Feature Extraction through Deepwalk on Weighted Graph

Jayesh Soni,
Nagarajan Prabakar
School of Computing and Information Sciences
Florida International University
Miami, USA
{jsoni, prabakar}@fiu.edu

Himanshu Upadhyay
Applied Research Center
Florida International University
Miami, USA
upadhyay@fiu.edu

***Abstract*— Prediction tasks over nodes and edges of a network require careful efforts in defining features used by machine learning algorithms. Current research in the broader field of feature representation learning has led to significant progress in automating prediction of labels by deriving the features. However, modern feature extraction approaches are not expressive enough to capture the diversity of connectivity patterns observed in networks.**

We present a feature extraction technique for a Weighted Graph (DW-WG) using Deep Walk approach, an enhanced neural network model for learning vertex representations of weighted graphs. This approach for machine learning is suitable for a variety of graph structures, including directed graphs, multi-graphs, weighted graphs, and knowledge graphs. Our approach extends the conventional Deepwalk (C-DW) model by allowing the graph to have weighted edges for the feature generation through random-walk and word2vec phases. This approach will enable us to use the predictive power of the weights of the edges of the graph. We argue that this enhancement is the key to learning a rich representation of complex networks. We demonstrate the efficacy of DW-WG over existing state-of-the-art techniques in several real-world networks from diverse domains. We demonstrate the competitiveness of this approach on two network datasets (Google play store and Bitcoin OTC +Alpha).

DW-WG supports scalability and is parallelizable. These qualities make it suitable for complex bigdata sets and a wide variety of real-world applications such as network classification, and anomaly detection.

Keywords- feature extraction; deepwalk; clustering; classification

I. INTRODUCTION

Information networks are diverse in the real world, for instance, publication networks, airline networks, and social networks. Analyzing large information networks has been attracting increased attention in both academia and industry. Graph embedding (feature representation) is very useful in a variety of applications such as visualization [7], node classification [3], link prediction [5], and recommendation [8]. Several machine learning literature [2, 4, 6] propose various methods of graph embedding. They generally perform well on small unweighted graphs. We lose essential information when the processing model does not take into account the weights of edges of a graph. The problem becomes difficult when a real-world information network contains millions of nodes and billions of weighted edges.

Network analysis involves predictions over nodes and edges of a graph. In a typical classification task, we are mainly interested in predicting the most probable labels of nodes [14]. For example, in a protein-protein interaction network, we are interested in predicting functional labels of proteins [13, 16], in a social network, we are interested in predicting groups of users. Link prediction (associated edges) is useful in a wide variety of domains; for instance, in genomics, it helps us to identify the novel interactions among genes, and in social networks, it can detect real-world friends [5, 9, 15]. A supervised machine learning algorithm requires a set of informative and independent features. In prediction problems, one has to generate feature vector representations for the nodes and edges. A typical solution is to create domain-specific features based on expert knowledge. However, such features are designed for specific tasks and do not generalize across different prediction tasks.

An alternative approach is to learn feature representation through unsupervised algorithms [10]. The challenge in learning feature representation is to define an objective function, which can balance computational efficiency and predictive accuracy. On one side, one could aim to find a feature representation that optimizes the performance of a prediction task. However, this leads to high training time complexity due to the number of parameters that need to be estimated. At the other side, the objective function can be defined to be independent of the prediction task that requires an unsupervised approach for learning the representations. The latter makes the optimization computationally efficient which results in task-independent features that closely match task-specific approaches for predictive accuracy [11, 12, 39].

Learning network representations poses the following problems: (1) Information-preservation: network embedding is required to preserve the network structural information. However, the underlying structure of the network is very complex [18]. (2) High non-linearity: the structure of the network is highly non-linear [17]. (3) Sparsity: Many real-world networks are often so sparse that it is not enough to reach a satisfactory performance [1].

To overcome these problems, DeepWalk [1] transforms a graph structure into a sample collection of linear sequences consisting of vertices using uniform sampling (known as a truncated random walk). The skip-gram model [19], originally

designed for learning word representations from linear sequences, can also be used to learn the representations of vertices from such samples. Although this method is empirically effective to find the latent representation for the graph with unweighted edges.

Although the conventional DeepWalk (C-DW) model generates efficient latent representations of the graph, it does not allow the graph to have weighted edges. In this paper, we propose DeepWalk for Weighted Graphs (DW-WG), an enhanced model for learning graph representations for knowledge management with the graph having weighted edges. Such learned global representations can be used as features for further processing of the graph. We present a formal treatment of this model, showing the distinctions between our model and the C-DW model. We demonstrate the effectiveness of the learned representations by experimenting with real-world problems. In all such tasks, DW-WG performs better than C-DW.

The remaining segments of this paper are organized as below. Section 2 describes the related work. The subsequent section proposes our model. We present the experiments and results in Section 4. The last section outlines the summary of this work.

II. Related Work

Feature engineering is being studied by the machine learning community for various purposes. In networks, the mechanism for generating features for nodes is based on feature extraction techniques that involve manually designed features, based on network properties [20, 21, 40].

Unsupervised feature learning approaches use the spectral properties of various matrix representations of graphs.

Current classical approaches to learning low dimensional graph representations are multidimensional scaling (MDS) [4], IsoMap [6], LLE [23], and Laplacian Eigenmaps [2]. Recently, Tang et al. [24] presented methods for learning latent representational vectors of the graphs which can then be applied to social network classification. Ahmed et al. [22] proposed a graph factorization method, which used stochastic gradient descent to optimize matrices from large graphs.

Recent advancements in representational learning for natural language processing opened new ways for feature learning of discrete objects such as words.

In particular, the skip-gram model [11] aims to learn continuous feature representations for words by optimizing a neighborhood preserving likelihood objective.

Representation learning aims to create readily usable data representations from raw data. In contrast to manual feature engineering, representation learning seeks to do this automatically [1, 26, 27, 28]. Often, data is not formatted for specific algorithms to detect features. As a result, representation learning algorithms aim to provide a general representation, which is usually in the form of latent feature vectors that may work for a variety of datasets and data mining applications. Machine learning ties in well to achieve this goal.

Current research on representation learning has produced many representation learning models [1, 26, 29, 30, 31, 32, 33, 34, 41]. With each new model comes with innovative new ideas that solve some feature generation problem.

Three instances of these models are as below:

1) DeepWalk: The DeepWalk algorithm [1] is a graph embedding algorithm. This algorithm is designed to create node embeddings for nodes in a graph network. The embeddings are generated by taking n random walks from the center node and applying the skip-gram algorithm [11] on the walks which boils down to predicting the features for all the nodes.

2) RDF2Vec: RDF2Vec [35] is an approach similar to DeepWalk that aims to create graph embeddings from a Resource Description Framework (RDF). In contrast to DeepWalk, RDF2Vec, uses two different strategies to extract subgraphs: Graph walks and Weisfeiler-Lehman subtree RDF graph kernels. These two strategies are similar to DeepWalk, but the authors note that their main difference is that their work uses a directed network instead of an undirected network.

3) GraphSAGE: Another graph embedding approach is called GraphSAGE [36]. The authors propose a method for inductive node embedding. In their approach, a node's neighborhood is aggregated by a set of trainable aggregator functions. Their approach generalizes using the aggregator functions that can be applied to unseen nodes as well as neighboring nodes in predicting the time, producing an embedding for the node and its context.

Even though all of the above algorithms are capable of generating a latent representation of the graph, none of them have considered a graph with weighted edges. We extended C-DW algorithm that can generate a latent representation for the graph having weighted edges. Further, we evaluated the DW-WG model for a wide range of parameters on the weighted graph.

III. DeepWalk for Weighted Graph

In this section, first, we define the problem, and then we introduce the proposed DW-WG model.

A. Problem Definition

We consider the problem of classifying nodes of a weighted graph into clusters of nodes as categories. Let the weighted graph G = (V, E), where V is the set of nodes of the network, and E is the set of all weighted edges. The problem is to find significant features that can represent relationships among the nodes of the network.

In a traditional machine learning classification setting, we learn a hypothesis that maps elements of feature set X to the labels set Y. C-DW approach generates features of an unweighted graph G. In DW-WG, we consider a weighted graph with weights on each edge to generate a realistic representation of features.

B. The Proposed Model

The proposed model DW-WG consists of two phases namely random-walk and word2vec. In the random-walk phase, we sample the weighted graph along the edges to generate a path that is a sequence of nodes visited in the graph.

We use the following three user parameters to generate random-walks:
NW: Number of walks for each node.
Walk_length: Number of edges on each random-walk.
Distance: Minimum distance (sum of edge weights along the walk) to be covered for each random alk.
Subsequently, the neural analysis of word2vec model accepts the random-walks generated in the previous phase and generates feature vectors for the graph as illustrated in Figure 1.

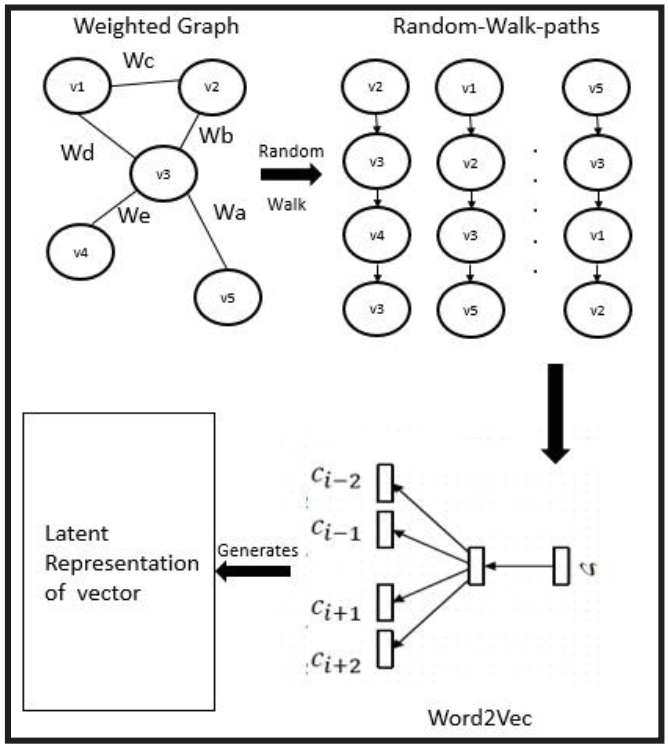

Figure 1. DW-WG (Deepwalk for Weighted Graph)

```
procedure DW-WG (G, pl, pd, start_node, walk)
input:
   G = (V, E): weighted edge graph
   pl: path_length; // no. of nodes along the path
   pd: path_distance; // total edge weights along the path
   start_node: start node for the random walk generation
output:
   walk: random walk path for the start node
begin
   walk =[ start_node ];
   walk_nodes = 0;
   current_distance = 0;
   while ((current_distance < pd) or (walk_nodes < pl))
      select a neighbor node randomly;
      append the current node to the walk;
      add the current edge weight to current_distance;
      increment walk_nodes;
      update the current node to the next random node;
   return walk;
end
```

Algorithm 1. Deepwalk path generation

1) Random walk: Given a graph with weighted edges and a starting node, it selects one of its neighbor nodes at random, and moves to their neighbor node; then it selects a neighbor of this node at random and moves to it, etc. The random sequence of nodes selected this way is a random walk on the graph. A random walk is a stochastic or random process which generates a path on the graph that consists of a sequence of randomly selected weighted edges. For instance, the path sequence of a molecule in a liquid or a gas, the price of a product, and the weather fluctuations can all be approximated by random walk models.

2) Word2Vec: The word2vec model and its application by Mikolov et al. [11] have attracted a great amount of attention in recent years. The vector representations of words (i.e., nodes of the semantic graph), learned by word2vec model from the random-walks, have been shown to carry semantic meanings and are useful in various NLP (Natural Language Processing) tasks. Word2vec is a predictive neural analysis model for feature generation from the random-walks. It has two different approaches namely the Continuous Bag-of-Words (CBOW) and the skip-gram. Algorithmically, these approaches are similar, except that CBOW predicts target words (e.g. 'pond') from source context words ('the water runs downhill to the'), while the skip-gram does the inverse and predicts source context-words from the target words. This inversion might seem like an arbitrary choice, but statistically, it has a similar effect as that of CBOW which approximates over a lot of the distributional information (by treating an entire context as one observation). In this work, after generating random walks, we use the skip-gram approach of the word2vec model to generate feature representations of random-walks. In the skip-gram model, focus word (the starting node of the random-walk) is used for single input layer, and target context words (nodes related to the focus node) are used for the output layer. The training objective is to minimize the sum of prediction errors across all context words (nodes along all random-walks) in the output layer.

3) Parallelizability: To identify the most suitable set of feature vectors for the weighted graph that matches the ground truth result, first we need to generate several sets of feature vectors for a different range of values for all the parameters of the DW-WG. This process requires the execution of the model with different sets of parameter values. Since this model is highly scalable, we implemented this process on multiple cores with multi-threading.

We generate one set of feature vectors for each combination of parameter values. To identify the best set of feature vectors, we employ supervised machine learning algorithms only on the ground truth nodes of feature vectors. The set of feature vectors that yields the highest accuracy in this approach is chosen as the best set of feature vectors. The nodes of the corresponding entire set of feature vectors (includes both ground truth as well as non-ground truth nodes) are clustered using an unsupervised machine learning algorithm to establish the clustering association among the

non-ground truth nodes. This method reduces the search complexity of fake review of a node (reviewer account) to only the associated cluster of nodes instead of the entire set of nodes.

IV. Experiments and Results

We evaluate our proposed method on two different real-world datasets and applications. The experimental results demonstrate significant improvements of the DW-WG over the C-DW.

A. Datasets

To comprehensively evaluate the effectiveness of the feature representations, we use two networked datasets with three real-world metrics namely accuracy for classification, silhouette and average measures for clustering efficiency. For both datasets, we measure the performance on each metric. The detailed descriptions of the datasets are listed as follows.

Bitcoin OTC+Alpha Trust Network [37]: This dataset represents who-trusts-whom relationships among a network of people who trade using Bitcoin on a platform called Bitcoin OTC and Bitcoin Alpha. As the Bitcoin users are unnamed, to prevent transactions with fraudulent and risky users we need to maintain a record of users' reputation. Bitcoin OTC members rate other members on a range of -10 (total distrust) to +10 (complete trust) with an increment of 1. From this dataset, the best set of feature vectors are generated using our DW-WG model. The labels for the clusters of the best set of feature vectors are of the range from -10 to +10.

Google play store app review network data: This dataset describes the relationship between the app and the reviewer. We have review data for 640 apps from Google Play Store with a co-review graph for each app. Each vertex of a co-review graph represents a reviewer account, and each edge indicates the number of apps co-reviewed by the reviewer accounts of the associated vertices. We integrated all 640 co-review graphs into one unified graph. For impersonation, some individual reviewers use several fake reviewer accounts. Each fraudulent individual reviewer corresponds to a set of fake reviewer accounts. The ground truth dataset consists of 2207 nodes (reviewer accounts) and these nodes correspond to 23 clusters (individual reviewers). For the entire dataset, the best set of feature vectors is generated using our DW-WG model. The clusters for the ground truth section of this set of feature vectors are of the values from 1 to 23.

For evaluation, we conduct experiments on both weighted and unweighted, sparse and dense, small and large networks. Therefore, these datasets can comprehensively reflect the characteristics of feature generations. The detailed statistics of the datasets are presented in Table I.

TABLE I. Dataset statistics

Dataset	#(V)	#(E)
Google Play Store	38,123	3,572,409
Bitcoin OTC+ALPHA trust	9,664	59,778

B. Experiment results

In our experiment, we evaluate classification accuracy and clustering efficiency on the datasets. We split data for training and testing using k-fold cross-validation.

1) Classification Accuracy: It is the ratio of the number of correct predictions over the total number of predictions made.

We trained three most widely used machine learning algorithms (Support Vector Machine, Random-Forest, and K-Nearest Neighbor) on the following datasets and the results are as follows:

a) Google Play Store dataset:

TABLE II. Classification accuracy for google dataset

Algorithm	SVM	Random-Forest	K-NN
DW-WG	87%	85%	81%
C-DW	80%	79%	76%

b) Bitcoin OTC+Alpha dataset:

TABLE III. Classification accuracy for bitcoin dataset

Algorithm	SVM	Random-Forest	K-NN
DW-WG	89%	84%	84%
C-DW	83%	80%	79%

In all the above three machine learning algorithms, the enhanced deepwalk model (DW-WG) has shown improved performance over the conventional deepwalk (C-DW) model.

2) Clustering approach: We applied the K-Means clustering algorithm on both datasets. In the K-Means algorithm, we specify a value for the parameter K (for the number of clusters to be formed). Since there is no well-known method to find the best value of K, we use the following two different approaches to establish the most suitable K value.

a) Silhouette Measure: The Silhouette Coefficient is defined for each data point and is composed of two scores:

$\mu_{in}(X_i)$: The mean distance between a data point and all other points in the same cluster.

$\mu_{out}(X_i)$: The mean distance between a data point and all other points in the next nearest cluster.

The Silhouette Coefficient Si for a single data point i is given as:

$$S_i = \frac{\mu_{out}(X_i) - \mu_{in}(X_i)}{\max\{\mu_{out}(X_i),\ \mu_{in}(X_i)\}} \qquad (1)$$

The silhouette coefficient for the entire dataset is the mean silhouette coefficient of all data points of the data file.

i) Google Play Store Dataset: We first calculate the silhouette coefficient for ground truth data points with the number of clusters ranging from 20 to 36, to cross-check the correctness of this method. We find that the maximum

silhouette score for DW-WG corresponds to 23 clusters and this matches with the number of clusters of the ground truth data set (23 clusters), whereas the maximum silhouette for C-DW is 28 and that does not match with the ground truth clusters as illustrated in Figure 2.

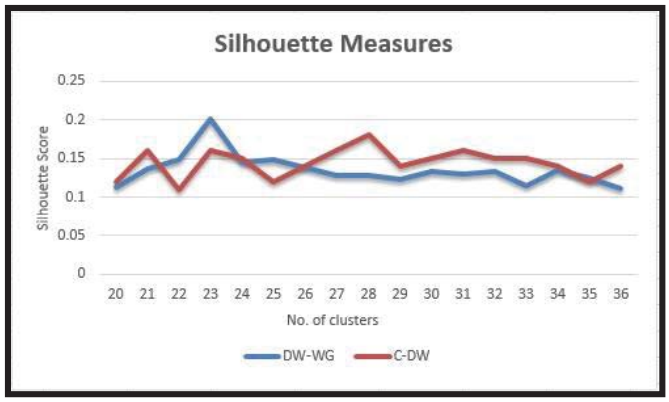

Figure 2. Silhouette Measures for Ground Truth Clusters

Subsequently, we compute the silhouette coefficient for all data points of the data set with the number of clusters (K value) ranging from 400 to 700. For the given data set, the optimal number of clusters for DW-WG is 430 whereas for the C-DW is 520 as shown in Figure 3.

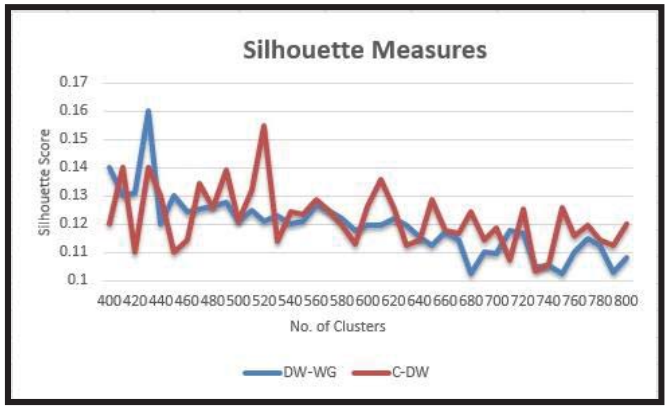

Figure 3. Silhouette Measures for the entire dataset

As the number of clusters found by DW-WG (430) for the ground truth data was more accurate than the number of clusters found by the C-DW (520), the number of clusters formed by DW-WG for the entire dataset will be the optimal K-value for clustering.

ii) Bitcoin OTC and Alpha Trust Network Dataset: This dataset is the merger of two clusters of data namely Bitcoin OTC and Alpha. We perform Silhouette measures using both DW-WG and C-DW, and we find that the number of clusters for the highest silhouette score for DW-WG is two that matches with the actual number of clusters, while the highest silhouette score for C-DW corresponds to three clusters as depicted in Figure 4.

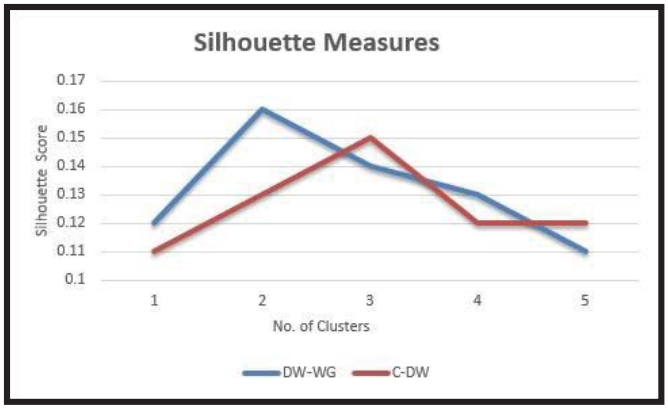

Figure 4. Silhouette Measures

b) Average Measures: For further consolidation of the evaluation results, we perform an additional approach to arrive at the optimal K value. In this approach, we use the mean of the following measures to find the optimal number of clusters (K value):

- Homogeneous
- Completeness
- Silhouette
- Adjusted Rand Index

i) Google Play Store Dataset: We find the optimal number of clusters for the entire dataset by DW-WG (430) that matches with the number of clusters found in Silhouette Measures approach. Also, the number of clusters by C-DW (490) that does not match with the corresponding result in Silhouette Measures as illustrated in Figure 5.

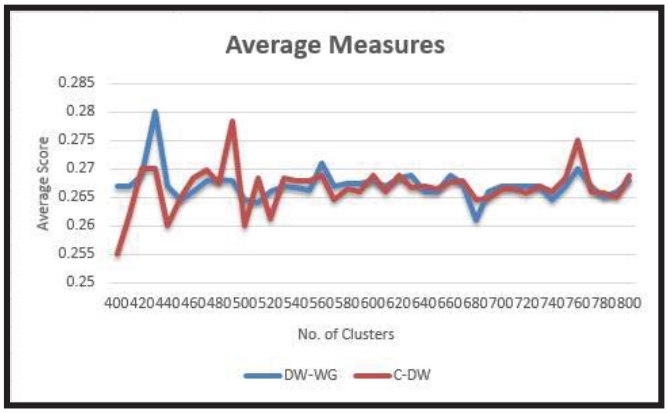

Figure 5. Average Measures for Google Play Store Dataset

ii) Bitcoin OTC and Alpha Trust Network Dataset: We find the optimal number of clusters for the dataset by DW-WG (2 clusters) that matches with the same number of clusters found in Silhouette Measures approach. Also, the number of clusters by C-DW (4 clusters) that does not match

with the corresponding result in Silhouette Measures as illustrated in Figure 6.

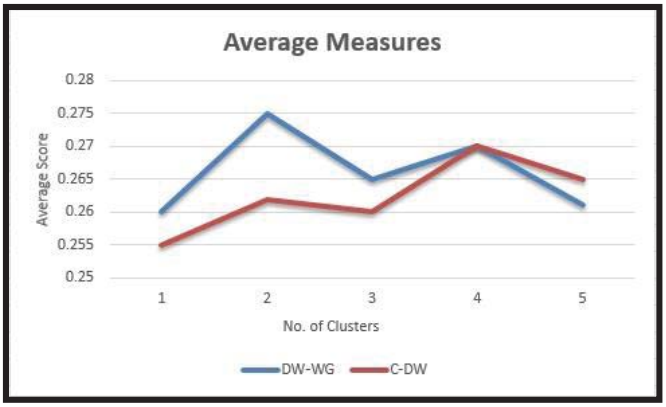

Figure 6. Average Measures for Bitcoin OTC+ Alpha Dataset

From our experiment, we determine the optimal number of clusters (K value) for K-Means algorithm as 430 clusters for Google Play Store Dataset and 2 clusters for Bitcoin OTC and Alpha Trust Network Dataset.

V. CONCLUSION

In this paper, we proposed an enhanced deepwalk model for a weighted graph (DW-WG), to perform network feature representations. Specifically, to capture the high-level representation of the vertices, we modified the existing deepwalk model (C-DW), by allowing the random-walk to include weights for edges. To speed up the feature generation for complex graphs, we introduced a parallel code that runs on multiple cores with multi-threading. Empirically, we evaluated the generated network feature representations in two network datasets with different metrics.

We find DW-WG yields improved classification accuracy compared to C-DW for both datasets. Similarly, DW-WG computes the optimal number of clusters for the K-Means algorithm more accurately than the C-DW for both datasets. Additionally, the optimal number of clusters value is more precise for DW-WG on both Silhouette Measures as well as Average Measures approaches compared to C-DW.

The experimental results demonstrate the significance of the proposed deepwalk model for weighted graphs (DW-WG) in terms of classification accuracy and clustering efficiency compared to the conventional deepwalk model (C-DW). The source code for DW-WG [38] is written in Python. We propose to continue this research on system call dataset collected through virtual machine introspection at hypervisor level to classify benign processes from malwares in a virtualized environment.

ACKNOWLEDGMENT

We want to thank Bogdan Carbunar at Florida International University for introducing the co-reviewers problem and sharing the Google Play Store Dataset. Also, we are thankful to the Department of Defense – Test Resource Management Center (Grant# W900KK-16-C-0043) for supporting this research.

REFERENCES

[1] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: online learning of social representations," Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, August 24-27, 2014, New York, New York, USA.

[2] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in NIPS, vol. 14, pp. 585-591, 2001.

[3] S. Bhagat, G. Cormode, and S. Muthukrishnan, "Node classification in social networks," in Social Network Data Analytics, pp. 115-148. Springer, 2011.

[4] T. F. Cox and M. A. Cox, Multidimensional scaling. CRC Press, 2000.

[5] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," Journal of the American society for information science and technology, vol. 58, pp.1019-1031, 2007.

[6] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," Science, vol. 290, pp. 2319-2323, 2000.

[7] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne," Journal of Machine Learning Research, vol. 9, pp. 2579-2605, 2008.

[8] X. Yu, X. Ren, Y. Sun, Q. Gu, B. Sturt, U. Khandelwal, B. Norick, and J. Han, "Personalized entity recommendation: A heterogeneous information network approach," in Proceedings of the 7th ACM international conference on Web search and data mining, pp. 283-292, 2014.

[9] L. Backstrom and J. Leskovec, "Supervised random walks: predicting and recommending links in social networks," in WSDM, 2011.

[10] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," IEEE TPAMI, vol. 35, pp. 1798–1828, 2013.

[11] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in ICLR, 2013.

[12] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global vectors for word representation," in EMNLP, 2014.

[13] P. Radivojac, W. T. Clark, T. R. Oron, A. M. Schnoes, T. Wittkop, A. Sokolov, K. Graim, C. Funk, Verspoor, et al., "A large-scale evaluation of computational protein function prediction," Nature Methods, vol. 10, pp. 221–227, 2013.

[14] G. Tsoumakas and I. Katakis, "Multi-label classification: An overview," Dept. of Informatics, Aristotle University of Thessaloniki, Greece, 2006.

[15] A. Vazquez, A. Flammini, A. Maritan, and A. Vespignani, "Global protein function prediction from protein-protein interaction networks," Nature biotechnology, vol. 21, pp. 697–700, 2003.

[16] S.-H. Yang, B. Long, A. Smola, N. Sadagopan, Z. Zheng, and H. Zha , "Like like alike: joint friendship and interest propagation in social networks," in WWW, 2011.

[17] D. Luo, F. Nie, H. Huang, and C. H. Ding, "Cauchy graph embedding," in Proceedings of the 28th International Conference on Machine Learning (ICML-11), pages 553–560, 2011.

[18] B. Shaw and T. Jebara, "Structure preserving embedding," in Proceedings of the 26th Annual International Conference on Machine Learning, pp. 937–944. ACM, 2009.

[19] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in NIPS, pp. 3111-3119, 2013.

[20] B. Gallagher and T. Eliassi-Rad, "Leveraging label-independent features for classification in sparsely labeled networks: An empirical study," in Lecture Notes in Computer Science: Advances in Social Network Mining and Analysis. Springer, 2009.

[21] K. Henderson, B. Gallagher, L. Li, L. Akoglu, T. Eliassi-Rad, H. Tong, and C. Faloutsos, "It's who you know: graph mining using recursive structural features," in KDD, 2011.

[22] A. Ahmed, N. Shervashidze, S. Narayanamurthy, V. Josifovski, and A. J. Smola, "Distributed large-scale natural graph factorization," in WWW, pp. 37-48. International World Wide Web Conferences Steering Committee, 2013.

[23] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," Science, vol. 290, pp. 2323-2326, 2000.

[24] L. Tang and H. Liu, "Relational learning via latent social dimensions," in SIGKDD, pp. 817-826. ACM, 2009.

[25] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," Science, vol. 290, pp. 2319-2323, 2000.

[26] T.-Y. Fu, W.-C. Lee, and Z. Lei, "HIN2Vec: Explore Meta-paths in Heterogeneous Information Networks for Representation Learning," in Proceedings of the 2017 ACM on Conference on Information and Knowledge Management. ACM, 2017.

[27] H. Oh Song, Y. Xiang, S. Jegelka, and S. Savarese, "Deep Metric Learning via Lifted Structured Feature Embedding," arXiv, 19 Nov. 2015.

[28] R. Feng, Y. Yang, H. Wenjie, W. Fei, and Z. Yueting, "Representation Learning for Scale-free Networks," AAAI, 2018.

[29] A. Grover and J. Leskovec, "node2vec: Scalable Feature Learning for Networks," in Proceedings of 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 855-864, 2016.

[30] J. Tang, M. Qu, M. Wang, Z. Ming, J. Yan, and M. Qiaozhu, "LINE: Large-scale Information Network Embedding," in Proceedings of International Conference on World Wide Web (WWW 2015). ACM, pp. 1067–1077.

[31] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Chang, "Network Representation Learning with Rich Text Information," in Proceedings of IJCAI'15 24th International Conference on Artificial Intelligence pp. 2111-2117.

[32] S. Chang, W. Han, J. Tang, G.-J. Qi, C. Aggarwal, and T. Huang, "Heterogeneous Network Embedding via Deep Architectures," in the Proceedings of KDD '15 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 119-128.

[33] Y. Dong, N. Chawla, and A. Swami, "metapath2vec: Scalable Representation Learning for Heterogeneous Networks," in the Proceedings of KDD '17 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 135-144.

[34] L. Ribeiro, P. Saverese, and D. Figueiredo, "struc2vec: Learning Node Representations from Structural Identity," KDD, 2017.

[35] P. Ristoski and H. Paulheim, "Rdf2vec: Rdf graph embeddings for data mining," in International Semantic Web Conference, pp. 498-514. Springer, 2016.

[36] W. L. Hamilton, R. Ying, and J. Leskovec, "GraphSage: Inductive representation learning on large graphs," arXiv preprint arXiv:1706.02216, 2017.

[37] S. Kumar, F. Spezzano, V. S. Subrahmanian, and C. Faloutsos, "Edge Weight Prediction in Weighted Signed Networks," IEEE International Conference on Data Mining (ICDM), 2016.

[38] https://users.cs.fiu.edu/~prabakar/DW-WG/

[39] Soni, J., Prabakar, N. and Kim, J-H. (2017) "Prediction of Component Failures of Telepresence Robot with Temporal Data," 30th Florida Conference on Recent Advances in Robotics.

[40] Soni, J., Prabakar, N. and Upadhyay, H. (2019) "Deep Learning approach to detect malicious attacks at system level : poster", Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks. ACM, pp. 314-315.

[41] Soni, J. and Prabakar N. (2018) "Effective Machine Learning Approach to Detect Groups of Fake Reviewers," " Proceedings of the 14th International Conference on Data Science (ICDATA'18), Las Vegas, NV, 3-9, 2018.