the predictions of models from the perspective of specific instances. For example, PGM-Explainer [20] and XGNN [24] are global models, whereas GNN-explainers [23], GNN-LRP [14], and SubgraphX [26] are local explanation models, but those are applicable to supervised node representation models.

In particular, XGNN [24] and SubgraphX [26] provide explanations for GNN-based supervised representation learning models by finding subgraphs with reinforcement learning. XGNN [24] utilizes the policy gradient method in the same manner as the REINFORCE algorithm [21]. It sets the reward function as the probability of a label class from the predictive model that we want to explain, and the policy network generates a subgraph with a maximum value of the label probability, which means that it explains the predictive model. SubgraphX exploits the MCTS algorithm, and its action is to perturb (remove) one node until the minimum number of nodes is achieved. The value is calculated using the Shapley value [7], which shows high mutual information to predict the label class. In summary, various methods have been proposed to explain supervised node representation learning models; however, they are still limited in terms of their ability to provide explanations of unsupervised node representation learning models.

To resolve this limitation, in recent years, the authors of [8] attempted to interpret the results of network embedding, aiming to understand the distribution of nodes in the embedding space. They provide an explanation in the form of a taxonomy, which shows the inherent structure using hierarchical clustering. Given that it explains learned node embedding models, it is a global and post-hoc XAI method; it does not provide node-level local explanations to the learned network embeddings.

Algorithm 1 *EdgePerturbation*$(\mathcal{G}, \mathcal{G}_i)$

```
1: Input: Graph $\mathcal{G} = (V, E, W)$, Subgraph $\mathcal{G}_i = (V', E', W')$
2: Copy graph $\mathcal{G}$ as new graph $\mathcal{G}'' = (V'', E'', W'')$
3: for $e_k$ in $E'$ do
4:     Update the weight of new $\mathcal{G}$ as $W''_{e_k} = 0.5 * W''_{e_k}$
5: end for
6: Return new graph $\mathcal{G}''$
```

Algorithm 2 Computation of *Importance*$(emb, \mathcal{G}, \mathcal{G}_i, k, v)$

```
1: Input: $emb$ = embedding vector of an input graph $\mathcal{G}$, $\mathcal{G}_i$ = a candidate subgraph, $v$ = a target node, $k$ = the number of neighbors to consider in $kNN$
2: $G' \leftarrow EdgePerturbation(\mathcal{G}, \mathcal{G}_i)$
3: $emb' \leftarrow NodeEmbedding(\mathcal{G}')$  # Fast re-learning (Section 3.4)
4: $O \leftarrow kNN(emb, k, v)$  # find the top-k nearest neighbors of $v$ in $emb$
5: $N \leftarrow kNN(emb', k, v)$  # find the top-k nearest neighbors of $v$ in $emb'$
6: Return $|(set(O) - set(N)|/k$  # calculate the ratio of difference
```

3 OUR PROPOSE METHOD

3.1 Finding the important subgraph for node-level explanation

In this study, we assume that there are subgraphs that affect the learned node embedding of a target node at most. For example, in a citation network, a recent GPT-3 work can be explained by a subgraph, which is formed among GPT-2 and transformer-based methods. To obtain the importance scores of the subgraph, most models such as the GNNExplainer [23] and PGExplainer [9] use a perturbation method [25]. These perturbation methods usually measure the change in the prediction probabilities after perturbing the input graph. Similarly, in this study, we define the notion of importance in unsupervised representation learning methods using a perturbation method. For the perturbation, we weaken the edges of the input graph, which are equivalent to the edges of the subgraph, as a perturbation to observe the impact of the subgraph. The procedure is described in Algorithm 1.

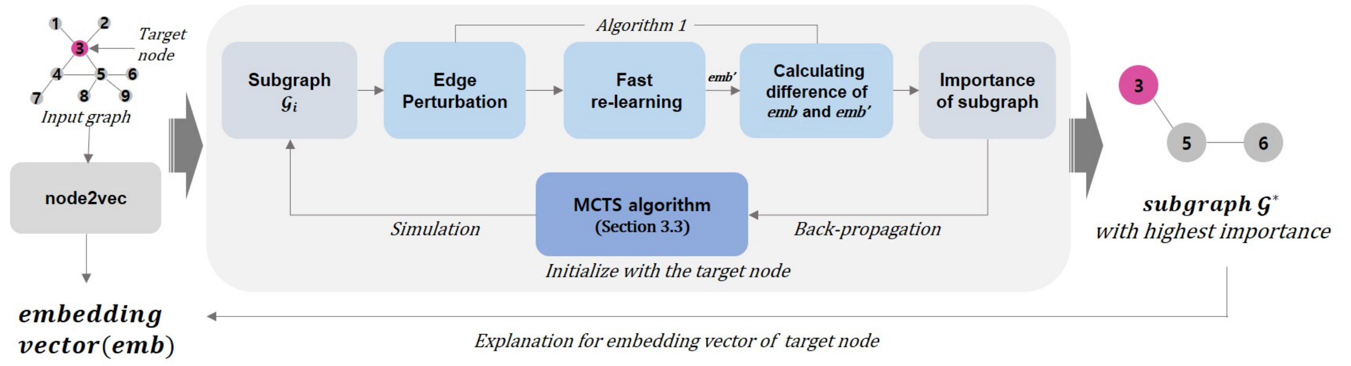

Target node
Input graph
node2vec
embedding vector(emb)
Algorithm 1
Subgraph $\mathcal{G}_i$
Edge Perturbation
Fast re-learning
emb'
Calculating difference of emb and emb'
Importance of subgraph
MCTS algorithm (Section 3.3)
Simulation
Back-propagation
Initialize with the target node
subgraph $\mathcal{G}^*$ with highest importance
Explanation for embedding vector of target node

Figure 1: Overall procedure for calculating the importance of the subgraph. To explain the node2vec embedding of node 3 (target node), our proposed method considers multiple candidate subgraphs via the MCTS. Given a subgraph, which is being considered, the original input graph is perturbed, and embeddings are re-learned on the perturbed graph. In Section 3.4, we propose a method to alleviate the re-learning burden. We then calculate the importance of the subgraph by measuring the local change in the embedding vectors and proceed to the next traversal (or terminate). (Best viewed in color.)

The perturbed graph is represented with different distributions in the updated embedding space after employing the node representation model, and we exploit the magnitude of change of the embedding vector between the original input graph and perturbed graph. If the subgraph plays an important role in the node embedding of the target node, the effect of perturbation would be significant; hence, the representation of the target node may be located in a different space. In other words, it has different surrounding neighbors in the new embedding space. We define the magnitude of the subgraph's importance as the change ratio of the surrounding neighbors in the new embedding space. To find the top-k nearest neighbors, we leverage an LSH hashing [15] to obtain the nearest neighbors efficiently. If the score is 0, there is no effect on the top-k neighboring nodes in the updated embedding space. Meanwhile, when the score is 1, all of the surrounding neighbors are changed because of the effect of the subgraph. Details of the entire procedure are given in Algorithm 2. Our goal is to search for the most important subgraph with the highest *importance score* as the local explanation for the node embedding vector of a target node v. It is defined as

$$\mathcal{G}^* = \arg\max_{\mathcal{G}_i} Importance(emb, \mathcal{G}, \mathcal{G}_i, k, v), \tag{1}$$

where emb is the node embedding vector, $\mathcal{G}$ is the input graph, and $\mathcal{G}_i$ is a set of subgraphs that include the target node v. k is a hyperparameter for the kNN function.

3.2 Illustrative example

Here, we use Zachary's karate club dataset to describe why the important score is related to the local impact. By setting a target node, we can compare the importance of two different subgraphs as an example. In Figure 2, our target node (node 8) is placed close to nodes 9, 15, 22, 28, and 30, which are shown with red lines in the embedding space learned from node2vec. In the case of A, subgraph 1, which consists of nodes 8 and 33, is a candidate explanation for node 8. After perturbing the edges of subgraph 1 from the input graph and re-learning the perturbed graph, we can obtain a new embedding vector. In the new embedding vector space, the top-5 nearest neighbor nodes of node 8 are 9, 13, 19, 28, and 30, which means that 40% of neighbors are changing. However, in the case of B, subgraph 2 does not affect the top-5 nearest neighbor nodes of the target node after perturbing and re-learning. Other nodes of the top-k nearest neighbors may change, but not for node 8. As a result, subgraph 1 plays a more important role in the embedding vector of node 8 than subgraph 2.

3.3 MCTS-G: MCTS-based explanation subgraph exploration method

There are numerous possible subgraphs for searching around a target node. Therefore, an efficient method for obtaining an important subgraph is required. In this paper, the Monte Carlo tree search (MCTS) [17] in reinforcement learning is leveraged to search for the subgraph with the highest importance. MCTS uses Monte Carlo simulations to accumulate value estimates that guide toward highly rewarding trajectories in the search space, and helps to avoid brute force searches; it exhibits good performance in many challenging tasks (e.g., the Go game [16]).

There are four main steps in the MCTS algorithm: 1) selection, 2) expansion, 3) simulation, and 4) backpropagation. When we initialize the MCTS algorithm, we set the target node as the root node in the tree graph and gradually expand the graph with neighboring nodes of the input graph.

Selection. Given the tree graph, in this study, the search strategy is to select the child node with a higher value using the following UCB1 formula [1] below.

$$UCB1 = v_i + C\sqrt{\frac{ln(N)}{n_i}} \tag{2}$$

In the aforementioned UCB1 formula, i indicates the i^{th} child node of the parent node, and v represents the importance of the subgraph. v_i is acquired by averaging the total value of the node and

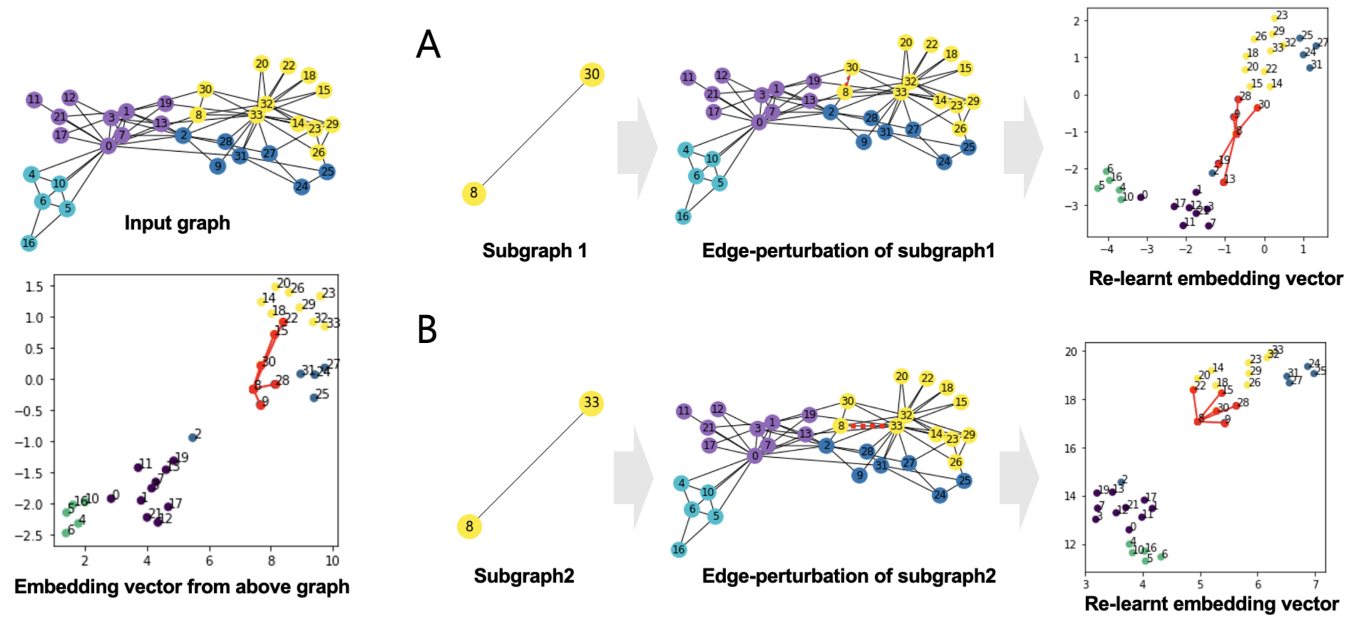


Figure 2: Comparison between two subgraphs with respect to the local importance to the learned embeddings. Cases A and B show two example subgraphs as candidate explanations for target node 8 on the Karate club datasets. As a result, subgraph 1 has a more important role in the embedding vector of node 8 rather than subgraph 2 because it changes the top k nearest neighbor nodes after perturbation and re-learning. (Best viewed in color.)

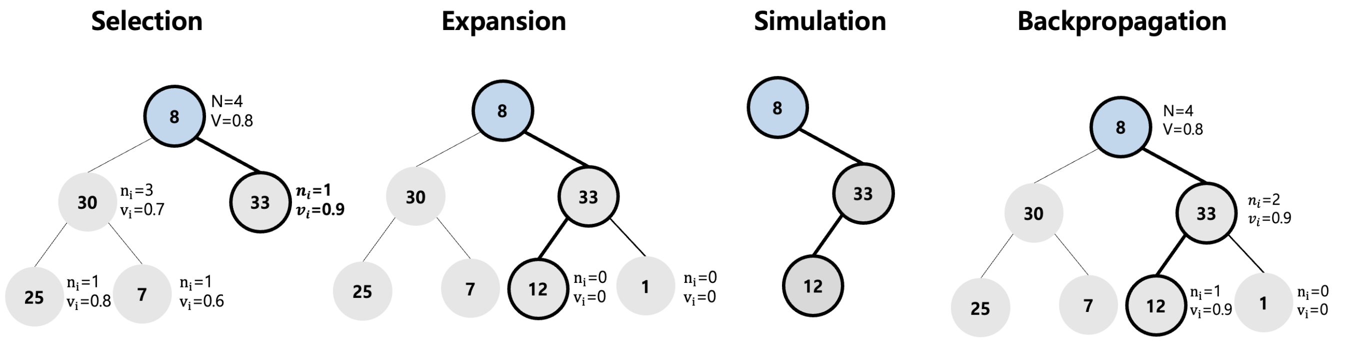


Figure 3: An example of our MCTS algorithm to search for the important subgraph for a target node 8. For the selection, we chose the child node with a higher UCB1 value. When we reach the leaf node with a visiting number that exceeds one, we expand its child node from its neighbor nodes in the input graph. Following the path from bottom to top, we generate the subgraph and update the importance of the subgraph as the value and visiting number into related nodes. (Best viewed in color.)

its child nodes. C is the exploration term that provides a chance to select the unvisited node, relaxing the exploration and exploitation issues. N is the total number of visiting counts in the current tree, and n_i is the visiting number of the i^{th} child node.

Expansion. When selecting the child node using the UCB1 formula until the leaf node is reached, we consider the number of visits to the leaf node to determine whether to expand the tree. When the number of visits to the leaf node is more than one, the tree satisfies the condition of expansion; hence, N nodes are randomly sampled among the leaf node's neighbors in the input graph.

Simulation. When we reach the unvisited leaf node, we generate a subgraph in the form of a path to travel from the bottom to the root node on top. The generated subgraph is regarded as the candidate of the important subgraph, and its *importance score* is measured using Algorithm 1.

Backpropagation. To update the obtained information, we renewed the value and visiting number by using the nodes of the generated subgraph. We simply add 1 to the visiting number of visited nodes; however, in the case of the value, we should average the importance score of the related node with the equation $\frac{\text{The sum of values}}{\text{The number of visits}}$. This process is repeated when the termination condition is satisfied. We pre-define the maximum number of nodes in the subgraph (i.e., *maxnode*) as the termination condition. When the number of nodes in the generated subgraph exceeds a predefined number, the termination is declared. As a result, we can find the importance subgraph with the highest value empirically among the numerous option candidates as the subgraph using MCTS algorithms. An example of our MCTS learning is shown in Figure 3.

3.4 Efficient re-learning

When we measure the importance of a subgraph, re-learning is mandatory in the current problem setting. To reduce the computational burden, we developed an efficient way to obtain the updated embedding vector. The main idea of our efficient re-learning method is to update only the renewed part, which is affected by perturbation. To obtain the new embedding, we leverage EvoNRL [5], which updates the random walk using an edge–index dictionary that indicates the location of edges in generated walks. Then, to apply renewed walks into the embedding vector, we fine-tune the neural weights using only the renewed part. The update strategy can reduce the time complexity to $O(at)$, where a is the number of affected random walks, and t is the number of epochs in fine-tuning. Notably, they are all constants, which are much smaller than the total number of random walks. Overall, the total time complexity of our model in the generation stage corresponds to $O(at \cdot |V| \cdot 2^{maxnode-1})$, where *maxnode* represents the number of possible nodes for an explanation and $|V|$ is the number of vertices. *maxnode* is given for the MCTS as a hyper-parameter, and $|V|$ is required to obtain the top-k nearest neighbors.

4 EXPERIMENTS

We evaluate our proposed method by computing how much found subgraphs are important to the input node embedding. In the evaluation, we use two real-world datasets; Zachary's karate club dataset as a social network and Cora dataset [3] as a citation network. For an evaluation metric, the subgraph importance score (as in Algorithm 2) is used to measure the local impact with respect to target nodes, and our model is compared to other baselines. We note that the fast embedding update method (Section 3.4) is not used when evaluating the found subgraphs.

4.1 Experimental setting

Common parameters for all experiment are following; p=1, q=1, and dimension=128 for node2vec [3]. For other hyper-parameters on the Karate club dataset, we set the parameters as window-size=2, num-walks=300, walk-length=5, top-k neighbors=5, and C=1.5 for a better separation among nodes. In the experiment using the Cora dataset, we set the parameters as window-size=5, num-walks=10, walk-length=100, top-k neighbors=10, and C=1. For learning the node2vec, there are various random seeds in the generator of random walks and samplers of the skip-gram model. All of them should be controlled strictly to capture the impact of the subgraph. Furthermore, when training the skip-gram model, we need to set the number of workers as one. By controlling the random seed initialization, we could achieve robust reproducibility in all experiments.

Table 1: Averaged importance scores of the found explanations in the Karate dataset. The values 3, 5, and 7 represent *maxnode* when subgraphs are generated. We note that *Taxonomy induction* does not consider *maxnode*.

Model	3	5	7	Average
Taxonomy induction	-	-	-	0.287
RW-G*	0.294	0.3	0.294	0.296
RW-G	0.329	0.365	0.382	0.359
MCTS-G* (Our model)	0.435	0.594	0.647	0.559
MCTS-G (Our model)	0.518	0.618	0.7	0.612

4.2 Baseline methods

To demonstrate the performance of our proposed method, we compare our proposed method to the existing post-hoc explanation methods and data-level explanation methods as below.

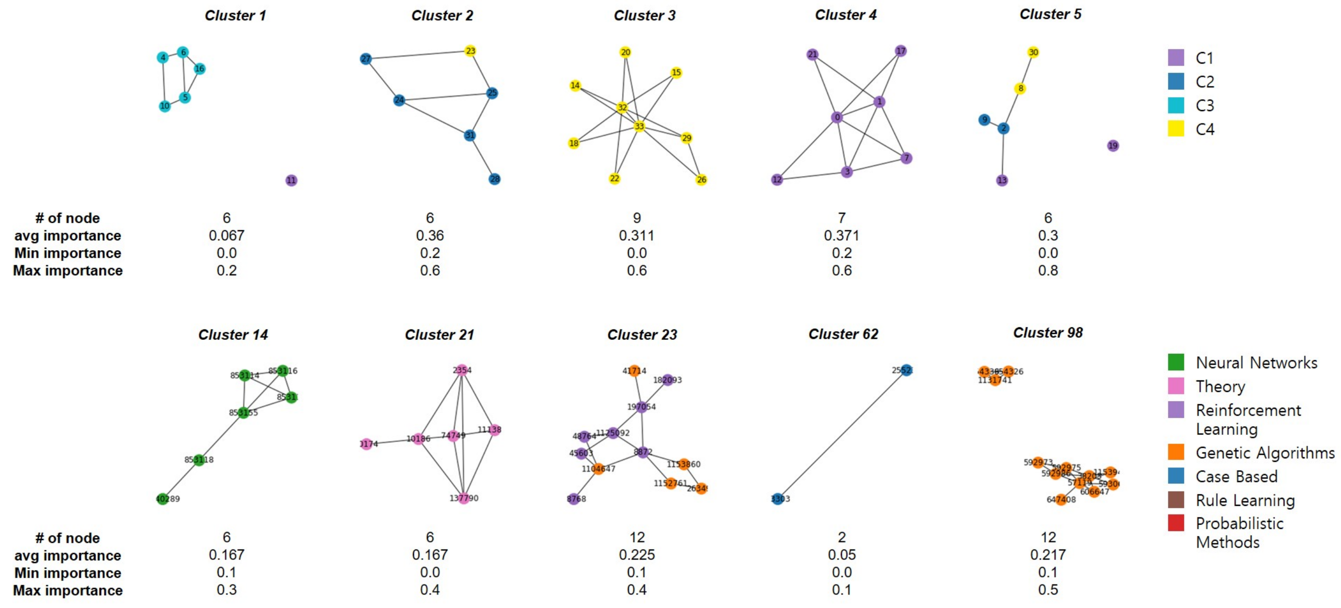
First, we use the Taxonomy induction method [8] as a baseline by utilizing the clusters that are obtained from the graph construction on the embedding vector. Although it is designed to provide a global view of the learned embeddings, we expect that the cluster, which includes the target node, can also provide a meaningful post-hoc explanation with respect to the target node. We set the number of clusters in the Karate club dataset as 5 and in the Cora dataset as 100.

Second, by generating a random walk (RW) from a target node over the input graph, we can also leverage the RW for an (alternative) explanation of the target node. Because the RW can also represent neighbors of the target node and is used for learning the node2vec, this allows us to evaluate whether explanations of our model have meaningful gains. In other words, RWs can provide the *data-level* explanations in input graph space. We note that our MCTS-G finds *post-hoc* explanations in the learned embedding space. We summarize baselines as below:

- *Taxonomy induction* identifies an important subgraph that is a cluster, which includes a target node. The cluster is found from [8].
- RW-G* finds an important subgraph that is generated based on the random walk. For a fair comparison, the edge sampling of RW-G* was repeated until the number of nodes and edges in the graph was the same as that of our proposed method. Here, * means that we do not use the re-learning method (Section 3.4).
- RW-G finds an important subgraph, that is generated based on the random walk. For a fair comparison, the edge sampling of RW-G* was repeated until the number of nodes and edges in the graph was the same as that of our proposed method. Here, we use the re-learning method (Section 3.4).

The baselines above are compared to our methods:

- MCTS-G* finds an important subgraph, which is generated from our proposed method in Section 3.3. Here * means that we do not use the re-learning method (Section 3.4).
- MCTS-G finds an important subgraph, which is generated from our proposed method in Section 3.3. Here we use the re-learning method (Section 3.4).



	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
# of node	6	6	9	7	6
avg importance	0.067	0.36	0.311	0.371	0.3
Min importance	0.0	0.2	0.0	0.2	0.0
Max importance	0.2	0.6	0.6	0.6	0.8

	Cluster 14	Cluster 21	Cluster 23	Cluster 62	Cluster 98
# of node	6	6	12	2	12
avg importance	0.167	0.167	0.225	0.05	0.217
Min importance	0.1	0.0	0.1	0.0	0.1
Max importance	0.3	0.4	0.4	0.1	0.5

Figure 4: Generated explanations from a baseline, *Taxonomy induction*[8]. The above five clusters are from the Karate dataset, and the five clusters below are from the Cora dataset. We set the number of clusters in the Karate club dataset to five and in the Cora dataset to 100. In the Cora dataset, we sampled five clusters to explain the clustering results, and we measured their importance. In the case of the Karate club dataset, we used the same method to color the clusters in [3] because of the lack of pre-defined label information, as shown. In the case of the Cora dataset, we utilized pre-defined label information for color. The brown as rule-learning and the red as probabilistic methods are not shown in this figure. (Best viewed in color.)

Table 2: Averaged importance scores of the found explanations in the Cora dataset. The values 3, 5, and 7 represent *maxnode* when subgraphs are generated. We note that *Taxonomy induction* does not consider *maxnode*.

Model	3	5	7	Average
Taxonomy induction	-	-	-	0.195
RW-G*	0.171	0.164	0.205	0.180
RW-G	0.196	0.21	0.17	0.192
MCTS-G* (Our model)	0.269	0.298	0.343	0.303
MCTS-G (Our model)	0.276	0.356	0.4	0.344

4.3 Experimental Results

Table 1 and 2 show the results obtained from the experiment using the Karate club dataset and the Cora dataset. In the Karate club dataset, we average the importance score from all 34 nodes as target nodes. In the Cora dataset, we randomly select 10 nodes per 7 label classes. In both experiments, MCTS-G and MCTS-G* are significantly better than RW-G and RW-G* in terms of the importance scores. It also shows the tendency that when nodes of the subgraph increase, the gap between MCTS-G and RW-G is higher, which means that our proposed method works well in complex search spaces. The performance gap between our models (MCTS-G and MCTS-G*) and random-walk-based baselines (RW-G and RW-G*) becomes larger when the *maxnode* increases. This observation implies that our proposed MCTS-G and MCTS-G* find more precise and important subgraphs, which can change the local neighbors more.

Figure 4 shows the result of *taxonomy induction*. Each color represents true cluster IDs in the graph space. Even the result of clustering appears reasonable in the sense that nodes with similar labels are obtained. However, the average importance of subgraphs from clusters is not better than that of RW-G*, even without considering the number of nodes and edges. From this analysis, we can see that our proposed MCTS method enables us to have more locally important explanations. However, explanations of *taxonomy induction* are generated from clusters, which are found by all nodes, so they often do not produce target node-specific local explanations.

5 CONCLUSION

In this paper, a post-hoc and local explanation method is proposed to interpret the embedding vector of the target node in a random walk-based unsupervised node representation learning. Our proposed reinforcement-learning-based explanation method can identify the subgraph as a node-level explanation of a target node. In the experiments, the important subgraphs from our proposed method showed higher local impacts than the baseline methods. In future work, to evaluate our proposed method, we plan to apply it to other embedding models (e.g., unsupervised GNNs [4] and LINE [18]) and to study more real-world cases in recommender systems, prediction tasks, and other applications.

6 ACKNOWLEDGMENT

This work has supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (NRF-2021R1C1C1005407) and Institute of Information & communications Technology Planning & evaluation (IITP) grant (No. 2019-0-00421, Artificial Intelligence Graduate School Program, Sungkyunkwan University).

REFERENCES

[1] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine learning* 47, 2 (2002), 235–256.

[2] Arun Das and Paul Rad. 2020. Opportunities and challenges in explainable artificial intelligence (xai): A survey. *arXiv preprint arXiv:2006.11371*.

[3] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proc. of SIGKDD*. 855–864.

[4] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proc. of NeurIPS*. 1025–1035.

[5] Farzaneh Heidari and Manos Papagelis. 2020. Evolving network representation learning based on random walks.. In *Appl Netw Sci 5*. 1–38.

[6] Thomas N Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

[7] Stan Lipovetsky and Michael Conklin. 2001. Analysis of regression in game theory approach. *Applied Stochastic Models in Business and Industry* 17, 4 (2001), 319–330.

[8] Ninghao Liu, Xiao Huang, Jundong Li, and Xia Hu. 2018. On interpretation of network embedding via taxonomy induction. In *Proc. of SIGKDD*. 1812–1820.

[9] Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. 2020. Parameterized Explainer for Graph Neural Network. In *Proc. of NeurIPS*. 19620–19631.

[10] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proc. of NeurIPS*. 3111–3119.

[11] Walter Nelson, Marinka Zitnik, Bo Wang, Jure Leskovec, Anna Goldenberg, and Roded Sharan. 2019. To embed or not: network embedding as a paradigm in computational biology. *Frontiers in genetics* 10 (2019).

[12] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: Online Learning of Social Representations. In *Proc. of SIGKDD*. 701–710.

[13] Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo. 2017. struc2vec: Learning node representations from structural identity. In *Proc. of SIGKDD*.

[14] Thomas Schnake, Oliver Eberle, Jonas Lederer, Shinichi Nakajima, Kristof T Schütt, Klaus-Robert Müller, and Grégoire Montavon. 2021. Higher-order explanations of graph neural networks via relevant walks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).

[15] Anshumali Shrivastava and Ping Li. 2014. Asymmetric LSH (ALSH) for sublinear time maximum inner product search (MIPS). *arXiv preprint arXiv:1405.5869* (2014).

[16] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. 2017. Mastering the game of go without human knowledge. *nature* 550, 7676 (2017), 354–359.

[17] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

[18] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proc. of WWW*.

[19] Anton Tsitsulin, Davide Mottin, Panagiotis Karras, and Emmanuel Müller. 2018. VERSE: Versatile Graph Embeddings from Similarity Measures. In *Proc. of WWW*. 539–548.

[20] Minh N Vu and My T Thai. 2020. Pgm-explainer: Probabilistic graphical model explanations for graph neural networks. *arXiv preprint arXiv:2010.05788* (2020).

[21] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8, 3 (1992), 229–256.

[22] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *Proc. of ICLR*.

[23] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. 2019. Gnnexplainer: Generating explanations for graph neural networks. In *Proc. of NeurIPS*. 9240–9251.

[24] Hao Yuan, Jiliang Tang, Xia Hu, and Shuiwang Ji. 2020. XGNN: Towards Model-Level Explanations of Graph Neural Networks. In *Proc. of SIGKDD*. 430–438.

[25] Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. 2020. Explainability in graph neural networks: A taxonomic survey. *arXiv preprint arXiv:2012.15445*.

[26] Hao Yuan, Haiyang Yu, Jie Wang, Kang Li, and Shuiwang Ji. 2021. On explainability of graph neural networks via subgraph explorations. *arXiv preprint arXiv:2102.05152*.

[27] Marinka Zitnik, Francis Nguyen, Bo Wang, Jure Leskovec, Anna Goldenberg, and Michael M Hoffman. 2019. Machine learning for integrating data in biology and medicine: Principles, practice, and opportunities. *Information Fusion* 50 (2019).