struc2vec: Learning Node Representations from Structural Identity

Leonardo F. R. Ribeiro
Federal University of Rio de Janeiro
Systems Eng. and Comp. Science Dep.
leo@land.ufrj.br

Pedro H. P. Saverese
Federal University of Rio de Janeiro
Systems Eng. and Comp. Science Dep.
savarese@land.ufrj.br

Daniel R. Figueiredo
Federal University of Rio de Janeiro
Systems Eng. and Comp. Science Dep.
daniel@land.ufrj.br

Structural identity is a concept of symmetry in which network nodes are identified according to the network structure and their relationship to other nodes.

struc2vec uses a hierarchy to measure node similarity at different scales, and constructs a multilayer graph to encode structural similarities and generate structural context for nodes.

struc2vec improves performance on classification tasks that depend more on structural identity.

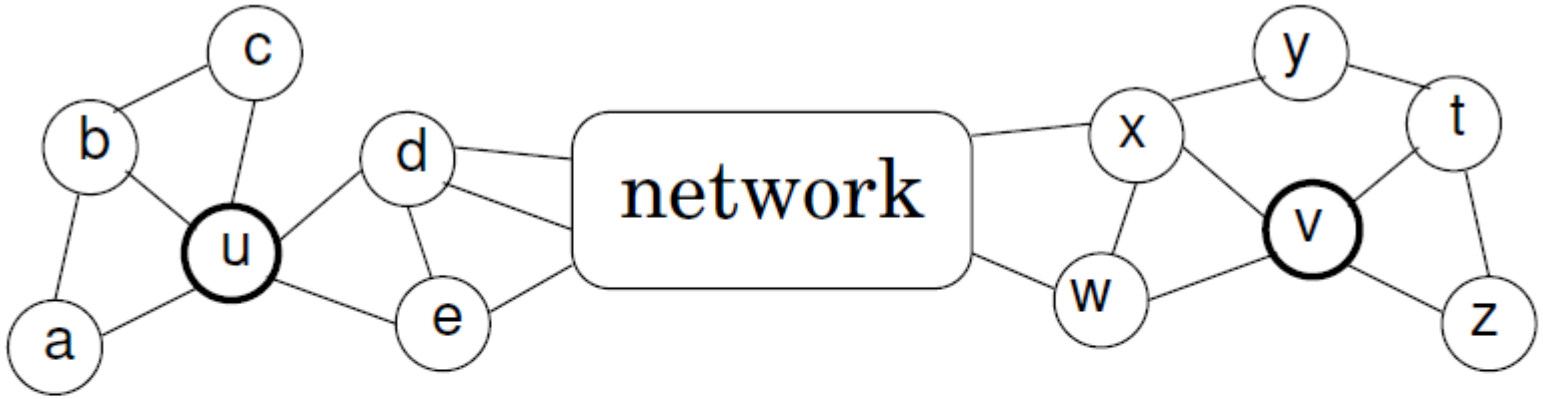


Figure 1: An example of two nodes (u and v) that are structurally similar (degrees 5 and 4, connected to 3 and 2 triangles, connected to the rest of the network by two nodes), but very far apart in the network.

Figure 1 illustrates the
problem, where nodes u and v play similar roles (i.e., have similar local structures) but are very far apart in the network. Since their neighborhoods have no common nodes, recent approaches cannot capture their structural similarity (as we soon show).

- Assess structural similarity between nodes independently of node and edge attributes as well as their position in the network. Thus, two nodes that have a similar local structure will be considered so, independent of network position and node labels in their neighborhoods. Our approach also does not require the network to be connected, and identifies structurally similar nodes in different connected components.
- Establish a hierarchy to measure structural similarity, allowing progressively more stringent notions of what it means to be structurally similar. In particular, at the bottom of the hierarchy, structural similarity between nodes depend only on their degrees, while at the top of the hierarchy similarity depends on the entire network (from the viewpoint of the node).
- Generates random *contexts* for nodes, which are sequences of structurally similar nodes as observed by a weighted random walk traversing a multilayer graph (and *not* the original network). Thus, two nodes that frequently appear with similar contexts will likely have similar structure. Such context can be leveraged by language models to learn latent representation for the nodes.

Steps:

(1) Determine the structural similarity between each vertex pair in the graph for different neighborhood sizes. This induces a hierarchy in the measure for structural similarity between nodes, providing more information to assess structural similarity at each level of the hierarchy.
(2) Construct a weighted multilayer graph where all nodes in the network are present in every layer, and each layer corresponds to a level of the hierarchy in measuring structural similarity. Moreover, edge weights among every node pair within each layer are inversely proportional to their structural similarity.
(3) Use the multilayer graph to generate context for each node. In particular, a biased random walk on the multilayer graph is used to generate node sequences. These sequences are likely to include nodes that are more structurally similar.
(4) Apply a technique to learn latent representation from a context given by the sequence of nodes, for example, Skip-Gram.

Measuring structural similarity

Let $G = (V, E)$ vertex set V and edge set E, where $n = |V|$ k^* its diameter.

$R_k(u)$ denotes the ring of nodes at distance k. $R_1(u)$ denotes the set of neighbors of u and in general,

$s(S)$ denote the ordered degree sequence of a set $S \subset V$ of nodes.

let $f_k(u, v)$ denote the *structural distance* between u and v

$$f_k(u, v) = f_{k-1}(u, v) + g(s(R_k(u)), s(R_k(v))),$$
$$k \geq 0 \text{ and } |R_k(u)|, |R_k(v)| > 0$$

where $g(D_1, D_2) \geq 0$ measures the distance between the ordered degree sequences D_1 and D_2 and $f_{-1} = 0$.

Informally, DTW finds the optimal alignment between two sequences A and B. Given a distance function $d(a, b)$ for the elements of the sequence, DTW matches each element $a \in A$ to $b \in B$, such that the sum of the distances between matched elements is minimized. Since elements of sequence A and B are degrees of nodes, we adopt the following distance function:

$$d(a, b) = \frac{\max(a, b)}{\min(a, b)} - 1 \tag{2}$$

Constructing the context graph

M denote the multilayer graph where layer k is defined using the k-hop neighborhoods of the nodes.

Each layer $k = 0, \ldots, k^*$ is formed by a weighted undirected complete graph with node set V, and thus, $\binom{n}{2}$ edges. The edge weight between two nodes in a layer is given by:

$$w_k(u, v) = e^{-f_k(u,v)}, \quad k = 0, \ldots, k^* \tag{3}$$

Thus, every vertex $u \in V$ in layer k is
connected to the corresponding vertex u in layer $k+1$ and $k-1$.
The edge weight between layers are as follows:

$$\begin{aligned} w(u_k, u_{k+1}) &= \log(\Gamma_k(u) + e), \quad k = 0, \ldots, k^* - 1 \\ w(u_k, u_{k-1}) &= 1, \quad k = 1, \ldots, k^* \end{aligned} \tag{4}$$

where $\Gamma_k(u)$ is number of edges incident to u that have weight larger than the average edge weight of the complete graph in layer k. In particular:

$$\Gamma_k(u) = \sum_{v \in V} \mathbb{1}(w_k(u, v) > \overline{w_k}) \tag{5}$$

where $\overline{w_k} = \sum_{(u,v) \in \binom{V}{2}} w_k(u, v) / \binom{n}{2}$. Thus, $\Gamma_k(u)$ measures the similarity of node u to other nodes in layer k.

Generating context for nodes

In particular, we consider a biased random walk that moves around M making random choices according to the weights of M. Before each step, the random walk first decides if it will change layers or walk on the current layer (with probability $q > 0$ the random walk stays in the current layer).

Probability q:

Given that it will stay in the current layer, the probability of stepping from node u to node v in layer k is given by:

$$p_k(u,v) = \frac{e^{-f_k(u,v)}}{Z_k(u)} \tag{6}$$

where $Z_k(u)$ is the normalization factor for vertex u in layer k, simply given by:

$$Z_k(u) = \sum_{\substack{v \in V \\ v \neq u}} e^{-f_k(u,v)} \tag{7}$$

Probability 1-q:

$$p_k(u_k, u_{k+1}) = \frac{w(u_k, u_{k+1})}{w(u_k, u_{k+1}) + w(u_k, u_{k-1})} \tag{8}$$
$$p_k(u_k, u_{k-1}) = 1 - p_k(u_k, u_{k+1})$$

Finally, for each node $u \in V$, we start a random walk in its corresponding vertex in layer 0. Random walks have a fixed and relatively short length (number of steps), and the process is repeated a certain number of times, giving rise to multiple independent walks (i.e., the multiple contexts of node u).

Learning a language model

For this work we use Hierarchical Softmax, where conditional symbol probabilities are calculated using a tree of binary classifiers. For each node $v_j \in V$, Hierarchical Softmax assigns a specific path in the classification tree, defined by a set of tree nodes $n(v_j, 1), n(v_j, 2), \ldots, n(v_j, h)$, where $n(v_j, h) = v_j$. In this setting, we have:

$$P(v_j|v_i) = \prod_{k=1}^{h} C(n(v_j, k), v_i) \tag{9}$$

where C is a binary classifier present in every node in the tree. Note that since Hierarchical Softmax operates on a binary tree, we have that $h = O(\log |V|)$.

We train Skip-Gram according to its optimization problem given by equation (9). Note that while we use Skip-Gram to learn node embeddings, any other technique to learn latent representations for text data could be used in our framework.

Complexity and optimizations

The final complexity is then $O(k^* n^3)$.

Reducing the length of degree sequences (OPT1).

Reducing the number of pairwise similarity calculations (OPT2).

Reducing the number of layers (OPT3).

EXPERIMENTAL EVALUATION

Barbell graph

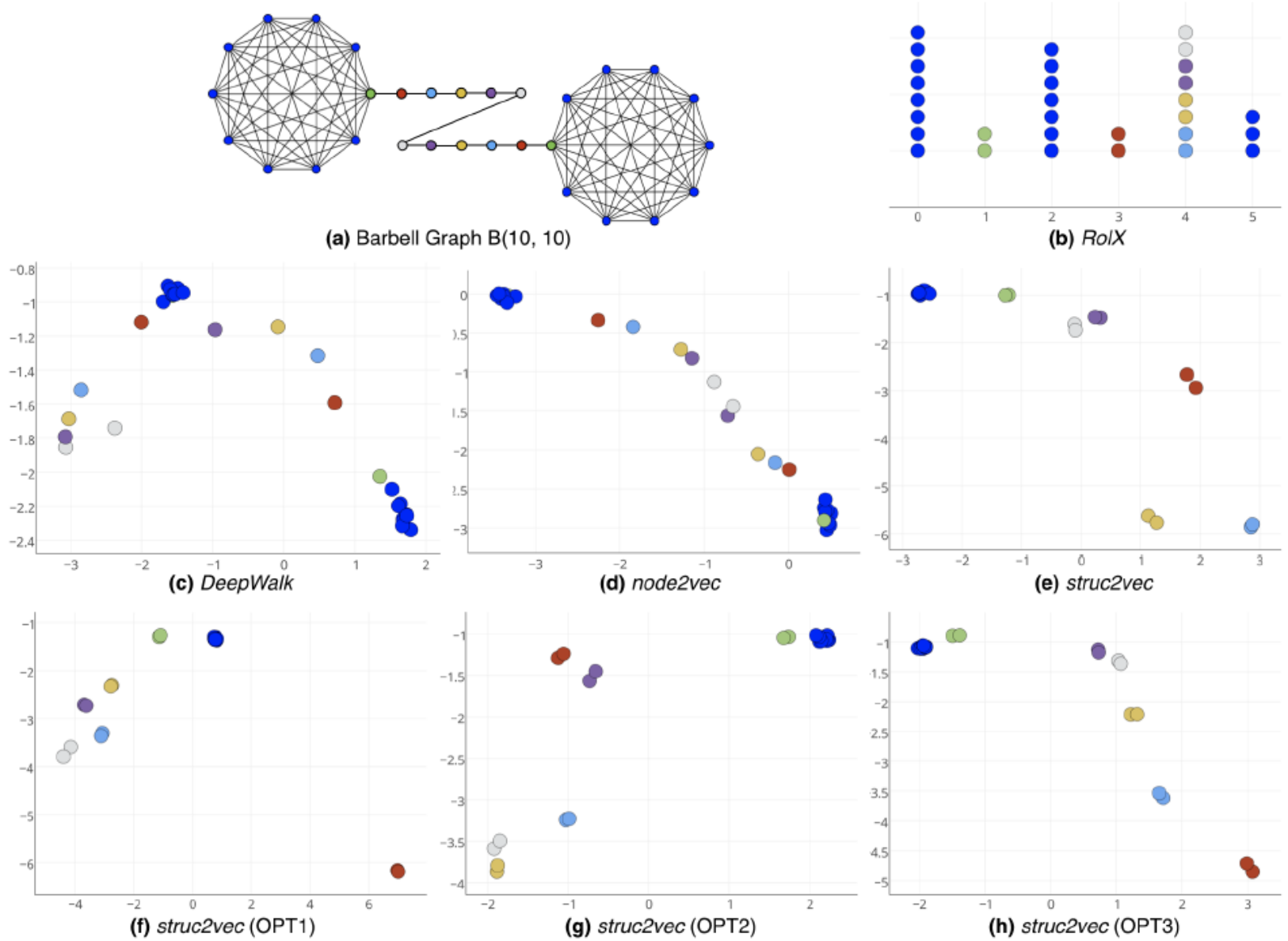
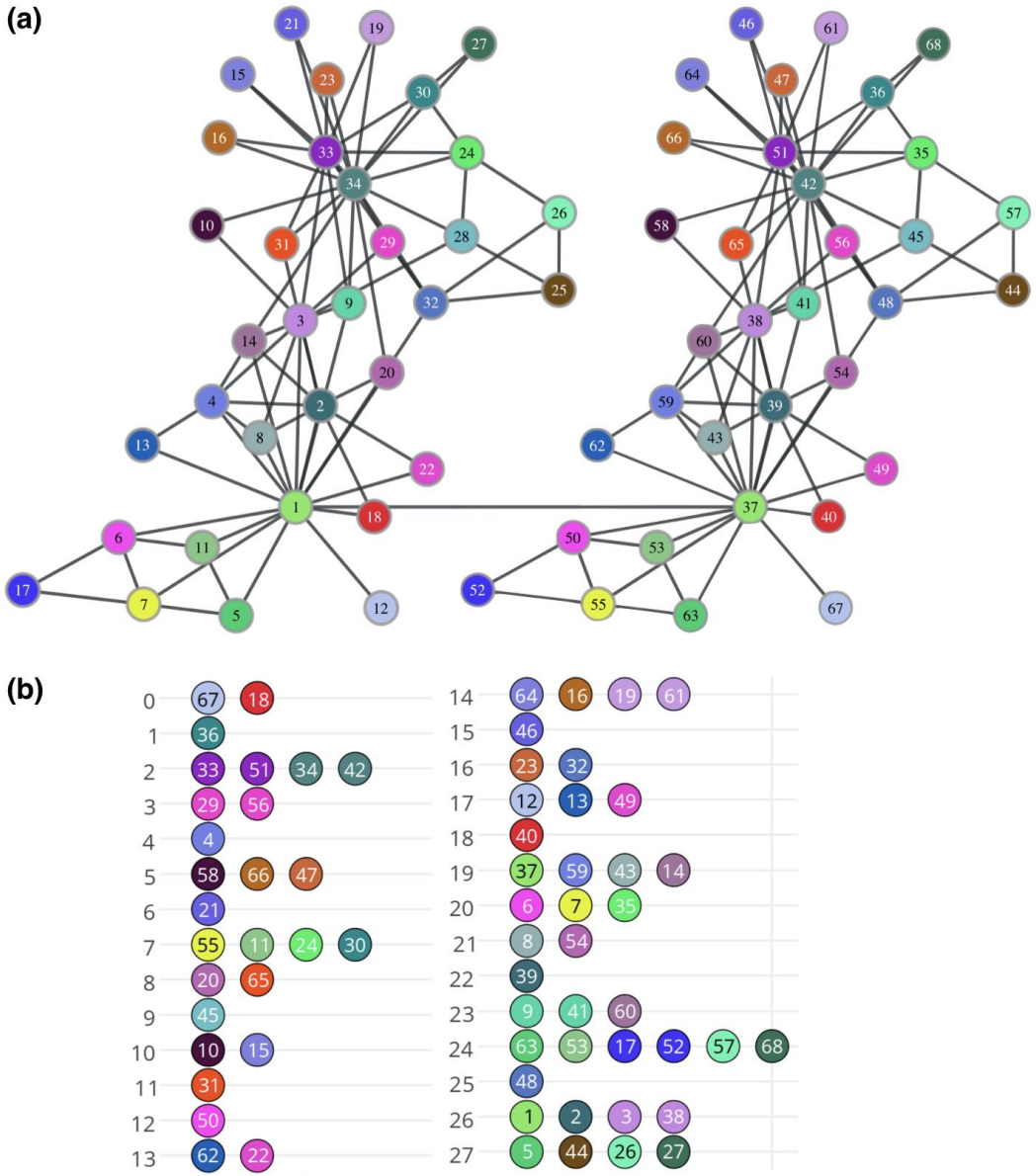
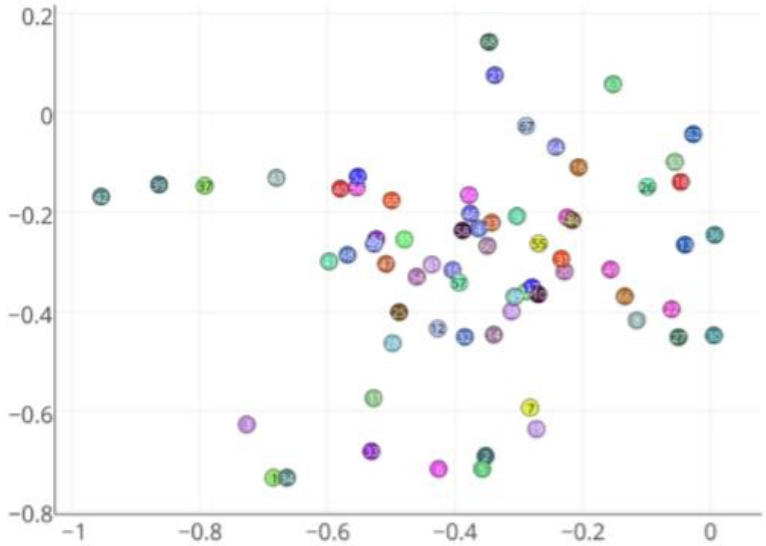


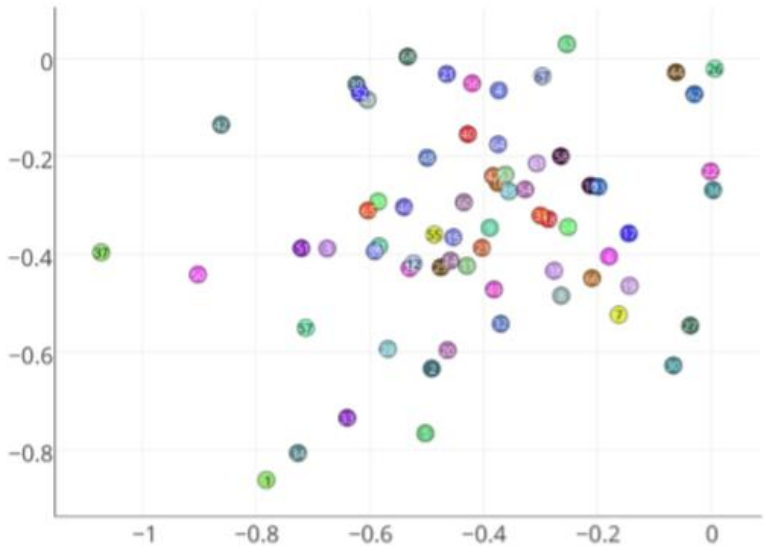
Figure 2: (a) Barbell graph $B(10, 10)$. (b) Roles identified by ***RolX***. Latent representations in $\mathbb{R}^2$ learned by (c) DeepWalk, (d) ***node2vec*** and (e,f,g,h) ***struc2vec***. Parameters used for all methods: number of walks per node: 20, walk length: 80, skip-gram window size: 5. For ***node2vec***: $p = 1$ and $q = 2$.

Karate network

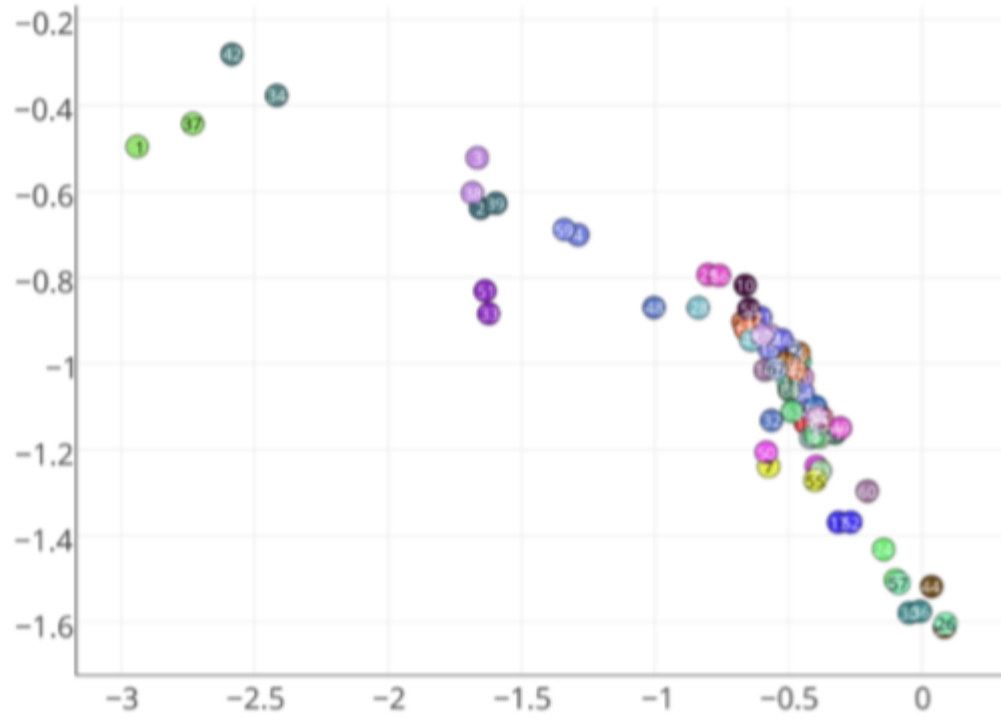




(a) *DeepWalk*



(b) *node2vec*



(c) *struc2vec*