On Interpretation of Network Embedding via Taxonomy Induction

Ninghao Liu,[†] Xiao Huang,[†] Jundong Li,[‡] Xia Hu[†]
[†]Department of Computer Science and Engineering, Texas A&M University
[‡]Computer Science and Engineering, Arizona State University
{nhliu43,xhuang,xiahu}@tamu.edu,jundongl@asu.edu

ABSTRACT

Network embedding has been increasingly used in many network analytics applications to generate low-dimensional vector representations, so that many off-the-shelf models can be applied to solve a wide variety of data mining tasks. However, similar to many other machine learning methods, network embedding results remain hard to be understood by users. Each dimension in the embedding space usually does not have any specific meaning, thus it is difficult to comprehend how the embedding instances are distributed in the reconstructed space. In addition, heterogeneous content information may be incorporated into network embedding, so it is challenging to specify which source of information is effective in generating the embedding results. In this paper, we investigate the interpretation of network embedding, aiming to understand how instances are distributed in embedding space, as well as explore the factors that lead to the embedding results. We resort to the post-hoc interpretation scheme, so that our approach can be applied to different types of embedding methods. Specifically, the interpretation of network embedding is presented in the form of a taxonomy. Effective objectives and corresponding algorithms are developed towards building the taxonomy. We also design several metrics to evaluate interpretation results. Experiments on real-world datasets from different domains demonstrate that, by comparing with the state-of-the-art alternatives, our approach produces effective and meaningful interpretation to embedding results.

KEYWORDS

Machine Learning Interpretation, Network Embedding, Taxonomy

ACM Reference Format:
Ninghao Liu,[†] Xiao Huang,[†] Jundong Li,[‡] Xia Hu[†]. 2018. On Interpretation of Network Embedding via Taxonomy Induction. In *KDD '18: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3219819.3220001

1 INTRODUCTION

Network embedding has been increasingly applied to learn the representation of network data before using off-the-shelf machine learning models to conduct advanced analytic tasks such as classification [22, 52], clustering [12, 57], link prediction [55] and recommendation [4, 23]. Network embedding projects nodes to a low-dimensional space in which each node is represented by a vector. Network embedding preserves certain structural and content information of original networks. Nodes that are similar to each other, with respect to the pre-defined proximity measures, are mapped to the neighboring regions in the embedding space.

Similar to many traditional machine learning techniques, network embedding also suffers from the problem of lacking interpretability. Usually each dimension in the embedding space does not have any specific meaning. Also, although network embedding can generate effective feature representation, we lack an overall comprehension of how embeddings distribute in the new space, due to the obscurity of the applied network embedding models. Interpretability plays a crucial role in many application scenarios. On one hand, as heterogeneous information sources such as links [46, 52], attributes [25, 58], labels [26, 29, 56] and local structures [11, 22, 52] are incorporated into network embedding algorithms in computing the similarity between nodes, interpretation approaches can provide clues about which information is important in producing the outcome, and whether it is in accordance with the application [47]. On the other hand, the "black box" nature of a model may impede users from trusting the generated analytical results [37]. For example, many recommender systems map users and products into embedding spaces, followed by some matching algorithms to recommend products to users. Users may better trust the recommendation results if the underlying reasons could be specified [8]. Thus we propose to investigate the important problem of enabling interpretation in network embedding.

Understanding network embedding is a nontrivial task due to unique properties of the problem and characteristics of the network data. First, we cannot directly apply existing interpretation methods designed for prediction models [5, 20, 47]. These methods require class labels of instances, which are however not available from embedding results. Second, many real-world networks are of large volume, contain diverse information and tend to be noisy. These data characteristics require the interpretation schema to effectively utilize various information sources and process them efficiently. Third, although visualization techniques can be applied to understand network embedding results [21, 41, 50], it is not intuitive for users with limited data science background to manually discover complex patterns from visualization. Some information even cannot be rendered merely through visualization.

To tackle the aforementioned challenges, in this paper, we propose a novel interpretation framework for understanding network

KDD '18, August 19–23, 2018, London, United Kingdom

ACM ISBN 978-1-4503-5552-0/18/08…$15.00
https://doi.org/10.1145/3219819.3220001

embedding. The approach is post-hoc, i.e., we focus on interpreting the given network embedding result, so that it is applicable to different types network embedding methods. Unlike many existing frameworks that provide local interpretation to individual instances [3, 47], our method focuses on capturing the global characteristics of embedding result. Our interpretation of network embedding is presented in the form of a taxonomy. We first extract the backbone of the taxonomy to know how instances are distributed in the embedding space, and then provide descriptions to different concepts in the taxonomy by utilizing the property of network homophily. Proper data structures and algorithms are presented to improve the efficiency of the approach. The resultant interpretation, together with visualization tools, could provide richer information to end users. The major contributions of this paper are as follows:

- We design a model-agnostic interpretation framework to understand network embedding result through taxonomy induction.
- We propose clear objectives in each step of the taxonomy induction, and develop effective algorithms towards the objectives.
- We design new metrics for evaluating interpretation accuracy. Experiments on real-world networks are conducted to demonstrate the effectiveness of the proposed approach.

2 PROBLEM FORMULATION

2.1 Notations

We use boldface uppercase alphabets (e.g., $\mathbf{A}$) to denote matrices, boldface lowercase alphabets (e.g., $\mathbf{z}$) as vectors, calligraphic alphabets (e.g., $\mathcal{T}$) as sets, and normal characters (e.g., i, K) as scalars. The size of a set $\mathcal{T}$ is denoted as $|\mathcal{T}|$. The i-th row, j-th column and (i,j) entry of matrix $\mathbf{A}$ are denoted as $\mathbf{A}_{i,:}$, $\mathbf{A}_{:,j}$ and $\mathbf{A}_{i,j}$, respectively. Let $\mathcal{N} = \{\mathcal{V}, \mathcal{E}, \mathbf{X}\}$ be the input network, where $\mathcal{V}$ denotes the set of nodes, $\mathcal{E}$ denotes the edge set, and $\mathbf{X} \in \mathbb{R}^{N \times M}$ denotes the attribute matrix. Specifically, the network has N nodes, and each node is associated with M attributes. Some examples of node attributes include biographical information of users in social networks, and reviews of products in co-purchase networks. The output of network embedding is the representation matrix $\mathbf{Z} \in \mathbb{R}^{N \times D}$, where $\mathbf{Z}_{i,:} \in \mathbb{R}^{D}$ denotes the *embedding instance* of the i-th node. In this paper, we assume there is only one type of nodes and relations in the network, but the work can be extended to heterogeneous networks with various types of nodes and relations.

2.2 Objectives of Network Embedding

In order to provide directions in designing an appropriate interpretation method, we first elaborate the commonalities of different network embedding approaches. Specifically, a vast majority of existing network embedding approaches can be reduced to solving the following optimization problem:

$$\begin{aligned} &\min_{\mathbf{Z}} \quad \mathcal{L}_{emb} = \mathcal{L}_{app}(\mathcal{N}, \mathbf{Z}) + \alpha_0 \mathcal{L}_{reg}(\mathbf{Z}), \\ &\text{where} \quad \mathcal{L}_{app}(\mathcal{N}, \mathbf{Z}) = \sum_{i,j \in \mathcal{V}} l(s_{\mathcal{N}}(i,j), s_{\mathbf{Z}}(i,j)). \end{aligned} \tag{1}$$

Here $\mathcal{L}_{emb}$, $\mathcal{L}_{app}$ and $\mathcal{L}_{reg}$ denote the overall loss function, approximation loss function and regularization terms, respectively. α_0 is a balancing parameter. $s_{\mathcal{N}}(i,j)$ and $s_{\mathbf{Z}}(i,j)$ represent the similarity measure between node i and j in the original network and embedding space, respectively. l is the element-level loss function measuring the disparity between $s_{\mathcal{N}}(i,j)$ and $s_{\mathbf{Z}}(i,j)$. Examples of $s_{\mathbf{Z}}(i,j)$ include logistic error [52], square error [55] and inner product [56]. $s_{\mathcal{N}}(i,j)$ can be measured based on neighborhoods [11, 52], node attributes [25] and labels [29, 56]. For example, in [52], the objective function considering the first-order proximity is $-\sum_{(i,j)\in\mathcal{E}} w_{i,j} \log p(i,j)$, where $w_{i,j}$ is the edge weight and $p(i,j)$ is the probability between node i and j. Here $s_{\mathcal{N}}(i,j)$ and $s_{\mathbf{Z}}(i,j)$ correspond to $w_{i,j}$ and $p(i,j)$, and l is the KL-divergence. The similarity between nodes in the original network are thus encoded into the proximity between embedding instances. From the analysis above, we are interested in two aspects for understanding network embedding results: how do the embedding instances distribute in the latent space (i.e., which nodes are mutually adjacent or separated in the latent space), and what are the possible factors leading to the embedding distribution?

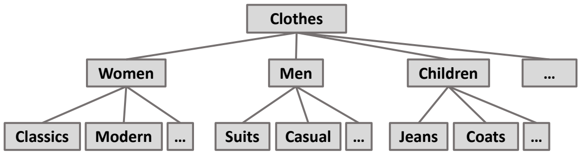


Figure 1: A toy example of taxonomy for "clothes".

2.3 Taxonomy Induction as Interpretation

We consider several elements when designing the interpretation schema. First, as different methods adopt different strategies for embedding representation learning, we leverage the post-hoc strategy [37] and attempt to extract explanation information from the obtained embedding results. Second, we focus on explaining the overall embedding results rather than providing local explanations for each individual node, due to the existence of autocorrelations [32] among connected nodes. Third, as community structures are naturally observed in real-world networks, it motivates us to explain the embedding results based on communities.

Considering the factors above, in this paper, we tackle the problem of explaining network embedding via *taxonomy induction*. A taxonomy is a structured organization of knowledge to facilitate the information searching. An example of taxonomy is shown in Figure 1, where different classes are organized in a hierarchical structure, and classes of coarse granularity are gradually split into refined ones. Following the existing work on taxonomy construction [10, 40, 44], we map the terminology in network analysis to taxonomy induction. We define the "domain" in a taxonomy as the whole data at hand, including network $\mathcal{N}$ and its embedding $\mathbf{Z}$. The attributes $\mathbf{X}$ and edges $\mathcal{E}$ in networks are regarded as "terms" as they describe the properties of nodes. The backbone of a taxonomy is usually a hierarchy of "concepts", which correspond to clusters implicitly contained in $\mathbf{Z}$. The embedding vectors of all nodes are divided into smaller clusters in an iterative manner. The "hypernym" relation in a hierarchy is modeled by the directed link between a parent cluster and child cluster. In this way, we distill the implicit relations and patterns concealed in embeddings into explicit organizations of knowledge.

Taxonomy induction tackles the two aspects of problems proposed in Section 2.2. The cluster hierarchy in the taxonomy can unveil how embedding instances distribute in the latent space, while summarizing the characteristics of each cluster discovers the factors that lead to such distribution. In this paper, in particular, we refer to the former aspect as the global-view interpretation, and the latter aspect as the local-view interpretation.

3 GLOBAL-VIEW INTERPRETATION

The goal of this section is to extract the backbone of the taxonomy based on the embedding instances. Concretely, the backbone is represented as a hierarchy of clusters in the embedding space. In this way, we can gain more insights about how nodes distribute in the embedding space, and gradually unveil the structural patterns among nodes in the embedded space.

3.1 Embedding-based Graph Construction and Clustering

Given the embedding result $\mathbf{Z}$, we first build a graph G, whose affinity matrix is denoted as $\mathbf{A}^G$, to store the node-to-node similarity in the embedded space. The edge weight between each pair of nodes is defined as:

$$\mathbf{A}^G_{i,j} = \exp(-\|\mathbf{Z}_{i,:} - \mathbf{Z}_{j,:}\|_2^2/2\sigma^2). \tag{2}$$

Since computing and storing weights for all pairs of nodes could be extremely expensive when the number of nodes is large, here we only maintain a sparse affinity matrix defined as follows:

$$\mathbf{A}_{i,j} = \begin{cases} \mathbf{A}^G_{i,j}, & \text{if } j \in \text{Neighbors}(i) \text{ or } i \in \text{Neighbors}(j) \\ 0, & \text{otherwise} \end{cases}, \tag{3}$$

where the number of neighbors is $|\text{Neighbors}(\cdot)| = b\lceil \log_2 N \rceil$ according to [54], and b is a constant integer. The resultant affinity matrix $\mathbf{A}$ is symmetric. The cluster structure of embedding instances is obtained by solving the problem below:

$$\min_{\mathbf{W} \geq 0} \|\mathbf{A} - \mathbf{W}\mathbf{W}^T\|_F^2, \tag{4}$$

where $\mathbf{W} \in \mathbb{R}^{N \times C}$, C is the number of clusters and $\|\cdot\|_F$ denotes Frobenius norm. The above formulation is equivalent to performing kernel K-means clustering on $\mathbf{Z}$ with the orthogonality constraint $\mathbf{W}^T\mathbf{W} = \mathbf{I}$ relaxed [18]. A larger value of $\mathbf{W}_{i,c}$ indicates a stronger affiliation of the node i to cluster c.

It is a nontrivial problem to solve Equation 4, since the objective function is a fourth-order function which is non-convex with respect to $\mathbf{W}$, so we reformulate the problem as below:

$$\min_{\mathbf{W},\mathbf{H} \geq 0} \|\mathbf{A} - \mathbf{W}\mathbf{H}^T\|_F^2 + \alpha\|\mathbf{W} - \mathbf{H}\|_F^2, \tag{5}$$

where $\alpha > 0$ controls the tradeoff between approximation error and factor matrices difference. Traditional iterative optimization methods such as coordinate descent methods [24, 36] can be applied to solve the problem by updating $\mathbf{W}$ and $\mathbf{H}$ alternatively. The value of α is gradually increased to reduce the difference between $\mathbf{W}$ and $\mathbf{H}$ as iterations proceed until convergence.

3.2 Hierarchy Establishment from Embedding-based Graph

Rather than performing conventional flat clustering on G, we leverage the hierarchical strategy to resolve the graph recursively. The reasons are two-fold. First, it is hard to know the optimal number of clusters in G. By adopting hierarchical clustering, we avoid starting over and exhaustively trying different K values, which is often time-consuming. Second, multilevel abstraction of concepts naturally exists in many real-world networks, such as product catalogues in e-commerce networks, and topic hierarchies in document networks. In addition, practitioners can trim the obtained hierarchical structure based on their own needs, which is helpful in interpretation where the interactions exist between users and models.

Algorithm 1: Taxonomy Backbone Extraction from G

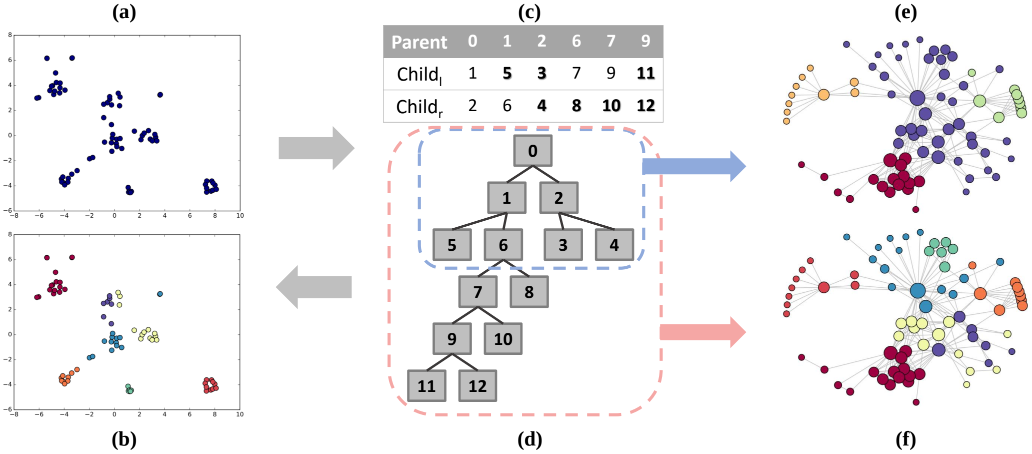
```
   Input: Affinity matrix $\mathbf{A}$ of $G$, maximum number of clusters $C$,
          parameters $\alpha \in \mathbb{R}_+$, $\gamma \in (0, 1)$, $I \in \mathbb{N}_+$
   Output: Tree-structured hierarchy $\mathcal{T}$
1  Create a root cluster $\mathcal{M}^1 = \mathcal{V}$ containing all graph nodes, and let
     the partition score $s(\mathcal{M}^1) \leftarrow 0$ (Note: $0 \leq s(\cdot) \leq 1$)
2  Number of leaf node $c \leftarrow 1$, time step $t \leftarrow 1$
3  while $c \leq C$ do
4      Choose a cluster $\mathcal{M}^t$ of the smallest $s(\mathcal{M}^t)$ to be partitioned
5      Outliers $\mathcal{O}^t = \emptyset$
6      for $i = 1 : I$ do
7          Obtain the submatrix $\mathbf{A}^t$ corresponding to $\mathcal{M}^t$
8          Run rank-K NMF on $\mathbf{A}^t$, and create $K$ potential clusters
             $\{\mathcal{M}^t_k; k \in [1, K]\}$ based on $\mathbf{W}^t$ ( or $\mathbf{H}^t$)
9          if $|\mathcal{M}^t_k| < \gamma|\mathcal{M}^t|$ && $s(\mathcal{M}^t_k)$ is the largest among all leaf
             clusters, $\exists k \in [1, K]$, then
10             $\mathcal{M}^t \leftarrow \mathcal{M}^t - \mathcal{M}^t_k$, $\mathcal{O}^t \leftarrow \mathcal{O}^t \cup \mathcal{M}^t_k$
11         else
12             break
13     if $i \leq I$ then
14         Partition $\mathcal{M}^t$ as $\{\mathcal{M}^t_k; k \in [1, K]\}$, and compute $s(\mathcal{M}^t_k)$
15     else
16         $\mathcal{M}^t \leftarrow \mathcal{M}^t \cup \mathcal{O}^t$, $s(\mathcal{M}^t) \leftarrow 1$
17     $c \leftarrow c + K$, $t \leftarrow t + 1$
```

We adapt the previously mentioned NMF algorithm for clustering G in a hierarchical manner, as summarized in Algorithm 1. After initialization (line 1~2), we repeatedly split large clusters into smaller ones. An example can be found in Figure 2. The details of the hierarchical clustering process are introduced as follows:

- **Partition Score** (line 4): At each step t, we need to choose the "best" cluster to be partitioned. A cluster is suitable for further partitions if it contains several smaller cluster components which are densely intra-connected and loosely inter-connected. We use the normalized cut (*ncut*) [49] as the partition score $s(\cdot)$:

$$s(\mathcal{M}^t) = ncut(\{\mathcal{M}^t_k; k \in [1,K]\}) = \sum_{k=1}^{K} \frac{cut(\mathcal{M}^t_k, \mathcal{V} - \mathcal{M}^t_k)}{assoc(\mathcal{M}^t_k, \mathcal{V})}, \tag{6}$$

  where $cut(\mathcal{M}_1, \mathcal{M}_2) = \sum_{i \in \mathcal{M}_1, j \in \mathcal{M}_2} \mathbf{A}_{i,j}$ and $assoc(\mathcal{M}, \mathcal{V}) = \sum_{i \in \mathcal{M}, j \in \mathcal{V}} \mathbf{A}_{i,j}$. In this way, if each sub-cluster $\mathcal{M}^t_k$ is well isolated from other nodes, then $cut(\mathcal{M}^t_k, \mathcal{V} - \mathcal{M}^t_k)$ will be small. Besides, if the nodes within $\mathcal{M}^t_k$ are well connected, then $assoc(\mathcal{M}^t_k, \mathcal{V})$ is large, since $assoc(\mathcal{M}^t_k, \mathcal{V}) = assoc(\mathcal{M}^t_k, \mathcal{M}^t_k) + cut(\mathcal{M}^t_k, \mathcal{V} - \mathcal{M}^t_k)$. Therefore, the cluster with smallest partition score s is assigned with top priority to be further split.
- **Rank-K NMF** (line 7~8, 14): Let $\mathbf{A}^t \in \mathbb{R}^{n \times n}$ denote the affinity matrix of cluster $\mathcal{M}^t$ which is selected to be further partitioned



Parent	0	1	2	6	7	9
Child$_l$	1	**5**	**3**	7	9	**11**
Child$_r$	2	6	**4**	**8**	**10**	**12**

Figure 2: A taxonomy extracted by hierarchical clustering on embeddings from node2vec [22] on the Les-Misérables network. (a): Visualization of original embedding results. (b): Visualization of embeddings after clustering where $C = 7$. (c): Taxonomy represented in a table, where bold numbers denote leaves. (d): Taxomony represented by a tree. (e): Visualization of the network with 4 clusters obtained from a subtree. (f): Visualization of the network with 7 clusters obtained from the taxonomy, which corresponds to the embedding visualization in figure (b).

at step t. The size of $\mathcal{M}^t$ is n. We fix the number of sub-clusters as K, so each partition step is transformed into a local rank-K NMF problem: $\min_{\mathbf{W}^t,\mathbf{H}^t\geq 0} \|\mathbf{A}^t - \mathbf{W}^t\mathbf{H}^{tT}\|_F^2 + \alpha\|\mathbf{W}^t - \mathbf{H}^t\|_F^2$. For generality, in this work we set $K = 2$ if no other prior knowledge is available. After initializing $\mathbf{H}^t$ and α, the NMF problem is solved in an iterative manner until convergence [31]:

$$\mathbf{W}^t \leftarrow \mathbf{H}^t,\quad \alpha \leftarrow c_{scale}\cdot\alpha$$

$$\mathbf{H}^t \leftarrow \operatorname*{argmin}_{\mathbf{H}^t\in\mathbb{R}^{n\times K}_{\geq 0}} \left\| \begin{bmatrix}\mathbf{A}^t\\ \sqrt{\alpha}\mathbf{W}^{tT}\end{bmatrix} - \begin{bmatrix}\mathbf{W}^t\\ \sqrt{\alpha}\mathbf{I}_K\end{bmatrix}\mathbf{H}^{tT}\right\|_F^2, \qquad (7)$$

where c_{scale} is slightly larger than 1, so that α keeps increasing throughout iterations to force $\mathbf{H}^t$ and $\mathbf{W}^t$ to be gradually close to each other. Here $\mathbf{I}_K$ is the $K \times K$ identity matrix. After convergence, we assign each node i to the new sub-cluster $k_i = \operatorname{argmax}_k \mathbf{W}_{i,k}$. In the taxonomy tree $\mathcal{T}$, the new K sub-clusters are appended as the children of $\mathcal{M}^t$.

- **Outliers Identification** (line 9~12, 16): Outliers adversely affect the clustering quality as they are likely to be separated as sub-clusters but usually do not contain patterns of interest. We treat a sub-cluster $\mathcal{M}_k^t$ as outliers if it is of extremely small size compared with its parent (i.e., $|\mathcal{M}_k^t| < \gamma|\mathcal{M}^t|$ and $0 < \gamma < 1$) and has high partition scores (low priority). We keep excluding outliers for at most I rounds, so that $\mathcal{M}^t$ will not be partitioned if its main components are outliers (i.e., $i \leq I$ does not hold).

4 LOCAL-VIEW INTERPRETATION

In the previous section, we discussed how to obtain the backbone of taxonomy through hierarchical clustering. In this section, we concentrate on each cluster in the taxonomy to summarize its unique characteristics or properties.

Specifically, we utilize node attributes to delineate the properties of clusters. Typical examples of node attributes include the user profile information in social networks, reviews of products in e-commercial networks, and research areas of scholars in academic networks. The reasons are twofold for using node attributes to delineate the properties of clusters in the embedded space. First, many network embedding algorithms already incorporate attribute information to learn informative embedding representations [25, 26, 29, 56, 58]. In this case, our goal is to find the significant contents that lead to the embedding results. Second, according to the principle of network homophily [42], nodes with similar attributes are more likely to be linked with each other. Hence, even though an embedding algorithm does not explicitly leverage node attribute information, we can still use it to derive a palpable understanding of the obtained clusters.

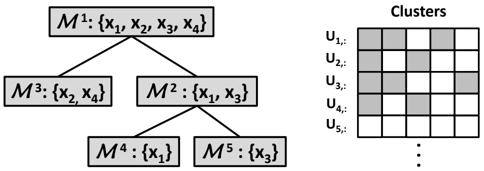

$\mathcal{M}^1$: {x$_1$, x$_2$, x$_3$, x$_4$}
$\mathcal{M}^3$: {x$_2$, x$_4$}
$\mathcal{M}^2$: {x$_1$, x$_3$}
$\mathcal{M}^4$: {x$_1$}
$\mathcal{M}^5$: {x$_3$}
Clusters
U$_{1,:}$
U$_{2,:}$
U$_{3,:}$
U$_{4,:}$
U$_{5,:}$
⋮

Figure 3: Left: Groups of important attributes associated with each cluster in a subtree of the taxonomy. Right: Corresponding activated entries in the weight matrix U.

We tackle the problem of characterizing clusters as a multitask feature selection problem. Here the "feature" refers to the node attributes. Each cluster is then characterized by the important attributes shared by the majority of nodes in the cluster. The cluster structure identified by the global-view interpretation provides the discriminative information which supports feature selection to be implemented in a supervised manner. Concretely, let $\mathbf{Y} \in \mathbb{R}^{N\times C}$ denote the class label matrix, where $\mathbf{Y}_{n,t} = 1$ if node n belongs to cluster $\mathcal{M}^t$ and $\mathbf{Y}_{n,t} = 0$ otherwise. Here C is the number of tasks. If we want to describe all clusters in the taxonomy, then C equals to the total number of internal nodes and leaf nodes in the tree $\mathcal{T}$. Let $\mathbf{U} \in \mathbb{R}^{M\times C}$ denote the attribute weight matrix, where $\mathbf{U}_{m,t}$ indicates the importance of attribute m in cluster $\mathcal{M}^t$. For a pair

Algorithm 2: Optimization algorithm for Equation 11.

```
  Input: Affinity matrix $\mathbf{A}$ of $G$, attribute matrix $\mathbf{X}$, hierarchical
         clusters $\mathcal{T}$, $\lambda \in \mathbb{R}_{+}$.
  Output: Weight matrix $\mathbf{U}$, coefficients $\{g_{m,\tau}\}$.
1 Initialize $\mathbf{U}$ and $\{g_{m,\tau}\}$
2 while not converged do
3     for $t = 1, 2, \ldots, C$ do
4         Collect samples $\{(i,j) | i, j \in \mathcal{M}^t\}$, and negative samples
            $\{(i^-, j^-)\}$ from other clusters
5         Update $\mathbf{U}_{:,t}$ using mini-batch stochastic gradient descent
6     Update $g_{m,\tau} = \frac{l_\tau \|\mathbf{U}_m^\tau\|_2}{\sum_m \sum_{\tau \in \mathcal{T}} l_\tau \|\mathbf{U}_m^\tau\|_2}$
```

of nodes i, j within a certain cluster $\mathcal{M}^t$, the consistency of node similarity in the embedding space and that in the selected feature space can be quantified as below:

$$\sigma(i,j) = \frac{1}{1 + \exp(-\mathbf{A}_{i,j} \mathbf{x}_i{}^T diag(\mathbf{U}_{:,t}) \mathbf{x}_j)}, \tag{8}$$

where $diag(\mathbf{U}_{:,t})$ turns the column vector $\mathbf{U}_{:,t}$ into the diagonal matrix form. $\mathbf{x}_i{}^T diag(\mathbf{U}_{:,t})\mathbf{x}_j$ is the similarity between attributes of i and j weighted by $\mathbf{U}_{:,t}$, while $\mathbf{A}_{i,j}$ is the edge strength between instance i and j in graph G. Then the objective function with respect to $\mathbf{U}$ is given as follows:

$$\max_{\mathbf{U}} \sum_{t=1}^{C} \Big(\sum_{i,j \in \mathcal{M}^t} \log(\sigma(i,j)) + \sum_{i^-,j^-} \log(1 - \sigma(i^-, j^-)) \Big). \tag{9}$$

The samples i^- and j^- are negative samples from other clusters. By optimizing the above objective function, we obtain $\mathbf{U}$ that selects a subset of important attributes for each cluster $\mathcal{M}^t$ such that node proximity in the selected feature space are in line with the proximity in the embedding space.

To take advantage of the hierarchical structures contained in the taxonomy, some constraints are required to regularize $\mathbf{U}_{:,t}$. Let $\mathcal{P} = (t_0, t_1, t_2, ..., t_h)$ denote the path from the root to a leaf in the taxonomy tree $\mathcal{T}$, where t_i refers to a tree node and t_{i-1} is the parent of t_i. Two clusters $\mathcal{M}^{t_i}$ and $\mathcal{M}^{t_j}$ are expected to share some common characteristics if t_i and t_j lie on the same path $\mathcal{P}$. An example is shown in Figure 3. For $\mathcal{M}^5$, its important attribute is x_3, while x_3 is also one of the important attributes of $\mathcal{M}^2$ and $\mathcal{M}^1$ as they are in the same root-to-leaf path. To incorporate the prior knowledge of tree-based structure in taxonomy, we introduce tree-based group lasso regularization [28] to the objective function. Therefore, the overall objective function for local-view interpretation is formulated as:

$$\begin{aligned} \max_{\mathbf{U}} \sum_{t=1}^{C} \Big(\sum_{i,j \in \mathcal{M}^t} \log(\sigma(i,j)) + \sum_{i^-,j^-} \log(1 - \sigma(i^-, j^-)) \Big) \\ - \lambda \sum_{m} \sum_{\tau \in \mathcal{T}} l_\tau \|\mathbf{U}_m^\tau\|_2. \end{aligned} \tag{10}$$

For the regularization term, $\mathbf{U}_m^\tau$ is a vector of weight coefficients $\{\mathbf{U}_{m,t} : t \in \tau\}$, where τ refers to a tree node in taxonomy $\mathcal{T}$. Here $t \in \tau$ if the cluster $\mathcal{M}^t$ is part of $\mathcal{M}^\tau$. This also implies that if $t \in \tau$, then $t \in parent(\tau)$. Considering the tree structure, one way for selecting negative samples i^- and j^-, w.r.t. $i, j \in \mathcal{M}^t$, is to sample from clusters that are not in the same path as $\mathcal{M}^t$.

Dataset	N	M	$\lvert\mathcal{E}\rvert$	$\#class$
BlogCatalog	5,196	50	343,486	6
Flickr	7,575	60	479,476	9
20NewsGroups	18,774	60	401,108	[6, 20]

Table 1: Statistics of the datasets.

The above optimization problem is non-smooth due to the existence of L_1/L_2-norm regularization term and is hence difficult to optimize. Therefore, we use an alternative formulation previously introduced for group lasso [2]:

$$\begin{aligned} \max_{\mathbf{U}} \sum_{t=1}^{C} \Big(\sum_{i,j \in \mathcal{M}^t} \log(\sigma(i,j)) + \sum_{i^-,j^-} \log(1 - \sigma(i^-, j^-)) \Big) \\ - \lambda \sum_{m} \sum_{\tau \in \mathcal{T}} \frac{l_\tau^2 \|\mathbf{U}_m^\tau\|_2^2}{g_{m,\tau}} \\ \text{subject to } \sum_{m} \sum_{\tau \in \mathcal{T}} g_{m,\tau} = 1, \; g_{m,\tau} \geq 0, \end{aligned} \tag{11}$$

where additional variables $\{g_{m,\tau}\}$ are introduced. The way of setting l_τ is suggested in [28]. The problem above is then solved by alternatively optimizing $\mathbf{U}$ and $g_{m,\tau}$, as shown in Algorithm 2. In each iteration, we first hold coefficients $g_{m,\tau}$ as constant and update each column $\mathbf{U}_{:,t}$ using stochastic gradient descent. Then we fix $\mathbf{U}$ and update $g_{m,\tau}$, where the update equation is given in Line 6 of Algorithm 2.

5 EXPERIMENTS

We evaluate the effectiveness of the proposed interpretation framework quantitatively on real-world datasets. A case study is also implemented to intuitively show the interpretation results.

5.1 Experimental Settings

Datasets We use three real-world datasets to evaluate the interpretation results. The detailed statistics of theses datasets are in Table 1. The detailed descriptions of these datasets are as follows.

- **BlogCatalog**[1]: An online community network from which links between users and posts of users are extracted. Users are represented as nodes, and their associated posts are used as node attributes. Predefined class labels of users are also available.
- **Flickr**[1]: An image and video hosting website. The following relationships among users form a network, in which the tags of interest are used as node attributes. The groups that users joined are treated as class labels.
- **20NewsGroups**[2]: A collection of news documents, each of which belongs to one of the twenty different topics. The topics are used as class labels, and a topic hierarchy is directly available from the data source. We use a tf-idf vector to represent each document and measure the similarities among documents using cosine similarity. A network is constructed based on document similarities, and news texts are attached as attributes. The attributes are only used in interpretation, while for network embedding only link information is considered.

[1] http://people.tamu.edu/~xhuang/Code.html

[2] http://qwone.com/~jason/20Newsgroups/

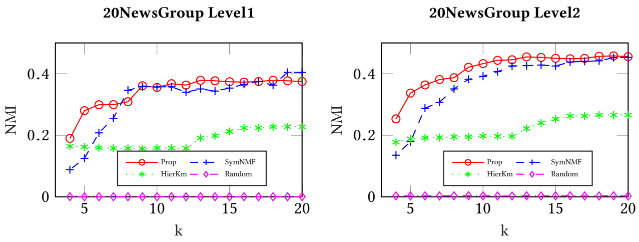


Figure 4: Hierarchical Clustering for LINE on 20NewsGroup.

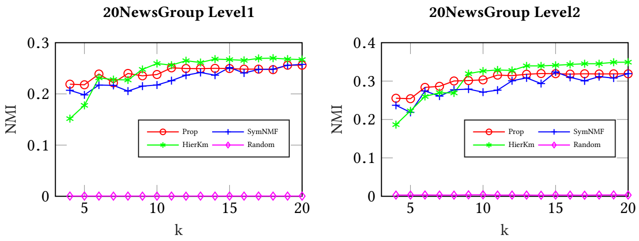


Figure 5: Hierarchical Clustering for node2vec on 20NewsGroup.

For all of the datasets above, we preprocess the attributes (i.e., word tokens) using LDA [7] to reduce the number of dimensions.

Alternative Methods for Global-View Interpretation. To evaluate whether the extracted hierarchical clusters by our approach are reasonable, we introduce two alternative clustering methods for comparative analysis. The goal is not to defeat these alternatives, since the proposed method can be seen as a variant of them.

- **SymNMF** [31]: A flat graph partitioning algorithm based on NMF. It has the same overall loss function as our method. The input is also the graph G constructed from embedding results.
- **HierKm** [30]: A hierarchical clustering method based on the k-means algorithm. Its workflow is the same as Algorithm 1. It operates directly on embedding instances.

Baseline Methods. We further verify if the description of each cluster, selected from node attributes, correctly reflects its characteristics. The baseline methods are introduced as below.

- **MTGL** [28]: A multitask classification model also guided by tree-based group lasso. It shares the same global-view interpretation result as the proposed method.
- **NDFS** [35]: An unsupervised feature selection algorithm. We use the graph Laplacian built from embedding vectors and the attribute information of nodes as the input.
- **LIME** [47]: A model originally designed for interpreting individual instances in classifiers. In our experiments, we select a number of target embedding instances. For each instance, we select important attributes of its neighborhood compared with some other distant clusters for local interpretation.

5.2 Evaluation on Global-View Interpretation

5.2.1 Evaluation Metric for Global-View Interpretation. The evaluation of global-view interpretation follows the common workflow of clustering evaluation. In particular, we use Normalized Mutual Information (NMI) [16] to measure the hierarchical clustering quality with respect to the ground-truth clusters. For a dataset with defined hierarchy, at each level, we compute the NMI between the clustering results and the ground-truth clusters at that level.

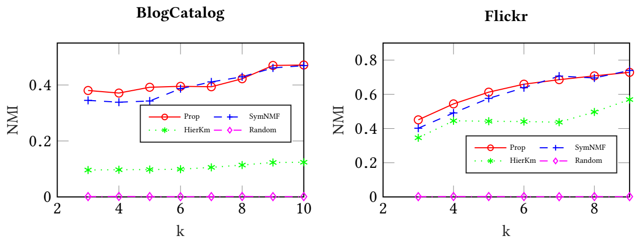


Figure 6: Global Interpretation for LANE on BlogCatalog and Flickr.

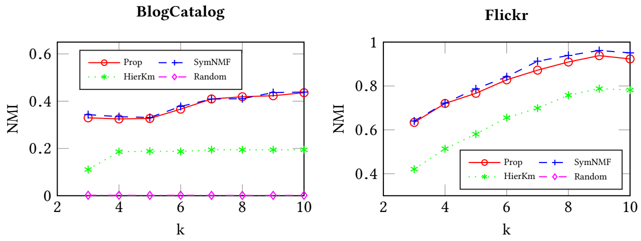


Figure 7: Global Interpretation for LCMF on BlogCatalog and Flickr.

5.2.2 Employed Network Embedding Methods. We introduce the employed network mebedding methods to be interpreted. Note that global-view interpretation takes the network embedding results as input rather than the original network. Therefore, to guarantee that ground-truth cluster labels of nodes can still be used on the embedding representations of nodes, we include the following embedding methods which: (1) preserve the first-order or high-order node proximity (e.g., LINE [52], SDNE [55], node2vec [22]); (2) incorporate labels with links and attributes (e.g., LCMF [58], LANE [26]). In particular, LINE, node2vec and SDNE are used to embed the 20NewsGroup network, while LANE and LCMF are performed on BlogCatalog and Flickr networks. For LINE, we set the number of negative samples as 10, the number of samples as 200 millions, $\rho = 0.025$, and enable second-order proximity. For node2vec, we set $p = 1$, $q = 0.5$, walk length as 80 and the number of walk per node as 15. For SDNE, there are two encoding layers with 600 and 128 latent factors, respectively. For LANE, we set the balancing parameter of labels as 300 for BlogCatalog and 150 for Flickr. For LCMF, we set the balancing parameter of attributes as 3 for BlogCatalog data and 5 for Flickr data. The notations above are borrowed from the corresponding reference papers.

5.2.3 Global-View Interpretation Results and Analysis. The global-view interpretation results are shown in Figures 4~7. We omit the results for SDNE on 20NewsGroup owing to lack of space, as they are similar to those for node2vec. Some observations are made as follows. First, in general, the proposed method and SymNMF have more stable clustering performances than HierKm, and even better performances in some cases. It means that, although we work on the embedding-based graph instead of directly on the embedding instances, the clustering results are not significantly affected. In certain cases, we even benefit from clustering on embedding-based graphs. Second, the proposed method is comparable to SymNMF as k increases, as they actually share the same original loss function in Equation 4. The proposed method has better performance when k is small. This is probably because a small k is largely inconsistent with the real number of clusters in the datasets, which hinders the

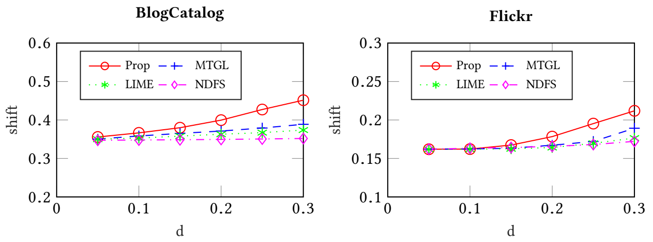


Figure 8: Local Interpretation for LANE on BlogCatalog and Flickr.

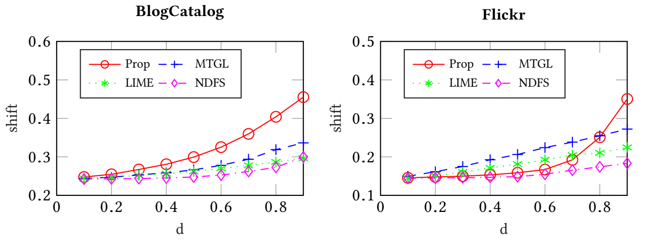


Figure 9: Local Interpretation for LCMF on BlogCatalog and Flickr.

performance of flat clustering. In this case, hierarchically clustering can more effectively discover the internal structures of the data. In conclusion, the observations above validate the soundness of the developed global-view interpretation method.

5.3 Evaluation on Local-View Interpretation

The goal of local-view interpretation is to identify the attributes shared by the nodes in the same cluster obtained from the global-view interpretation. We design two sets of experiments to verify the correctness of the identified attributes, through adversarial perturbation and network reconstruction, respectively.

5.3.1 Adversarial Perturbation Based on Interpretation. As it is difficult to obtain the ground truth information (e.g., significant attributes), we first evaluate the accuracy of generated interpretation through *adversarial perturbation*. After identifying the significant attributes of nodes, we select a number of seed nodes for generating adversarial samples. We create a copy for each of the seeds, distort the value of their attributes and fed the new nodes along with the original network into the embedding algorithms. After that, we measure the relative *shift* of the new embeddings from the original embeddings of seeds. Let $\mathbf{z}_n$ and $\mathbf{z}'_n$ be the embedding instance of node n before and after perturbation, respectively, the shift is defined as:

$$shift(n, d) = \frac{|\text{Neighbors}(\mathbf{z}_n) \cap \text{Neighbors}(\mathbf{z}'_n)|}{|\text{Neighbors}(\mathbf{z}_n)|}, \qquad (12)$$

where Neighbors($\mathbf{z}_n$) denotes the set of neighbor nodes around n in the embedding space, and d is the amplitude of adversarial perturbation. Large shift is expected if the generated interpretation faithfully reflects the mechanism of employed embedding models.

Specifically, the number of seeds is set as $0.03 \times N$ for each network, where N is the total number of nodes. For each seed's embedding vector, the number of neighbors is chosen as $0.01 \times N$ for LANE and $0.03 \times N$ for LCMF. The parameters settings of LANE and LCMF remain the same as in the previous experiments. For LCMF, since it outputs several matrices besides the embedding matrix $\mathbf{Z}$, we fix the values of these matrices and only allow $\mathbf{Z}$ to be updated during learning process. For each seed node in cluster $\mathcal{M}^t$, we identify 4 positively significant attributes and 4 negatively significant ones to be perturbed, corresponding to the largest and smallest entries in $\mathbf{U}_{:,t}$, respectively. The value of the former ones are decreased, while the value of latter ones are increased. The L_2 norms of both types of perturbation are normalized to d.

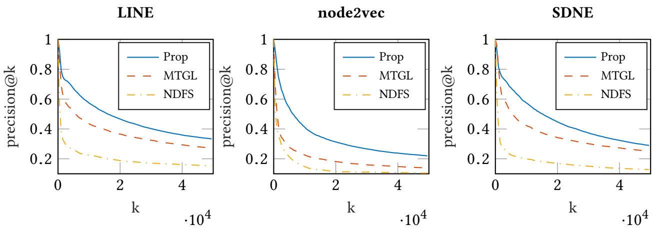


Figure 10: Network Reconstruction on 20NewsGroup Dataset.

The experiments are conducted on BlogCatalog and Flickr. The performance of adversarial perturbation is shown in Figure 8 for LANE and Figure 9 for LCMF. The figures show that the average shift of adversarial samples increases as we increase the perturbation amplitude. The adversarial samples created by the proposed method are more effective, as they are more dramatically shifted from their original neighborhood prior to adversarial perturbation. The proposed method has better performance than MTGL. The reason could be that the edge weights contained in $\mathbf{A}$ are more informative than binary class labels used in MTGL. In addition, interpretation approaches (e.g., Prop, MGTL), which focus on overall embedding instances, have better performances than LIME which interprets individual embeddings locally. It indicates the importance of considering a wider range of contexts in solving our problem. The embedding instances in other clusters provide contrastive information to filter out irrelevant attributes with respect to the current cluster, while instances within the same cluster help filter out noises in attribute selection.

5.3.2 Network Reconstruction from Interpretation. In this subsection, we evaluate interpretation results with respect to their capability of network reconstruction. The motivation for this experiment is that, while network embedding methods encode topological structures into embeddings, interpretation algorithms try to recover the attribute patterns consistent with the embedding distribution. We use the 20NewsGroup network, in which the links between documents (nodes) are established based on their content similarity. We build an LDA model with 60 latent topics from the documents, and assign each node with a 60-dimensional attribute vector. After solving $\mathbf{U}$, in each leaf cluster $\mathcal{M}^t$, we predict links based on the similarities between nodes with respect to attributes, the importance of which is weighted by $\mathbf{U}_{:,t}$. The predicted links correspond to node pairs with large similarity scores. The existing links in the original network serve as the ground-truth. LIME is not considered in this experiment since it interprets embedding vectors locally. We use *precision@k* as the evaluation metric.

The results are presented in Figure 10. In general, the proposed method achieves better performance than baseline methods. Its interpretation performances are relatively stable across different network embedding methods. The edge weights in $\mathbf{A}$ contain more information than class labels, which could explain why the proposed method performs better than MTGL. The advantages of NDFS are

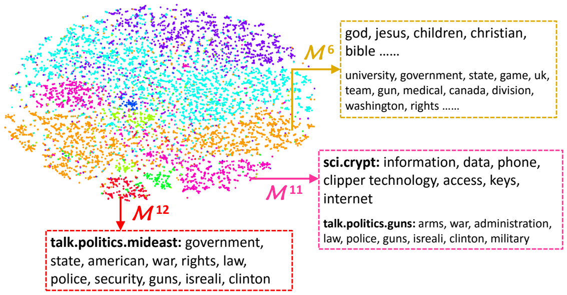


Figure 11: The inducted taxonomy with 7 leaf clusters.

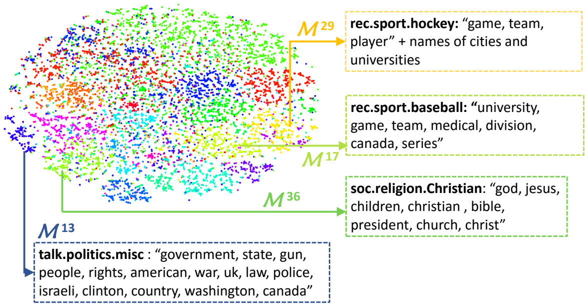


Figure 12: The inducted taxonomy with 20 leaf clusters.

not reflected via this experiment. The reason is that the attributes information are already modeled as links and fed into network embedding methods, so jointly considering embeddings and attributes does not bring additional benefits.

5.4 Case Study

We conduct a case study on 20NewsGroups network using LINE [52] to show the taxonomy induction result and the characteristics of extracted taxonomy concept (i.e., clusters). We present the inducted taxonomy with 7 leaf clusters and 20 leaf clusters in Figure 11 and Figure 12, denoted by $\mathcal{T}^{7}$ and $\mathcal{T}^{20}$, respectively. The two figures are generated from the same taxonomy, but with different depths of division. Some clusters in $\mathcal{T}^{20}$ are obtained by splitting the larger clusters in $\mathcal{T}^{7}$. Clusters are indicated with different colors. The visualization of embedding vectors are obtained using t-SNE [41]. We select several clusters and present the keywords of the documents in each cluster in the dashed boxes. For each cluster $\mathcal{M}^{t}$, we first find its significant attributes m corresponding to the large positive entries $\mathbf{U}_{m,t}$, and then identify the keywords associated with each attribute m. As introduced earlier, here the "attributes" are the latent factors extracted by LDA. Keywords of larger font size in boxes are ranked higher than those of smaller font size. Keywords that belong to the stop words or artifacts in email streams (e.g., com, edu) are not shown. The bold text prior to the keywords, if available, represent the manually-identified news topic (named by the dataset provider) which the majority of the nodes in the cluster belong to. There are 20 such topics in the dataset.

We make the following observations from Figure 11 and Figure 12. First, nodes in each cluster extracted by our interpretation method locate closely to each other. In general, the cluster structures are consistent with the visualization results, which reflects the effectiveness of the global-view interpretation method. Second, the keywords of each cluster are consistent with the ground truth news topics, which validates the effectiveness of the local-view interpretation. Third, we can observe that some major clusters, such as $\mathcal{M}^{6}$ and $\mathcal{M}^{11}$, contain multiple topics, since they have not been throughly decomposed in shallow levels of the taxonomy. Different topics are disentangled as we continue splitting clusters towards deeper levels. For example, many topics are mixed in $\mathcal{M}^{6}$ which is one of the major clusters in $\mathcal{T}^{7}$. However, as we split $\mathcal{M}^{6}$ into $\mathcal{M}^{13}$, $\mathcal{M}^{17}$, $\mathcal{M}^{29}$ and $\mathcal{M}^{36}$, each sub-cluster has a coherent topic. Some keywords in these sub-clusters are inherited from $\mathcal{M}^{6}$, such as the {god, jesus} in $\mathcal{M}^{36}$, {game, team} in $\mathcal{M}^{29}$ and {government, gun} in $\mathcal{M}^{13}$, so that a larger and coarser concept is split into smaller but refined ones.

6 RELATED WORK

Network embedding is attracting more and more attentions recently for its effectiveness in generating informative node representations. Network embedding methods can be divided into several categories according to the types of information that is preserved [17]. First, some approaches encode topological structures into embedding vectors, such as various orders of proximity [29, 46, 48, 52, 55], transitivity [45], community structures [56] and the structural roles of nodes [22]. Here first-order proximity usually refers to the directly links between nodes, and high-order proximity means node neighborhoods of various ranges. Second, some methods incorporate rich side information of networks, such as attributes of nodes and edges [25, 34, 58], as well as the labels of nodes [26]. Third, some existing work models different types of objects using networks with heterogeneous nodes and jointly learn embedding vectors [12, 19, 51]. Network embedding has been shown to be feasible in tacking problems such as link prediction [55], node classification [22, 52] and network alignment [14].

Despite superior performance and widespread application, many popular ML models such as deep models remain mostly black boxes to end users [13, 39, 47]. Existing methods for ML interpretation focus on explaining classification and prediction models. These methods mainly focus on: 1) explaining the working mechanisms or the learned concept after model establishment [13, 27, 43]; 2) extracting the important features or rules of how individual predictions are made [3, 15, 38, 47]. Specifically, for the former category, some methods select representative samples for each class to build the concept of different classes [27, 33], while some methods utilize model compression to extract or reformulate human-understandable knowledge from complex models [9, 13]. Visualization techniques can work with interpretation methods to facilitate the description of data and exploration of sense-making patterns in networks. The main idea is to project data into extremely low-dimensional (2D or 3D) spaces. Some representative techniques include Principal Component Analysis [1], nonlinear methods such as Isomap [53], Laplacian Eigenmaps [6], t-SNE [41] and LargeVis [50].

7 CONCLUSION AND FUTURE WORK

In this paper, we propose a novel post-hoc interpretation framework to understand network embedding. We formulate the problem as taxonomy induction from embedding results. We first build a graph

G from embeddings to encode the relations between embedding instances into graph edge weights. The backbone of the taxonomy is constructed as a tree by performing hierarchical clustering on the graph G. The characteristics of the clusters in taxonomy are identified as a multitask regression problem with tree-based regularization. Quantitative evaluations and case studies on real-world datasets demonstrate the effectiveness of the proposed framework.

Some future work towards interpreting network embedding are as follows. First, more sophisticated hierarchical clustering methods can be developed to extract more accurate structure hierarchies. Second, in addition to the node attributes, interpretation based on network structural characteristics can be further exploited. Third, combining embedding interpretation with existing ML interpretation is also a promising direction.

ACKNOWLEDGMENTS

The work is, in part, supported by DARPA (#N66001-17-2-4031) and NSF (#IIS-1657196, #IIS-1718840). The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing any funding agencies.

REFERENCES

[1] Hervé Abdi and Lynne J Williams. 2010. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics* (2010).

[2] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. 2008. Convex multi-task feature learning. *Machine Learning* (2008).

[3] David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert MÃžller. 2010. How to explain individual classification decisions. *Journal of Machine Learning Research* (2010).

[4] Oren Barkan and Noam Koenigstein. 2016. Item2vec: neural item embedding for collaborative filtering. In *MLSP Workshop*.

[5] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. 2017. Network dissection: Quantifying interpretability of deep visual representations. In *CVPR*. IEEE.

[6] Mikhail Belkin and Partha Niyogi. 2002. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *NIPS*.

[7] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *the Journal of Machine Learning Research* 3 (2003), 993–1022.

[8] Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. 2013. Recommender systems survey. *Knowledge-based systems* (2013).

[9] Cristian Bucilua, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *KDD*.

[10] Paul Buitelaar, Philipp Cimiano, and Bernardo Magnini. 2005. Ontology learning from text: An overview. *Ontology learning from text: Methods, evaluation and applications* (2005).

[11] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. Grarep: Learning graph representations with global structural information. In *CIKM*. ACM.

[12] Shiyu Chang, Wei Han, Jiliang Tang, Guo-Jun Qi, Charu C Aggarwal, and Thomas S Huang. 2015. Heterogeneous network embedding via deep architectures. In *KDD*. ACM.

[13] Zhengping Che, Sanjay Purushotham, Robinder Khemani, and Yan Liu. 2015. Distilling knowledge from deep networks with applications to healthcare domain. *arXiv preprint arXiv:1512.03542* (2015).

[14] Ting Chen and Yizhou Sun. 2017. Task-Guided and Path-Augmented Heterogeneous Network Embedding for Author Identification. In *WSDM*. ACM.

[15] Edward Choi, Mohammad Taha Bahadori, Jimeng Sun, Joshua Kulas, Andy Schuetz, and Walter Stewart. 2016. Retain: An interpretable predictive model for healthcare using reverse time attention mechanism. In *NIPS*.

[16] D Manning Christopher, Raghavan Prabhakar, and SCHÜTZE Hinrich. 2008. Introduction to information retrieval. (2008).

[17] Peng Cui, Xiao Wang, Jian Pei, and Wenwu Zhu. 2017. A Survey on Network Embedding. *arXiv preprint arXiv:1711.08752* (2017).

[18] Chris Ding, Xiaofeng He, and Horst D Simon. 2005. On the equivalence of nonnegative matrix factorization and spectral clustering. In *SDM*. SIAM.

[19] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *KDD*.

[20] Mengnan Du, Ninghao Liu, Qingquan Song, and Xia Hu. 2018. Towards Explanation of DNN-based Prediction with Guided Feature Inversion. In *KDD*.

[21] Linton C Freeman. 2000. Visualizing social networks. *JoSS* (2000).

[22] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *KDD*.

[23] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*.

[24] Cho-Jui Hsieh and Inderjit S Dhillon. 2011. Fast coordinate descent methods with variable selection for non-negative matrix factorization. In *KDD*. ACM.

[25] Xiao Huang, Jundong Li, and Xia Hu. 2017. Accelerated attributed network embedding. In *SDM*. SIAM.

[26] Xiao Huang, Jundong Li, and Xia Hu. 2017. Label informed attributed network embedding. In *WSDM*. ACM.

[27] Been Kim, Cynthia Rudin, and Julie A Shah. 2014. The bayesian case model: A generative approach for case-based reasoning and prototype classification. In *NIPS*.

[28] Seyoung Kim and Eric P Xing. 2010. Tree-guided group lasso for multi-task regression with structured sparsity. In *ICML*.

[29] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[30] Da Kuang and Haesun Park. 2013. Fast rank-2 nonnegative matrix factorization for hierarchical document clustering. In *KDD*. ACM.

[31] Da Kuang, Sangwoon Yun, and Haesun Park. 2015. SymNMF: nonnegative low-rank approximation of a similarity matrix for graph clustering. *Journal of Global Optimization* (2015).

[32] Timothy La Fond and Jennifer Neville. 2010. Randomization tests for distinguishing social influence and homophily effects. In *WWW*.

[33] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. 2015. Human-level concept learning through probabilistic program induction. *Science* (2015).

[34] Jundong Li, Harsh Dani, Xia Hu, Jiliang Tang, Yi Chang, and Huan Liu. 2017. Attributed Network Embedding for Learning in a Dynamic Environment. *CIKM*.

[35] Zechao Li, Yi Yang, Jing Liu, Xiaofang Zhou, Hanqing Lu, et al. 2012. Unsupervised feature selection using nonnegative spectral analysis.. In *AAAI*.

[36] Chih-Jen Lin. 2007. Projected gradient methods for nonnegative matrix factorization. *Neural computation* (2007).

[37] Zachary C Lipton. 2016. The mythos of model interpretability. *arXiv preprint arXiv:1606.03490* (2016).

[38] Ninghao Liu, Donghwa Shin, and Xia Hu. 2017. Contextual Outlier Interpretation. *arXiv preprint arXiv:1711.10589* (2017).

[39] Ninghao Liu, Hongxia Yang, and Xia Hu. 2018. Adversarial Detection with Model Interpretation. In *KDD*.

[40] Xueqing Liu, Yangqiu Song, Shixia Liu, and Haixun Wang. 2012. Automatic taxonomy construction from keywords. In *KDD*.

[41] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* (2008).

[42] Miller McPherson, Lynn Smith-Lovin, and James M Cook. 2001. Birds of a feather: Homophily in social networks. *Annual review of sociology* (2001).

[43] Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. 2017. Methods for Interpreting and Understanding Deep Neural Networks. *arXiv preprint arXiv:1706.07979* (2017).

[44] Roberto Navigli, Paola Velardi, and Stefano Faralli. 2011. A graph-based algorithm for inducing lexical taxonomies from scratch. In *IJCAI*.

[45] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. 2016. Asymmetric Transitivity Preserving Graph Embedding. In *KDD*.

[46] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *KDD*. ACM.

[47] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why Should I Trust You?: Explaining the Predictions of Any Classifier. In *KDD*.

[48] Blake Shaw and Tony Jebara. 2009. Structure preserving embedding. In *ICML*.

[49] Jianbo Shi and Jitendra Malik. 2000. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence* (2000).

[50] Jian Tang, Jingzhou Liu, Ming Zhang, and Qiaozhu Mei. 2016. Visualizing large-scale and high-dimensional data. In *WWW*.

[51] Jian Tang, Meng Qu, and Qiaozhu Mei. 2015. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *KDD*. ACM.

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

[53] Joshua B Tenenbaum, Vin De Silva, and John C Langford. 2000. A global geometric framework for nonlinear dimensionality reduction. *science* (2000).

[54] Ulrike Von Luxburg. 2007. A tutorial on spectral clustering. *Statistics and computing* (2007).

[55] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *KDD*. ACM.

[56] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. 2017. Community Preserving Network Embedding.. In *AAAI*.

[57] Junyuan Xie, Ross Girshick, and Ali Farhadi. 2016. Unsupervised deep embedding for clustering analysis. In *ICML*.

[58] Shenghuo Zhu, Kai Yu, Yun Chi, and Yihong Gong. 2007. Combining content and link for classification using matrix factorization. In *SIGIR*.