role2vec: Role-based Network Embeddings

Nesreen K. Ahmed
Intel Labs

Ryan A. Rossi
Adobe Research

John Boaz Lee
WPI

Theodore L. Willke
Intel Labs

Rong Zhou
Google

Xiangnan Kong
WPI

Hoda Eldardiry
Xerox PARC

ABSTRACT

Random walks are at the heart of many existing network embedding methods. However, such methods have many limitations that arise from the use of traditional random walks, *e.g.*, the embeddings resulting from these methods primarily capture proximity (communities) among the vertices as opposed to structural equivalence/similarity (roles). In this work, we introduce the *Role2Vec* framework which uses the flexible notion of *attributed random walks*, and serves as a basis for generalizing existing methods such as DeepWalk, node2vec, and many others that leverage random walks. Our proposed framework enables these methods to be more widely applicable by learning functions that capture the behavioral roles of the nodes. We show that our proposed framework is effective with an average AUC improvement of 16.55% while requiring on average 853x less space than existing methods.

KEYWORDS

Role discovery, roles/positions, structural similarity, proximity, node embedding, random walk, graph clustering, communities, feature-based walks

1 INTRODUCTION

Learning a useful feature representation from graph data lies at the heart and success of many machine learning tasks such as node classification [39], anomaly detection [3], link prediction [4], among others [29, 40]. Motivated by the success of word embedding models, such as the skip-gram model [37, 38], recent works extended word embedding models to learn graph embeddings [20, 42]. The primary goal of these works is to model the conditional probabilities that relate each input vertex to its context, where the context is a set of other vertices surrounding and/or topologically related to the input vertex. Many variants of graph embedding methods proposed *random walks* to generate the context vertices [9, 22, 42, 44]. For instance, DeepWalk [42] initiates random walks from each vertex to collect sequences of vertices (similar to sentences in language). Then, the skip-gram model is used to fit the embeddings by maximizing the conditional probabilities that relate each input vertex to its surrounding context. Thus, vertex identities are used as words in the skip-gram model, and the embeddings are tied to vertex ids.

In language, the central idea is that words with similar meanings will be surrounded by a similar context [24]. As such, in language models, the context of a word is defined as the surrounding words. However, this foundation does not directly translate to graphs. Since unlike words in languages that are universal with semantics and meaning independent of the corpus of documents, vertex ids obtained by random walks on graphs are not universal and are only meaningful within a particular graph. This key limitation has two main disadvantages. First, these embedding methods are inherently transductive, dealing essentially with isolated graphs, and unable to generalize to unseen nodes. Consequently, they are unsuitable for graph-based transfer learning tasks such as across-network classification [16, 30], and graph similarity/comparison [19, 55]. Second, by using this traditional definition of random walks, there is no general way to integrate vertex attributes/features to the network representation.

While similar words are typically surrounded by a similar context, there is no guarantee that similar vertices are surrounded by similar context (obtained using random walks on graphs). Recent empirical analysis shows that using random walks in graph embeddings primarily capture proximity among the vertices (see [20]), so that vertices that are close to one another in the graph are embedded together, *e.g.*, vertices that belong to the same community are embedded similarly. While proximity among the vertices does not guarantee their similarity, the concept of a network position or a *role* [25, 34, 46] is more suitable to represent the equivalence/similarity and structural relatedness among vertices.

Roles represent vertex connectivity patterns such as hubs, star-centers, star-edge nodes, near-cliques or vertices that act as bridges to different regions of the graph. Intuitively, two vertices belong to the same role if they are structurally similar/equivalent and need not be connected by an edge. Several methods propose properties under which vertices in a network are to be considered equivalent/similar [52]. However, all of these methods are based on the concept of a network position (*role*) as a collection of vertices (*e.g.*, actors in a social network) all of whom are similarly connected to other vertices or sets of vertices [52]. Since random walks will likely visit nearby vertices first, they are more suitable for finding communities (proximity/adjacency), rather than roles (structural similarity/equivalence, see [46] for a survey on roles).

In this work, we propose the *Role2Vec* framework which serves as a basis for generalizing many existing methods that use traditional random walks. In particular, we introduce the notion of attributed random walks that is not tied to vertex identity and is instead based on a function $\Phi : \mathbf{x} \rightarrow w$ that maps a vertex attribute vector to a role (label, attribute value), such that two vertices belong to the same role if they are structurally similar. Then *Role2Vec* learns the embeddings of vertex roles (as opposed to individual vertices). Thus, *Role2Vec* learns a representation that model the associations among subsets of vertices (*i.e.*, roles). The proposed framework provides

a number of important advantages to any method generalized using it. First, the proposed framework is naturally inductive as the learned features generalize to new nodes and across graphs and therefore can be used for transfer learning tasks. Second, they are able to capture structural similarity (roles) better. Third, the proposed framework is inherently space-efficient since embeddings are learned for roles (as opposed to vertices) and therefore requires significantly less space than existing methods. Fourth, the proposed framework naturally supports graphs with attributes (if available/-given as input). Furthermore, our approach is shown to be effective with an average improvement of 16.55% in AUC while requiring on average 853x less space than existing methods on a variety of graphs from different domains.

2 FRAMEWORK

We consider an (un)directed input graph $G = (V, E)$, where $N_v = |V|$ is the number of vertices in G, and $N_e = |E|$ is the number of edges in G. For any vertex $v_i \in V$, let $\Gamma(i)$ be the set of direct neighbors of v_i, and $d_i = |\Gamma(i)|$ is the vertex degree. In addition, we consider a matrix $\mathbf{X}$ of attributes/features, where each $\mathbf{x}_i$ is a K-vector for vertex v_i. For example, for graphs without attributes, $\mathbf{x}_i$ could simply be an indicator vector for vertex v_i and K is equivalent to the number of vertices (*i.e.*, having $x_{ij} = 1$ if $j = i$, and $x_{ij} = 0$ otherwise) [22, 42]. For attributed graphs, $\mathbf{x}_i$ may include observed attributes, topological features, and/or node types for heterogeneous graphs. The goal of an embedding method is to derive useful features of particular graph elements (*e.g.*, vertices, edges) by learning a model that maps each graph element to the latent D-dimension space. While the approach remains general for any graph element, this paper focuses on vertex embeddings.

To achieve this, an embedding is usually defined with three components: (1) the context function, which specifies a set of other vertices called the *context* c_i for any given vertex v_i, such that the context vertices are surrounding and/or topologically related to the given vertex. Each vertex is associated with two latent vectors, an *embedding* vector $\boldsymbol{\alpha}_i \in \mathbb{R}^D$ and a *context* vector $\boldsymbol{\beta}_i \in \mathbb{R}^D$. (2) the conditional distribution, which specifies the statistical distribution used to combine the embedding and context vectors. More specifically, the conditional distribution of a vertex combines its embedding and the context vectors of its surrounding vertices. (3) the model parameters (*i.e.*, embedding and context vectors) and how these are shared across the conditional distributions. Thus, an embedding method models the conditional probability that relate each vertex to its context as follows:

$$\mathbf{x}_{c_i} \mid \mathbf{x}_i \sim \mathbb{P} \tag{1}$$

where c_i is the set of context vertices for vertex v_i, $\mathbf{x}_i$ is its feature/attribute vector, and $\mathbb{P}$ is the conditional distribution.

Our goal is to model $\mathbb{P}[\mathbf{x}_{c_i}|\mathbf{x}_i] = \prod_{j \in c_i} \mathbb{P}(\mathbf{x}_j|\mathbf{x}_i)$, assuming the context vertices are conditionally independent. The most commonly used conditional distribution is the categorical distribution (see [48] for other distributions). In this case, a softmax function parameterized with the two latent vectors (*i.e.*, embedding and context vectors) is used. Thus, for each input-context vertex pair (v_i, v_j),

$$\mathbb{P}(\mathbf{x}_j|\mathbf{x}_i) = \frac{\mathrm{e}^{\boldsymbol{\alpha}_i \cdot \boldsymbol{\beta}_j}}{\sum_{v_k \in V} \mathrm{e}^{\boldsymbol{\alpha}_i \cdot \boldsymbol{\beta}_k}} \tag{2}$$

For sparse graphs, the summation in $\sum_{v_k \in V} \mathrm{e}^{\boldsymbol{\alpha}_i \cdot \boldsymbol{\beta}_k}$ contains many zero entries, and thus can be approximated by sub-sampling those zero entries (using negative sampling similar to language models [38]). Finally, the objective function of the embedding method is the sum of the logarithm of likelihood values of each vertex, *i.e.*,

$$\mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \sum_{i=1}^{N_v} \log\ \mathbb{P}[\mathbf{x}_{c_i}|\mathbf{x}_i] \tag{3}$$

Clearly, there is a class of possible embedding methods where each of the three components (discussed above) is considered a modeling choice with various alternatives. Recent work proposed *random walks* to sample/collect the context vertices c_i [22, 42]. Note for these random walk based embedding methods, $\mathbf{x}_i$ is simply an indicator vector for vertex v_i (*i.e.*, no attributes).

2.1 Mapping Vertices to Vertex-Roles

Given $N_v \times K$ matrix $\mathbf{X}$ of attributes and/or structural features, the *Role2Vec* framework starts by locating sets of vertices, independent of the distance (proximity) between any two in the set, who are placed similarly with respect to all other sets of vertices. Thus, two vertices belong to the same set if they are similar in terms of attributes and/or structural features. We achieve this by learning a function that maps the N_v vertices to a set $W = \{w_1, ..., w_M\}$ of M *vertex-roles* where M is often much smaller than N_v, *i.e.*, $M \ll N_v$,

$$\Phi\ :\ \mathbf{x}\ \rightarrow\ w \tag{4}$$

Thus, Φ is a function mapping vertices to *vertex-roles* based on the $N_v \times K$ attribute matrix $\mathbf{X}$. Clearly, the function Φ is a modeling choice [52], which could be learned automatically or defined manually by the user. We explore two general classes of functions for mapping vertices to their roles. The first class of functions are simple functions taking the form:

$$\Phi(\mathbf{x}) = x_1 \circ x_2 \circ \cdots \circ x_K \tag{5}$$

where $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \cdots & x_K \end{bmatrix}$ is an attribute vector and $\circ$ is a binary operator such as concatenation, sum, among others. Notably, suppose 3 or 4-node graphlets are used, then two nodes belong to the same role iff they have the same graphlet features. This is consistent with the definition of structural roles, see [46] for further details. As an aside, the above functions can be "relaxed" by performing log binning on each individual graphlet feature prior to using Eq. 5. This has two main consequences. First, it allows nodes to be in the same role even if they do not have identical feature values. Thus, nodes that are sufficiently similar with respect to their structural properties belong to the same role. Second, log binning and other feature transformation techniques can be used to control the number of roles (*e.g.*, by changing the bin size in log binning, etc.).

The second class of functions are learned by solving an objective function. This includes functions based on a low-rank factorization of the $N_v \times K$ matrix $\mathbf{X}$ having the form $\mathbf{X} \approx f\langle \mathbf{U}\mathbf{V}^T \rangle$ with factor matrices $\mathbf{U} \in \mathbb{R}^{N_v \times r}$ and $\mathbf{V} \in \mathbb{R}^{K \times r}$ where $r > 0$ is the rank and f is a linear or non-linear function. More formally,

$$\arg \min_{\mathbf{U}, \mathbf{V} \in C} \left[\mathcal{D}\left(\mathbf{X}, f\langle \mathbf{U}\mathbf{V}^T \rangle\right) + \mathcal{R}(\mathbf{U}, \mathbf{V}) \right] \tag{6}$$

where $\mathcal{D}$ is the loss, C is constraints (*e.g.*, non-negativity constraints $\mathbf{U} \geq 0, \mathbf{V} \geq 0$), and $\mathcal{R}(\mathbf{U}, \mathbf{V})$ is a regularization penalty. Then, we partition $\mathbf{U} \in \mathbb{R}^{N_v \times r}$ into M disjoint sets of nodes (for each of the

M vertex-roles) $V_1, \ldots, V_M$, where V_j is the set of vertices mapped to vertex-role $w_j \in W$, by solving the k-means objective:

$$\min_{\{V_j\}_{j=1}^{M}} \sum_{j=1}^{M} \sum_{\mathbf{u}_i \in V_j} \|\mathbf{u}_i - \mathbf{c}_j\|^2, \text{where } \mathbf{c}_j = \frac{\sum_{\mathbf{u}_i \in V_j} \mathbf{u}_i}{|V_j|} \tag{7}$$

2.2 Attributed Random Walks

Recently, random walks received much attention in learning network embeddings [22, 42], in particular to generate the context vertices (as discussed above). Consider a random walk of length L and starting at a vertex v_0 of the input graph G, if at time t we are at vertex v_t, then at time $t+1$, we move to a neighbor of v_t with probability $1/d_{v_t}$. Thus, the resulting randomly chosen sequence of vertex indices $(v_t : t = 0, 1, ..., L-1)$ is a Markov chain. However, a key limitation of these methods is that the embeddings learned based on random walks are fundamentally tied to vertex ids (as discussed above). By using this traditional definition of random walks, there is no general way to integrate vertex attributes and structural features to the network representation. On the other hand, vertex attributes and structural features can easily be represented by differentiating the edges according to the roles of their endpoints, which leads to the definition of *attributed random walks*.

Definition 1 (Attributed walk). *Let $\mathbf{x}_i$ be a K-dimensional vector for vertex v_i. An* attributed walk *of length L is a sequence of adjacent vertex-roles,*

$$\Phi(\mathbf{x}_{v_0}), \ldots, \Phi(\mathbf{x}_{v_t}), \Phi(\mathbf{x}_{v_{t+1}}), \ldots, \Phi(\mathbf{x}_{v_{L-1}}) \tag{8}$$

induced by a randomly chosen sequence of indices $(v_t : t = 0, 1, ..., L-1)$ generated by a random walk of length L starting at v_0, and a function $\Phi : \mathbf{x} \to w$ that maps an input vector $\mathbf{x}$ to a vertex role $\Phi(\mathbf{x})$.

The induced vertex-role sequence in the above definition is called *attributed random walks*.

The *Role2vec* framework uses vertex mapping and attributed random walks to learn the embeddings. Thus, our goal is to model the conditional probability that relate each vertex-role to the roles of its context,

$$\mathbb{P}\Big[\Phi\langle \mathbf{x}_{c_i}\rangle | \Phi\langle \mathbf{x}_i\rangle\Big] = \prod_{j \in c_i} \mathbb{P}(\Phi\langle \mathbf{x}_j\rangle \,|\Phi\langle \mathbf{x}_i\rangle) \tag{9}$$

Hence, the embedding structure (*i.e.*, the embedding and context vectors) is shared among the vertices with the same vertex-role. Specifically, we learn $\boldsymbol{\alpha}_j \in \mathbb{R}^D$ and $\boldsymbol{\beta}_j \in \mathbb{R}^D$ for each partition V_j of vertices, which are mapped to vertex-role w_j. Note that *Role2vec* learns an embedding for an aggregated network, where detailed relations among individual vertices are aggregated to total relations among vertex-roles.

2.3 Role2Vec Algorithm

The *Role2Vec* algorithm is shown in Alg. 1. Alg. 1 takes the following inputs: (1) graph G, (2) attribute matrix $\mathbf{X}$, (3) embedding dimension D, (4) walks per vertex R, (5) walk length L, (6) context window size ω. In Line 3, if $\mathbf{X}$ is not available, we derive structural features using the graph structure itself. For instance, in this paper, we use small subgraphs called motifs as higher-order structural features. Counts of motif patterns were found useful to capture the local structure of vertices and can be computed quickly and efficiently with parallel

Algorithm 1 role2vec

```
1  procedure Role2Vec(G = (V, E) and X, embedding dimensions D, walks per
   node R, walk length L, context (window) size ω)
2      Initialize set of attributed walks S to ∅
3      Extract (motif) features if needed and append to X
4      Transform each attribute in X (e.g., using logarithmic binning)
5      Map vertices to roles function Φ : x → w
6      Precompute transition probabilities π
7      G' = (V, E, π)
8      parallel for j = 1, 2, ..., R do                          ▷ walks per node
9          Set Π to be a random permutation of the nodes V
10         for each v ∈ Π in order do
11             S = AttributedWalk(G', X, v, Φ, L)
12             Add the attributed walk S to S
13     end parallel
14     α = StochasticGradientDescent(ω, D, S)                    ▷ parallel
15     return the learned role embeddings α
16 procedure AttributedWalk(G', X, start node s, function Φ, L)
17     Initialize attributed walk S to [Φ(x_s)]
18     Set i = s                                                 ▷ current node
19     for ℓ = 1 to L − 1 do
20         Γ_i = Set of the neighbors for node i
21         j = AliasSample(Γ_i, π)                               ▷ select node j ∈ Γ_i
22         Append Φ(x_j) to S
23         Set i to be the current node j
24     return attributed walk S of length L rooted at node s
```

algorithms [1, 5]. Since many graph properties including motifs exhibit power law distributions, we preprocess $\mathbf{X}$ using logarithmic binning, similar to [25] (Line 4). In Line 5, vertices are mapped to vertex-roles using a function $\Phi(\mathbf{x})$ as discussed in Section 2.1. Then, we precompute the random walk transition probabilities π, which could be uniform or weighted (Line 6). Lines 8-13 initiate random walks from each vertex using the notion of attributed random walks (using alias sampling).Finally, *Role2Vec* learns the embeddings using stochastic gradient descent in Line 14.

Recall that N_v is the number of nodes, M is the number of roles, and $M \ll N_v$. *Role2vec* has the following properties.

Corollary 2.1. Role2vec *is space-efficient with space complexity $O(MD + N_v)$.*

Corollary 2.2. *As $M \to N_v$ then we recover the traditional random walk methods [22, 42] as a special case of the framework.*

This is straightforward to see. Suppose $M \to N_v$, then *Role2vec* converges to the baseline random walk methods [22, 42], since each vertex is mapped to a new role that uniquely identifies it from other vertices, *i.e.*, Φ is a one-to-one function from V onto itself. Intuitively, this implies that each node is assigned a unique role that uniquely identifies it and thus corresponds exactly to the baseline random walk methods [22, 42] that use simple random walks based on node identity. Furthermore, notice that $M \to 1$ and $M \to N_v$ correspond to the two most extremes in the framework and the best algorithm is likely to lie in between these two extremes.

3 EXPERIMENTS

We investigate the effectiveness of the proposed framework using a variety of graphs. All graphs have been made available at NetworkRepository [45].[1]

Experimental Setup. All experiments use logarithmic binning[2] with the bin size δ chosen by searching over $\delta \in \{0.01, 0.1, 0.5, 0.9, 0.99\}$. In these experiments, we use a simple function $\Phi(\mathbf{x})$ that represents a concatenation of the attribute values in the node attribute vector $\mathbf{x}$. We searched over 10 subsets of the 9 motif features of size 2-4 nodes shown in Figure 1. We evaluate the *role2vec* approach presented in Section 2.3 that leverages the attributed random walk framework (Section 2) against a number of baseline methods including: node2vec [22], DeepWalk [42], struc2vec [44], and LINE [51]. For our approach and node2vec, we use the same hyperparameters ($D = 128$, $R = 10$, $L = 80$) and grid search over $p, q \in \{0.25, 0.50, 1, 2, 4\}$ as mentioned in [22]. For link prediction, we use logistic regression (LR) with an L2 penalty. The model is selected using 10-fold cross-validation on 10% of the labeled data. Experiments are repeated for 10 random seed initializations. All results are statistically significant with p-value < 0.01. We use AUC to evaluate the models. Data is available online [45].

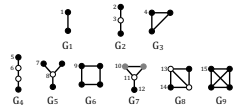

Figure 1: Summary of the 9 graphlets (motifs) and 15 orbits with 2-4 nodes.

Table 1: AUC scores for various methods for link prediction using $(\boldsymbol{\alpha}_i + \boldsymbol{\alpha}_j)/2$. Note R2V=Role2vec, N2V=node2vec, DW=DeepWalk and S2V=struc2vec.

Graph	R2V	N2V	DW	LINE	S2V
bn–cat	**0.710**	0.627	0.627	0.672	0.669
bn–rat–brain	**0.748**	0.716	0.716	0.691	0.729
bn–rat–cerebral	**0.867**	0.813	0.811	0.709	0.858
ca–CSphd	**0.838**	0.768	0.735	0.620	0.791
eco–fweb–baydry	**0.681**	0.655	0.627	0.660	0.623
ia–radoslaw–email	**0.867**	0.756	0.745	0.769	0.857
soc–anybeat	**0.961**	0.854	0.848	0.850	0.883
soc–dolphins	**0.656**	0.580	0.498	0.551	0.590
fb–Yale4	**0.793**	0.742	0.728	0.763	0.758
web–EPA	**0.926**	0.804	0.738	0.768	0.861

Comparison. We compare the proposed approach to other embedding methods for link prediction. Given a partially observed graph G with a fraction of missing edges, the link prediction task is to predict these missing edges. We generate a labeled dataset of edges as done in [22]. Positive examples are obtained by removing 50% of edges randomly, whereas *negative examples* are generated by randomly sampling an equal number of node pairs that are not connected with an edge, *i.e.*, each node pair $(i, j) \notin E$. For each method, we learn features using the remaining graph that consists of only positive examples. Using the learned embeddings from each method, we then learn a model to predict whether a given edge in the test set exists in E or not. Notice that embedding methods such as DeepWalk and node2vec require each vertex in G to appear in at least one edge in the training graph, otherwise these methods are unable to derive features for such vertices.

For comparison, we use the same set of binary operators as in [22] to construct features for the edges by combining the learned embeddings of its endpoints. The AUC results are provided in Table 1 and 2. The AUC scores from our method are all significantly better than the other methods at $p < 0.01$. In all cases, our method achieves better predictive performance over the other methods across a wide variety of graphs with different characteristics. Overall, the mean and product binary operators give an average gain in predictive performance (over all graphs) of 11.1% and 22%, respectively. As an aside, depending on the graph, the number of roles M can sometimes be relatively large. However, in all cases, the best M is obviously less than N_v. We note that our approach directly models the similarity between the vertices, hence, the embeddings learned by *Role2Vec* models the network structure better than traditional methods, capturing both homophily and heterophily [36].

4 CONCLUSION

In this work, we proposed the notion of role-based graph embeddings. Instead of learning individual embeddings for each node, embeddings are learned for each role based on functions that map feature vectors to roles. To learn such role-based embeddings, we proposed the notion of attributed random walk. Using this notion, we described a framework called Role2Vec that serves as a basis for generalizing existing methods that use random walks (based on vertex ids). The proposed framework enables these methods to be more widely applicable by learning functions that capture the behavioral roles of nodes. Instead of learning individual embeddings for each node, our approach learns embeddings for each role based on functions that map feature vectors to roles. It was shown that many existing methods are actually a special case in the Role2Vec framework when the number of roles equals the number of nodes. Finally, we demonstrated the effectiveness of the framework for link prediction task where it is shown to achieve an average gain in AUC of 16.55% while requiring 853x less space than existing methods on a wide variety of graphs.

Table 2: AUC scores for various methods for link prediction using $\boldsymbol{\alpha}_i \odot \boldsymbol{\alpha}_j$. Note R2V=Role2vec, N2V=node2vec, DW=DeepWalk and S2V=struc2vec.

Graph	R2V	N2V	DW	LINE	S2V
bn–cat	**0.694**	0.621	0.621	0.494	0.662
bn–rat–brain	**0.775**	0.715	0.715	0.562	0.736
bn–rat–cerebral	**0.867**	0.796	0.793	0.498	0.834
ca–CSphd	**0.758**	0.678	0.660	0.533	0.699
eco–fweb–baydry	**0.684**	0.673	0.631	0.516	0.599
ia–radoslaw–email	**0.852**	0.746	0.731	0.471	0.843
soc–anybeat	**0.945**	0.730	0.728	0.618	0.798
soc–dolphins	**0.787**	0.593	0.508	0.549	0.553
fb–Yale4	**0.940**	0.912	0.904	0.784	0.905
web–EPA	**0.907**	0.808	0.797	0.650	0.841

[1] http://networkrepository.com/

[2] Logarithmic binning assigns the first δN_v nodes with smallest attribute value to 0 (where $0 < \delta < 1$), then assigns the δ fraction of remaining unassigned nodes with smallest value to 1, and so on.

REFERENCES

[1] Nesreen K. Ahmed, Jennifer Neville, Ryan A. Rossi, and Nick Duffield. 2015. Efficient Graphlet Counting for Large Networks. In *ICDM*. 10.

[2] Nesreen K. Ahmed, Ryan A. Rossi, Theodore L. Willke, and Rong Zhou. 2017. *Edge Role Discovery via Higher-Order Structures*. 291–303.

[3] Leman Akoglu, Hanghang Tong, and Danai Koutra. 2015. Graph based anomaly detection and description: a survey. *DMKD* 29, 3 (2015), 626–688.

[4] Mohammad Al Hasan and Mohammed J Zaki. 2011. A survey of link prediction in social networks. In *Social Network Data Analytics*. 243–275.

[5] Austin R Benson, David F Gleich, and Jure Leskovec. 2016. Higher-order organization of complex networks. *Science* 353, 6295 (2016), 163–166.

[6] Toine Bogers. 2010. Movie recommendation using random walks over the contextual graph. In *Context-Aware Recommender Systems*.

[7] Aleksandar Bojchevski and Stephan Günnemann. 2017. Deep Gaussian Embedding of Attributed Graphs: Unsupervised Inductive Learning via Ranking. *arXiv:1707.03815* (2017).

[8] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. GraRep: Learning graph representations with global structural information. In *CIKM*. ACM, 891–900.

[9] Sandro Cavallari, Vincent W Zheng, Hongyun Cai, Kevin Chen-Chuan Chang, and Erik Cambria. 2017. Learning community embedding with community detection and node embedding on graphs. In *CIKM*. 377–386.

[10] Shiyu Chang, Wei Han, Jiliang Tang, Guo-Jun Qi, Charu C Aggarwal, and Thomas S Huang. 2015. Heterogeneous network embedding via deep architectures. In *SIGKDD*. 119–128.

[11] Winnie Cheng, Chris Greaves, and Martin Warren. 2006. From n-gram to skipgram to concgram. *Int. J. of Corp. Linguistics* 11, 4 (2006), 411–433.

[12] Fan Chung. 2007. Random walks and local cuts in graphs. *Linear Algebra and its applications* 423, 1 (2007), 22–32.

[13] Connor W Coley, Regina Barzilay, William H Green, Tommi S Jaakkola, and Klavs F Jensen. 2017. Convolutional Embedding of Attributed Molecular Graphs for Physical Property Prediction. *J. Chem. Info. & Mod.* (2017).

[14] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable Representation Learning for Heterogeneous Networks. In *SIGKDD*. 135–144.

[15] M.G. Everett. 1985. Role similarity and complexity in social networks. *Social Networks* 7, 4 (1985), 353–359.

[16] L. Getoor and B. Taskar (Eds.). 2007. *Intro. to SRL*. MIT Press.

[17] Sean Gilpin, Tina Eliassi-Rad, and Ian Davidson. 2013. Guided learning for role discovery (GLRD): framework, algorithms, and applications. In *SIGKDD*. 113–121.

[18] David F. Gleich and Ryan A. Rossi. 2014. A Dynamical System for PageRank with Time-Dependent Teleportation. *Inter. Math.* (2014), 188–217.

[19] Timothy E Goldsmith and Daniel M Davenport. 1990. Assessing structural similarity of graphs. (1990).

[20] Palash Goyal and Emilio Ferrara. 2017. Graph Embedding Techniques, Applications, and Performance: A Survey. *arXiv preprint arXiv:1705.02801* (2017).

[21] Leo Grady. 2006. Random walks for image segmentation. *TPAMI* 28, 11 (2006), 1768–1783.

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

[23] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. *arXiv:1706.02216* (2017).

[24] Zellig S Harris. 1954. Distributional structure. *Word* 10, 2-3 (1954), 146–162.

[25] Keith Henderson, Brian Gallagher, Lei Li, Leman Akoglu, Tina Eliassi-Rad, Hanghang Tong, and Christos Faloutsos. 2011. It's who you know: graph mining using recursive structural features. In *SIGKDD*. ACM, 663–671.

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

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

[28] Akshay Java, Pranam Kolari, Tim Finin, and Tim Oates. 2006. Modeling the spread of influence on the blogosphere. In *WWW*. 22–26.

[29] Mehmet Koyutürk, Yohan Kim, Umut Topkara, Shankar Subramaniam, Wojciech Szpankowski, and Ananth Grama. 2006. Pairwise alignment of protein interaction networks. *JCB* 13, 2 (2006), 182–199.

[30] Ankit Kuwadekar and Jennifer Neville. 2011. Relational active learning for joint collective classification models. In *ICML*. 385–392.

[31] Jean-Louis Lassez, Ryan Rossi, and Kumar Jeev. 2008. Ranking Links on the Web: Search and Surf Engines. In *IEA/AIE*. 199–208.

[32] Jiongqian Liang, Peter Jacobs, and Srinivasan Parthasarathy. 2017. SEANO: Semi-supervised Embedding in Attributed Networks with Outliers. *arXiv:1703.08100* (2017).

[33] W. Liu and L. Lü. 2010. Link prediction based on local random walk. *Europhysics Letters* 89 (2010), 58007.

[34] Francois Lorrain and Harrison C White. 1977. Structural equivalence of individuals in social networks. In *Social Networks*. Elsevier, 67–98.

[35] László Lovász. 1993. Random walks on graphs. *Combinatorics* 2 (1993), 1–46.

[36] Miller McPherson, Lynn Smith-Lovin, and James M Cook. 2001. Birds of a feather: Homophily in social networks. *Annual review of sociology* 27, 1 (2001), 415–444.

[37] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *ICLR Workshop*. 10.

[38] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[39] Jennifer Neville and David Jensen. 2000. Iterative classification in relational data. In *AAAI SRL Workshop*. 13–20.

[40] Andrew Y Ng, Michael I Jordan, and Yair Weiss. 2002. On spectral clustering: Analysis and an algorithm. In *NIPS*. 849–856.

[41] L. Page, S. Brin, R. Motwani, and T. Winograd. 1998. PageRank citation ranking: Bringing order to the web. *Stanford Tech. Report* (1998).

[42] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *SIGKDD*. 701–710.

[43] Pascal Pons and Matthieu Latapy. 2006. Computing communities in large networks using random walks. *J. Graph Alg. Appl.* 10, 2 (2006), 191–218.

[44] Leonardo F.R. Ribeiro, Pedro H.P. Saverese, and Daniel R. Figueiredo. 2017. Struc2Vec: Learning Node Representations from Structural Identity. In *SIGKDD*. 385–394.

[45] Ryan A. Rossi and Nesreen K. Ahmed. 2015. The Network Data Repository with Interactive Graph Analytics and Visualization. In *AAAI*. 4292–4293. http://networkrepository.com

[46] Ryan A. Rossi and Nesreen K. Ahmed. 2015. Role Discovery in Networks. *TKDE* 27, 4 (2015), 1112–1131.

[47] Ryan A. Rossi, Rong Zhou, and Nesreen K. Ahmed. 2017. Deep Feature Learning for Graphs. In *arXiv:1704.08829*.

[48] Maja Rudolph, Francisco Ruiz, Stephan Mandt, and David Blei. 2016. Exponential family embeddings. In *Advances in Neural Information Processing Systems*. 478–486.

[49] Sergio D Servetto and Guillermo Barrenechea. 2002. Constrained random walks on random graphs: routing algorithms for large scale wireless sensor networks. In *Wireless Sensor Networks & App*. 12–21.

[50] Chuan Shi, Xiangnan Kong, Yue Huang, S Yu Philip, and Bin Wu. 2014. HeteSim: A General Framework for Relevance Measure in Heterogeneous Networks. *TKDE* 26, 10 (2014), 2479–2492.

[51] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale Information Network Embedding. In *WWW*. 1067–1077.

[52] S. Wasserman and K. Faust. 1994. *Social network analysis: Methods and applications*. Cambridge University Press.

[53] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y Chang. 2015. Network Representation Learning with Rich Text Information.. In *IJCAI*. 2111–2117.

[54] Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. 2016. Revisiting semi-supervised learning with graph embeddings. *arXiv:1603.08861* (2016).

[55] Laura A Zager and George C Verghese. 2008. Graph similarity scoring and matching. *Applied mathematics letters* 21, 1 (2008), 86–94.

A SUPPLEMENTARY MATERIALS

A.1 Theoretical Analysis

We analyze the properties/parameters of attributed random walks. Lemmas 1–3 analyze the constraints and bounds on vertex reachability, expected access time, and representation of vertices/edges in attributed random walks.

We consider an attributed random walk of length L and starting at vertex v_0 of G, if at time t we are at vertex v_t with role $\Phi(\mathbf{x}_t)$, then at time $t+1$, we move to a neighbor of v_t with probability $1/d_{v_t}$. Clearly, the randomly chosen sequence of vertex roles $(\Phi(\mathbf{x}_t) : t = 0, 1, \ldots, L-1)$ is a Markov chain. We denote by P_t the distribution of v_t, where $P_i(t) = \Pr(v_t = i)$ is the probability that the attributed random walk visits vertex i at time t. Similarly, we denote by P_{ij} the transition probability from vertex i to vertex j in one step, where $P_{ij} = \Pr(v_t = j | v_{t-1} = i)$. Thus, the Markov property implies that this Markov chain is uniquely defined by its *one-step* transition matrix $\mathbf{P} = (P_{ij})_{v_i, v_j \in V}$,

$$P_{ij} = \begin{cases} 1/d_i, & \text{if } (i,j) \in E \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

Let $\mathbf{P}^m$ be the transition matrix whose entries are the m-step transition probabilities, such that

$$P_{ij}^m = \Pr(v_{t+m} = j \mid v_t = i) \tag{11}$$

is the probability that the attributed walk moves from vertex i to vertex j in exactly m steps. Finally, we denote by r_{ij}^t the probability that starting at vertex i, the *first transition* to vertex j occurs at time t,

$$r_{ij}^t = \Pr(v_t = j, \forall 1 \leq m \leq t-1, v_m \neq j \mid v_0 = i) \tag{12}$$

Lemma 1. *If u and v are two non-adjacent vertices in a connected graph G, then there is at least one neighbor $j \in \Gamma(u)$ where $r_{uj}^t \leq r_{jv}^{t_1}$ for $t_1 < t$.*

Proof. Let $d_u = |\Gamma(u)|$ be the degree of vertex u, and denote by $[d_u]$ the set of neighbors of u. For each neighbor $j \in [d_u]$, start an attributed random walk at j, and let $r_{jv}^{t_1}$ be the probability that the first transition from j to v occurs at time t_1. Now begin a attributed random walk at u and let r_{uv}^t be the probability that the first transition from u to v occurs at time t. By conditioning on the first transition, we have

$$r_{uv}^t = \sum_{j=1}^{d_u} P_{uj} \cdot r_{jv}^{t-1} = \frac{1}{d_u} \sum_{j=1}^{d_u} r_{jv}^{t-1}$$

Set $t_1 = t-1$, thus the probability r_{uv}^t is the mean of the probabilities of u's neighbors, $r_{jv}^{t_1}$ for $j \in [d_u]$ and $t_1 < t$. This implies that there is at least one neighbor j where $r_{uv}^t \leq r_{jv}^{t_1}$ for $t_1 < t$, and Lemma 1 is proved. ∎

Lemma 1 shows that the probability r_{uv}^t is upper bounded by the probability of at least one of u's neighbors (*i.e.*, $r_{jv}^{t_1}$). Thus, an attributed random walk starting at any vertex u will likely visit *nearby* vertices first before visiting *distant* vertices. And that a distant vertex v is more reachable from some neighbor j in less steps (*i.e.*, $t_1 < t$).

Lemma 2. *If u and v are two non-adjacent vertices in a connected graph G, with h_{uv} is the expected access time from u to v, and $\tilde{h}_{jv}$ is the average neighbor access time for u, then with probability less than $1/2$, an attributed random walk starting at u takes at least $L = 2\ h_{uv} > 2\ \tilde{h}_{jv}$ to reach v.*

Proof. Recall that r_{uv}^t is the probability that starting at u, the attributed random walk first visits v at time t, then the expected access time from u to v is $\mathbb{E}[t] = h_{uv} = \sum_{t \geq 1} t \cdot r_{uv}^t$. By conditioning on the first transition, we have

$$\begin{aligned} h_{uv} &= \sum_{t \geq 1} t \cdot \sum_{j=1}^{d_u} P_{uj} \cdot r_{jv}^{t-1} \\ &= \sum_{j=1}^{d_u} P_{uj} + \sum_{t>1} t \cdot \sum_{j=1}^{d_u} P_{uj} \cdot r_{jv}^t \\ &= 1 + \frac{1}{d_u} \sum_{j=1}^{d_u} \sum_{t>1} t \cdot r_{jv}^t = 1 + \frac{1}{d_u} \sum_{j=1}^{d_u} h_{jv} \end{aligned}$$

where d_u is the degree of u, and h_{jv} is the expected access time for some neighbor vertex $j \in [d_u]$. Since $t \geq 0$ for any vertex in G, then by Markov's inequality, for any $L > 0$,

$$\Pr(t \geq L) \leq \frac{\mathbb{E}[t]}{L} = \frac{h_{uv}}{L}$$

Let $\tilde{h}_{jv} = \frac{1}{d_u} \sum_{j=1}^{d_u} h_{jv}$ be the average neighbor access time for vertex u. Then, with $\Pr(t \geq L) \leq 1/2$, an attributed random walk starting at u takes at least $L = 2\ h_{uv} > 2\ \tilde{h}_{jv}$ to reach v. ∎

Lemma 3. *Suppose we start d_u attributed random walks of length L from any vertex $u \in V$ in G. For a given edge $e = (v, v')$, let I_e denote the total number of attributed random walks containing e. Then, the expectation of the random variable I_e is upper bounded by L,* i.e., $\mathbb{E}[I_e] \leq L$.

Proof. Recall that the probability of an attributed random walk starting at u visits v at time t is $P_{uv}^t = \boldsymbol{\lambda}_u \mathbf{P}^t \boldsymbol{\lambda}_v^\intercal$, where $\boldsymbol{\lambda}_u$ is the indicator vector for vertex u, which equals 1 in coordinate u and 0 otherwise. Then, for a given edge $e = (v, v')$, the probability that the attributed random walk visits v at time t and v' at time $t+1$ is $\boldsymbol{\lambda}_u \mathbf{P}^t \boldsymbol{\lambda}_v^\intercal / d_v$ (since the transition probability from v to v' is $1/d_v$). Suppose we start d_u attributed random walks of length L from u, let I_e denote the total number of attributed random walks containing e, then the expectation of I_e is the sum of the probabilities that there exists an attributed random walk visiting $e = (v, v')$ as follows

$$\begin{aligned} \mathbb{E}[I_e] &\leq \sum_{t=1}^{L} \sum_{u} d_u \boldsymbol{\lambda}_u \mathbf{P}^t \boldsymbol{\lambda}_v^\intercal / d_v \\ &= \sum_{t=1}^{L} \mathbf{1}\mathbf{D}\mathbf{P}^t \boldsymbol{\lambda}_v^\intercal / d_v = \sum_{t=1}^{L} \mathbf{1}\mathbf{D}\boldsymbol{\lambda}_v^\intercal / d_v = \sum_{t=1}^{L} 1 = L \end{aligned}$$

where $\mathbf{D}$ is the degree matrix with the ith diagonal entry is the vertex degree d_i, $\mathbf{1}$ is the unit vector with all entries equal to 1, and $(\mathbf{1}\mathbf{D})\mathbf{P}^t = \mathbf{1}\mathbf{D}$. ∎

The above implies attributed random walks are a generalization of random walks. For instance, if $M = N_v$, then random walks are recovered.

Corollary A.1. Role2vec *is space-efficient with space complexity $O(MD + N_v)$.*

Proof. To store the learned embeddings of the vertex-roles, *Role2vec* takes $O(MD)$ space. Also, *Role2vec* takes $O(N_v)$ space for a hash table mapping vertices to their corresponding roles. Thus, the total space used by *Role2vec* is $O(MD+N_v)$. Since $M \ll N_v$, then *Role2vec* takes less space compared to baseline methods that require $O(N_v D)$ space. ∎

A.2 Space-efficient Embeddings

We now investigate the space-efficiency of the learned embeddings from the proposed framework and intermediate representation. Observe that any embedding method that implements the proposed attributed random walk framework (and intermediate representation) learns an embedding for each distinct node role $w \in W$. As described earlier in Sec. 2.3, in the worst case, an embedding is learned for each of the N_v vertices in the graph and we recover the baseline methods [22, 42] as a special case. In general, the best embedding most often lies between such extremes and therefore the embedding learned from a method implementing *Role2Vec* is often orders of magnitude smaller in size, since $M \ll N_v$.

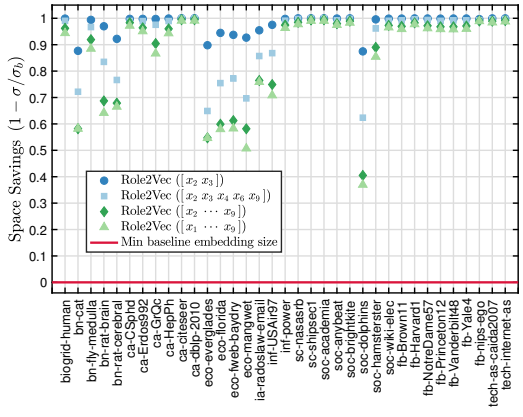

Figure 2: Evaluating the space savings of Role2Vec.

Given an attribute vector $\mathbf{x}$ of motif counts (Figure 1) for an arbitrary node in G, we derive embeddings using each of the following:

$$\Phi(\mathbf{x}_i = [\,\mathrm{x}_2\ \mathrm{x}_3\,]), \quad \text{for } i = 1, ..., N_v \tag{13}$$

$$\Phi(\mathbf{x}_i = [\,\mathrm{x}_2\ \mathrm{x}_3\ \mathrm{x}_4\ \mathrm{x}_6\ \mathrm{x}_9\,]), \quad \text{for } i = 1, ..., N_v \tag{14}$$

$$\Phi(\mathbf{x}_i = [\,\mathrm{x}_2\ \mathrm{x}_3\ \mathrm{x}_4\ \mathrm{x}_5\ \cdots\ \mathrm{x}_9\,]), \quad \text{for } i = 1, ..., N_v \tag{15}$$

$$\Phi(\mathbf{x}_i = [\,\mathrm{x}_1\ \mathrm{x}_2\ \mathrm{x}_3\ \mathrm{x}_4\ \mathrm{x}_5\ \cdots\ \mathrm{x}_9\,]), \quad \text{for } i = 1, ..., N_v \tag{16}$$

where $\Phi(\cdot)$ is a function that maps $\mathbf{x}_i$ to a role $w \in W$. In these experiments, we use logarithmic binning (applied to each N_v-dimensional motif feature) with $\delta = 0.5$ and use Φ defined as the concatenation of the logarithmically binned attribute values. Embeddings are learned using the different subsets of attributes in Eq. (13)-(16). For instance, Eq. (13) indicates that vertex roles are derived using the (logarithmic binned) number of 2-stars x_2 and triangles x_3 incident to the given vertex (Figure 1). We measure the space (in bytes) required to store the embedding learned by each method. As shown in Figure 2, space savings are defined as the reduction in embedding size (bytes) relative to the smallest embedding size among the baseline methods. Let σ denote the size (bytes) of a Role2Vec embedding and σ_b be the smallest embedding size from the baseline methods, then space savings is defined as $1 - \frac{\sigma}{\sigma_b}$. Hence, larger values indicate a larger space savings (reduction in size). In all cases, the embeddings learned from our approach require significantly less space (require on average 853x less space across all graphs) and thus more space-efficient.

A.3 Extensive Related Work

Random walks are widely used in graph embedding methods [9, 22, 42, 44]. Recent embedding methods for graphs have largely been based on the popular skip-gram model [11, 37] originally introduced for learning vector representations of words in text. In particular, DeepWalk [42] used this approach to embed the nodes such that the co-occurrence frequencies of pairs in short random walks are preserved. Recently, node2vec [22] introduced hyperparameters to DeepWalk that tune the depth and breadth of the random walks. These approaches are becoming increasingly popular and have been shown to outperform a number of existing methods. These methods (and many others) are all based on simple random walks and thus are well-suited for generalization using the *attributed random walk framework*. While most network representation learning methods use only the graph [8, 22, 42, 51], our framework exploits both the graph *and* structural features (*e.g.*, motifs).

Yang *et al.* [54] proposed an approach called Planetoid. Planetoid is an embedding-based approach for semi-supervised learning and does not use any structural features. Rossi *et al.* [47] proposed an approach for (attributed) networks called DeepGL that learns relational functions representing compositions of one or more operators applied to an initial set of graph features. Recently, Hamilton *et al.* [23] proposed a similar approach that also aggregates features from node neighborhoods. Heterogeneous networks [50] have also been recently considered [10, 14] as well as attributed networks [26, 27]. Huang *et al.* [27] proposed an approach for attributed networks with labels whereas Yang *et al.* [53] used text features to learn node representations. Liang *et al.* [32] proposed a semi-supervised approach for networks with outliers. Bojchevski *et al.* [7] proposed an unsupervised rank-based approach. Coley *et al.* [13] introduced a convolutional approach for attributed molecular graphs that learns graph embeddings rather than nodes.

Our work is also related to uniform and non-uniform random walks on graphs [12, 35]. Random walks are at the heart of many important applications such as ranking [41], community detection [40, 43], recommendation [6], link prediction [33], influence modeling [28], search engines [31], image segmentation [21], routing in wireless sensor networks [49], and time-series forecasting [18]. These applications and techniques may also benefit from the notion of *attributed random walks*.

Finally, our work is also related to role discovery [15] including recent feature-based roles [46]. In particular, recent feature-based role discovery methods [46] learn behavior roles automatically from the graph by first extracting a set of features from the graph and then using a (non-negative) matrix/tensor factorization to derive a low-rank approximation of $\mathbf{X}$ capturing the role mixed-memberships of nodes [17] or edges [2] in the graph.