BalNode2Vec: Balanced Random Walk based Versatile Feature Learning for Networks

Amirreza Salamat
Department of ECE
IUPUI
Indianapolis, Indiana
asalamat@iu.edu

Xiao Luo
Department of CIT
IUPUI
Indianapolis, Indiana
luo25@iupui.edu

Ali Jafari
Department of CIT
IUPUI
Indianapolis, Indiana
jafari@iupui.edu

***Abstract*—Research on social networks and understanding the interactions of the users can be modeled as a task of graph mining, such as predicting nodes and edges in networks. The challenges of building the graph representation include engineering features used by learning algorithms. Recent research in representation learning can automate the prediction by learning the features themselves. Many types of research have performed graph sampling using random walks or it's derivatives. However, the random walk sometimes can not represent the features of the graph accurately enough. In this research, we propose BalNode2Vec – a new sampling algorithm for learning feature representations for nodes in networks by using balanced random walks. We define a notion of a nodes network neighborhood and design a balanced random walk procedure, which adapts to the graph topology. We show that through exploring the graph through a balanced random walk can generate richer representations. Efficacy of BalNode2vec over existing state-of-the-art techniques on link prediction is demonstrated by using several real-world networks from different domains.**

***Index Terms*—Node Embedding, Representation Learning, Graph Representation**

I. Introduction

Graph networks play an essential role in social networks, genomics, and recommender systems [41] [38]. The user's information, interactions, characteristic features, and role in the network can be modeled as a graph. The most recent research in graph networks investigated approaches to generate graph embedding, which learns a mapping from a network to a vector space while preserving relevant network properties. DeepWalk [13], LINE [2], and node2vec [6] are the graph embedding algorithms used in various domains, such as assessing protein interactions or predicting genome interactions. Graph embedding not only improves the efficiency and accuracy of user profiling but also enables vector analysis techniques to be applied. So that data can be transformed with less computational costs. The applications built upon the graph network, such as the recommender system can be more scalable and also be used in real-time applications.

The Node2Vec is built based on the idea of word embedding generation techniques, such as the Skip Gram model [16]. The basic theory is that performing short random walks on a scale-free graph can generate node frequencies that closely follows Zipf's law [13], which is similar to the word distribution of a corpus. Although the random walk provides a means for sampling the graph, it may not represent the features of the graph accurately, especially with a directed graph. The first issue is that the initial starting node is chosen at random, meaning that the nodes with lower degrees are treated equally with the higher degree nodes. As the length of the random walk increases, the walks bias towards nodes with more in-degrees and moves away from the nodes with fewer in-degrees, which causes those nodes to be underrepresented while the nodes with more in-degrees are over-represented. This behavior is illustrated by Figure1, which shows that node A might not be walked on very often, and node E appears on several walks. While the latter issue emerges in directed graphs, the former appears in both directed and undirected graphs.

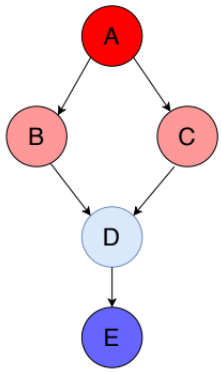

Fig. 1. A sample directed graph with nodes with various in-degrees to demonstrate that random walks on this graph tends to select bias towards the nodes with more direct or propagated in-degrees.

On the other hand, the notion of using random walks for sampling a graph was that in a scale-free network, the short random walks follow the power-law similar to the degree distribution [13]. Therefore, word embedding techniques like Skip Gram could be applied to the graph to create embeddings. However, with this type of approach, there are two issues:

- The sampling error increases as the size of the graph increases.
- The node distribution diverges from the power-law as the

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

length of the walk increases. Hence, more in-depth, more abstract features cannot be extracted from the network.

In this research, our objective is to develop a novel algorithm to prevent the sampling issues in calculating the probabilities of walking each node when using a random walk and comparing them with the actual degree distribution (which follows the power-law). Therefore, the sequences are more suitable for being used in an algorithm like the Skip Gram algorithm. Specifically, we propose a balanced random walk sampling algorithm – BalNode2Vec to generate the graph embedding. The algorithm is designed to walk on the nodes that are underrepresented using the random walk and put less emphasis on the nodes that are walked often. This new approach balances the node distribution and adjusts suit the Skip Gram optimizer. This proposed BalNode2Vec is able to produce relevant representations for large graphs without biasing on the nodes with more in-degrees and efficient for running and updating without advanced dedicated hardware, scalable to vast networks of hundreds of thousands of edges, adaptable to the variance of the data over time. We evaluate the BalNode2Vec on five different data sets. The comparison against the state-of-the-art algorithms demonstrates that BalNode2Vec gains the highest AUC scores on all data sets.

The rest of the paper is structured as follows. In Section II, we described the related work in graph embedding generation. We present the technical details of BalNode2Vec in Section III. In Section IV, we empirically evaluate BalNode2Vec on prediction tasks on various real-world networks and assess the parameter sensitivity. Section V presents our discussion about the BalNode2Vec framework and gives some directions for future work.

II. Related Work

Graph-based network analysis has caught the researchers' attention for many years. Several nonlinear dimension reduction techniques using spectral analysis were performed, but they were not suited for graph data sets and had statistical and performance shortcomings. Clustering techniques were also widely used as a means to categorize nodes based on various attributes, in [18] The largest eigenvalues of the Laplacian matrix were used for clustering the nodes. This algorithm implicitly assumed that the graph cuts would be enough for classification. GraRep generates vertex representations by explicitly computing successive powers of the random walk transition matrix and uses the SVD to reduce their dimensionality, and it's a very powerful baseline. However, the downside of this approach is that it is not scalable to larger networks.

Advances in word embedding were another breakthrough for the graph embedding algorithms, Noise Contrastive Estimation (NCE) loss [16]significantly improved the performance of word embedding algorithms. NCE postulates that a good model should be able to differentiate data from noise through the means of logistic regression. This model is highly efficient for performing language modeling. Skip Gram [16] was one of the algorithms that utilized NCE loss for optimizing a neighborhood preserving likelihood objective; this model is extremely efficient when dealing with vast corpora of words.

DeepWalk [13] introduced deep learning (unsupervised feature learning) [36] techniques that learn representations of graph vertices and by modeling a stream of random walks on the graph and feeding the generated strings into Skip Gram. DeepWalk achieved scalability and adaptability in large networks but was not configurable to the pattern that we wanted to capture from the graph.

LINE [2] is another embedding generation algorithm that embeds the graph into d dimensional vectors and learns $d/2$ embeddings using Breadth-First Search (BFS) and another $d/2$ using Depth First Search(DFS), this method loses several benefits of random walk which is explained in [6] and is not entirely configurable to different patterns of data.

In the paper of Node2Vec [6], it describes that the prediction task in graphs is comprised of two different aspects of the network, homophily and structural equivalence. Homophily hypothesis formulates the nodes which belong to the same community and have frequent interactions must be embedded together [20], while structural equivalence hypothesis [44] states that nodes with similar structural role should be embedded together. Node2vec [6] followed the same procedure as Deepwalk except that it used second-order random walks and introduced two new parameters, return parameter and in-out parameter, and claimed that these two parameters simulate the effects of BFS and DFS which represent structural equivalence and homophily respectively. This algorithm allowed the random walks to be configured based on the requirements, but it distorts the node frequencies that Negative Sampling needs to consider. This issue is elevated when dealing with graphs that have lower average edge per node. Also, in the same study, several link prediction algorithms were proposed to evaluate the performance of the model, these link prediction algorithms were compared in [19] [45] in terms of performance, the major shortcoming in all of these techniques has high computational efficiency.

The metapath2vec and metapath2vec++ were introduced in [33], which incorporated metapaths to model heterogeneous networks better, but the drawback of these methods is that domain knowledge is required to define these metapaths which may not be a viable option in many scenarios. Walklet [14] was another approach that captured multiscale node representation on graphs by sampling edges from higher powers of the adjacency matrix and as a consequence, skipping some nodes. The authors claimed that sampling from each order of the adjacency matrix captures a specific dimension of social interactions, but some deeper connections can only be extracted from a blend of different orders of the adjacency matrix. Additionally, the training was carried out by a MultiLayer Perceptron, which has performance issues on larger output classes.

Other deep learning models were also proposed [42] [35], which use architectures like Convolutional Neural Networks to perform node embedding. These algorithms have high computational costs, require complete or partial retraining after

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

the change, and are hard to scale to larger networks.

III. Representation Learning Framework

The main objective in language modeling is to estimate the likelihood of a specific sequence of words appearing in a corpus. Equivalently, we can perform a series of walks on a graph to turn them into a sequence of nodes. By that definition, our problem can be formally defined as estimating the probability of observing vertex v_i given all the previous vertices visited:

$$\Pr\left(v_i | \left(v_1, v_2, \cdots, v_{i-1}\right)\right) \tag{1}$$

Assuming our network graph is $G = (V, E)$, we define the mapping $\Phi = V \epsilon v \longrightarrow R^d$ as mentioned in [6]:

$$\Pr\left(v_i | \left(\Phi\left(v_1\right), \Phi\left(v_2\right), \cdots, \Phi\left(v_{i-1}\right)\right)\right) \tag{2}$$

Calculating the Equation 2 mentioned above is computationally expensive, so two assumptions are made to have the calculations more feasible. First, instead of using the context to predict a node, it uses one node to predict the context. Secondly, the context is composed of the adjacent nodes (both left and right) with any order, meaning that there are no differences between the adjacent nodes in the context. This simplifies our algorithm to Equation 3.

$$\underset{\Phi}{\text{maximize}} \quad \log \Pr\left(\left\{v_{i-w}, \cdots, v_{i+w}\right\} \setminus v_i | \Phi\left(v_i\right)\right) \tag{3}$$

In the following subsections, in subsection III-A, we describe how node sequences are generated using balanced random walks. Subsection III-C presents how the model learns the representations based on the input sequences.

A. Walk Prediction

As mentioned in the previous sections, node occurrences in random walks diverge from the degree distribution based on the graph topology. Initially, we calculate the frequency of each node in a random walk to adapt our walking strategy. The first order of the adjacency matrix of a graph shows the possible destinations after a single walk. Normalizing each row of this matrix shows the probability of moving to a node from a chosen source node. By summing over the column of this normalized adjacency matrix results in a vector, we can calculate the frequency of a destination node through a random walk of length 1.

The same process can be repeated for higher powers of the adjacency matrix, such as A^n, outputs a similar frequency vector except that the destination is reached after n walks. Through adding these vectors together, it shows the frequency of passing through each node in n walks.

Algorithm 1 Node frequency measurement in a random walk

Input: Graph G=(V, E, W),
Output: f node frequencies
1: A=Adjacency Matrix(G)
2: A_{norm} = normalize(A) over rows
3: A^n = normalize(A) over rows
4: $W = normalizeddegree_{histogram}(G)$
5: **for** $i = 1$ to walk length **do**
6: A^n = $matmul(A^n, A_{norm})$
7: W = $W + \sum_{<i>} A^n_{i,j}$
8: **end for**
9: W_{norm} = normalize(W) over rows
10: **return** W_{norm}

The adjacency matrix of a graph shows possible transitions from each node to another. By adding all the values in a column, we can acquire the number of times that a node was walked on in a single step. The second order of adjacency matrix has a similar effect except that the values are calculated for two steps. Continuing for higher orders of the adjacency matrix provides us with a simulation of the walker trajectory in a random walk. Finally, summing over all the values for each node in every order provides us with an estimate of the number of times that a specific node is walked on.

Having the estimation for each specific node, we can observe that the number of times each node is walked on is different from the degree of the node. Although this calculation adds some overhead to the random walk algorithm, the calculation is done only once for the graph and the sparsity of the adjacency matrix to keep this overhead to the minimum.

To choose the initial node, the probability of initially choosing a node to walk on is proportional to their degree, this ensures that the initial values follow the Zipf distribution. The algorithm 1 measures the frequency of each node in a random walk using the graph adjacency matrix.

Assuming the graph has E edges and V nodes, the adjacency matrix is $O(E)$ space complexity because of the sparse nature of it, however, the A^n is a space complexity between $O(E)$ and $O(V^2)$ and this complexity gravitates towards the latter since the matrix becomes dense as n increases. The space required for storing the frequencies, on the other hand, is $O(V)$ and therefore, comparatively trivial. The process of matrix multiplication, addition and normalization altogether have a time complexity of $O(N)$ with N being the number of non-zero elements in the matrix. So for A^n, the time maximum complexity is $O(V^2)$.

B. Balanced Random Walk

After finding the frequency of the nodes in a random walk, these frequencies are compared with the degree distribution to modify the node preferences when walking. The nodes with a higher frequency compared to their degree are at a penalty; conversely, the nodes with a lower frequency compared to their degree are preferred during the walk. The algorithm 2 performs the comparisons and carries out the balanced walk.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

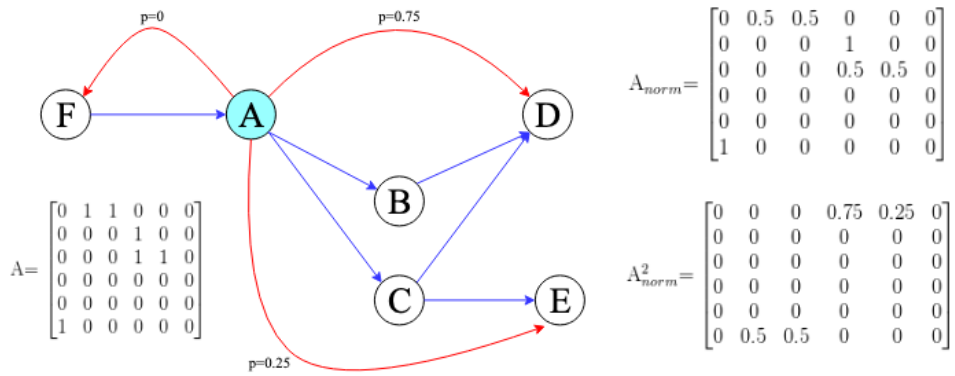

Fig. 2. An example of a simple graph with walk probabilities extracted from the normalized adjacency matrix.

Algorithm 2 Balanced Random Walk

Input: Graph G=(V, E, W), W_{norm}
Output: W walks

for all nodes **do**
  $coefficient[node] = degree[node]/W[node]$
end for
for $i = 1$ to number of walks **do**
  $weight[node] = degree[node]$
  initial node = choose random node with given weight
  $weight[node] = coefficient[node]$
  walk = random weighted walk from node with the given weight
  append walk to walks
end for
return walks

The initial value of A_0 ensures that the short walks are close to the power-law while the following weights ensure that the walks do not diverge from this initial value. It is also worth noting that random walks are random, and the variance of the node frequencies increase as the walk becomes longer. Since our objective is to ensure the walks follow the power-law on average, the balanced walks produce a more consistent output than the random walk.

C. Learning Features

After performing the balanced random walks, the strings of nodes have been created. First, these strings are shuffled to randomize their order, and then they fed to the Skip Gram model to extract the representations of the nodes. The training objective of the original Skip Gram model is to find word representations that are useful for predicting the surrounding words in a sentence or a document. Applying full softmax in training is not efficient for large networks because it evaluates all the outputs to obtain the results; therefore, other methods have replaced it.

Hierarchical softmax is a computationally efficient approximation of softmax. Instead of evaluating all the output nodes, hierarchical softmax only evaluates the log base 2 of the outputs, resulting in significant computational gains. Noise Contractive Estimate (NCE) loss is an alternative to hierarchical softmax. NCE loss reduces the language model estimation problem to the problem of estimating the parameters of a probabilistic binary classifier that uses the same parameters to distinguish samples from the empirical distribution from samples generated by the noise distribution. Essentially, this is more computationally attainable in large networks than softmax since it subsamples frequent words instead of considering all the words in the vocabulary. The Skip Gram model is concerned with learning high-quality vector representations, so we are free to simplify NCE as long as we maintain the vector representation quality. The negative sampling algorithm is then defined as:

$$\log \sigma\left({v'_{n_O}}^{\top} v_{n_I}\right) + \sum_{i=1}^{k} \mathbb{E}_{n_i \sim P_n(n)} \left[\log \sigma\left(-{v'_{n_i}}^{\top} v_{n_I}\right)\right] \quad (4)$$

Where P_n is the noise distribution, $n_1, n_2, ..., n_i$ is the sequence of nodes and $\sigma(x) = 1/(1 + exp(x))$. In language modeling, in very large corpora, the most frequent words can easily occur hundreds of millions of times (e.g., in, the, and a). Such words usually provide less information value than rare words. The same phenomenon happens in social networks, a few users possess most of the edges in the network; therefore, they don't contain valuable information in most cases. To counter the imbalance between the rare and frequent nodes, assuming $f(n_i)$ is the frequency of the i'th node, a simple subsampling approach was used which the nodes in the training set is discarded by:

$$P\left(n_i\right) = 1 - \sqrt{\frac{t}{f\left(n_i\right)}} \quad (5)$$

The context size is one of the primary parameters of the training, nodes in the context window can be paired together and fed into the network. Having a larger window allows our model to understand the more distant features in our network better. But the larger the context size, the more data it requires to train the model. It may not be necessary for most of the networks.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

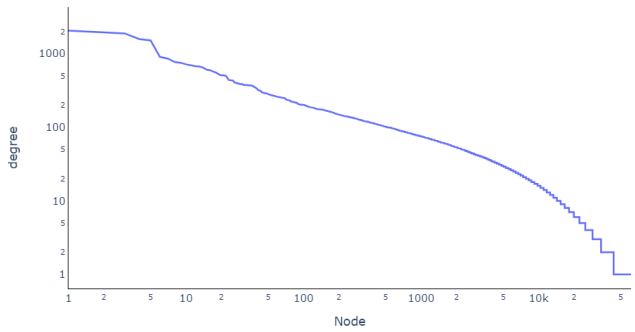

Fig. 3. Node frequency in CourseNetworking user-follower network in log-log scale.

TABLE I
BINARY OPERATORS FOR LEARNING EDGE FEATURES FROM NODE EMBEDDINGS.

Operator	Symbol	Definition
Average	$\oplus$	$[f(u) \oplus f(v)]_i = \frac{f_i(u)+f_i(v)}{2}$
Hadamard	$\odot$	$[f(u) \odot f(v)]_i = f_i(u) * f_i(v)$
Weighted-L1	$\|\cdot\|_{\overline{1}}$	$\|f(u) \cdot f(v)\|_{\overline{1}i} = \|f_i(u) - f_i(v)\|$
Weighted-L2	$\|\cdot\|_{\overline{2}}$	$\|f(u) \cdot f(v)\|_{2i} = \|f_i(u) - f_i(v)\|^2$

In a directed graph, each edge contains information about the directions too, therefore, the direction of the context is very helpful and depends on the type of data under study.

IV. Experimental Results

In this section, we provide the details of our datasets, then we evaluate the walking strategy, compare the performance of the whole model against other state-of-the-art algorithms, and evaluate the parameter sensitivity of the proposal BalNode2Vec. The experimental results are repeated for 10 random seed initialization and are statistically significant with a p value of less than 1%.

A. Dataset

We use five different data sets in this research, which 4 of them are publicly available. One of the data sets is generated from an academic and social networking site – CourseNetworking [43]. The user-follower network of the CourseNetworking is extracted from this academic and social networking site to illustrate the performance of the BalNode2Vec. There are 61151 nodes and 313923 edges. The nodes of this graph represent users, and an edge between a source and target node represents the follower and followee, respectively. The distribution of the edges closely follows the pattern of Zipf degree distribution, as shown in Figure 3. Therefore, the network graph is scale-free so that the BalNode2Vec model can be used to learn the representations of this network. This network consists of nodes with different "roles" like student or teacher, which means that a large number of source and destination nodes exist in this network.

The other data sets used in this research include the following:

- Reddit Hyperlink [39]: The hyperlink network represents the directed connections between two subreddits (a subreddit is a community on Reddit). The network is extracted from publicly available Reddit data of 2.5 years from Jan 2014 to April 2017. There are 55,863 nodes and 858,490 edges.
- Protein-Protein Interactions (PPI) [38]: In the PPI network for Homo Sapiens, nodes represent proteins, and an edge indicates a biological interaction between a pair of proteins. The network has 19,706 nodes and 390,633 edges.
- arXiv ASTRO-PH [39]: This is a collaboration network generated from papers submitted to the e-print arXiv where nodes represent scientists, and an edge is present between two scientists if they have collaborated in a paper. The network has 18,722 nodes and 198,110 edges.
- Facebook [39]: In the Facebook network, nodes represent users, and edges represent a friendship relation between any two users. The network has 4,039 nodes and 88,234 edges.

B. Walking Strategy

Figure 4 compares the node distribution by using the random walk and a balanced walk. As shown from the results, the balanced walk reduces the sampling error and generates samples that follow the power-law more closely and, as a result, produces a sequence that is more suitable for applying the Skip Gram algorithm.

To numerically measure the difference between the node frequencies in a walk, and the degree distribution, which is the target distribution, we used the statistical distance methods. The Jensen-Shannon divergence is used in this evaluation. The results of figure 5 show that the predictions have successfully managed to balance the node representations during the walk. Because of the initial condition of choosing nodes based on their degree, the BalNode2Vec has a minimal error in very short walks and even in longer walks, it manages reduce the rate of divergence significantly, thus it allows for a longer walk length in order to capture deeper network connections.

C. Link Prediction

To evaluate the link predictions, typically, we use the output embedding to predict the future edges in the graph [6]. This experiment is carried out by selecting 50% of the edges at random, with the constraint that the residual graph must be connected, and then performing the training on the residual graph and predicting the removed links. The output node embeddings are then converted to edge embeddings using binary operators mentioned in I. The edge embeddings are compared with the removed edges of the graph to measure a model's performance.

In this research, we compare the proposed BalNode2Vec against the most recent state-of-the-art algorithms DeepWalk [13], Node2Vec [6], LINE [2], GraRep [34], and Walklet [14]. Tables II to V show the Area Under the Curve (AUC) score for each binary operator mentioned in Table I. We

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

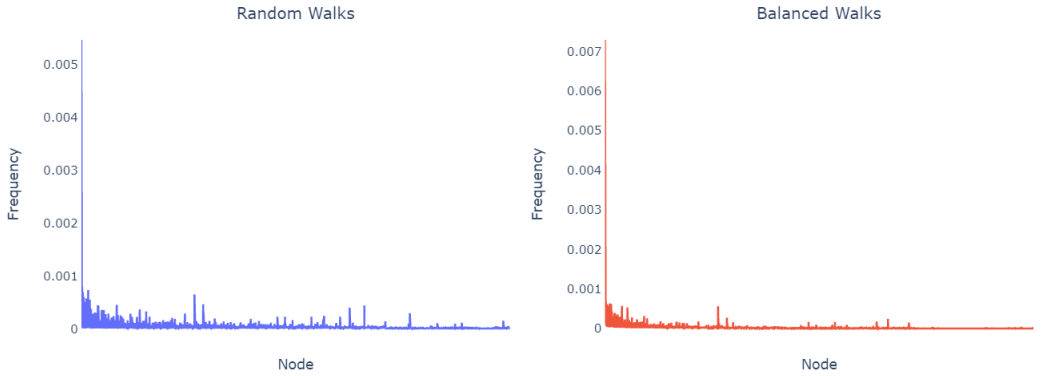

Fig. 4. Node frequencies in a Random Walk (left) and Balanced Walk (right) of length 8. (The nodes are sorted by degree.)

TABLE II
AREA UNDER CURVE (AUC) SCORES FOR LINK PREDICTION USING THE AVERAGE BINARY OPERATOR.

Algorithm	CourseNetworking	Reddit Hyperlink	PPI	arXiv	Facebook
BalNode2Vec	**0.7480**	**0.7644**	**0.7097**	**0.7157**	**0.7570**
DeepWalk	0.7324	0.7488	0.6647	0.6853	0.7360
Walklet	0.7127	0.7503	0.6451	0.6740	0.7470
LINE	0.7054	0.7096	0.6429	0.6843	0.7217
Node2Vec	0.7431	0.7422	0.6772	0.6932	0.7443
GraRep	0.7453	-	0.6819	0.7049	0.7518

TABLE III
AREA UNDER CURVE(AUC) SCORES FOR LINK PREDICTION USING THE HADAMARD BINARY OPERATOR.

Algorithm	CourseNetworking	Reddit Hyperlink	PPI	arXiv	Facebook
BalNode2Vec	**0.8837**	**0.9792**	**0.7796**	**0.9541**	**0.971**
DeepWalk	0.8452	0.9530	0.7481	0.9354	0.9591
Walket	0.8430	0.9618	0.7129	0.9110	0.9480
LINE	0.8247	0.9436	0.7244	0.887	0.9323
Node2Vec	0.8539	0.9627	0.7563	0.9374	0.9620
GraRep	0.8651	-	0.7580	0.9418	0.9567

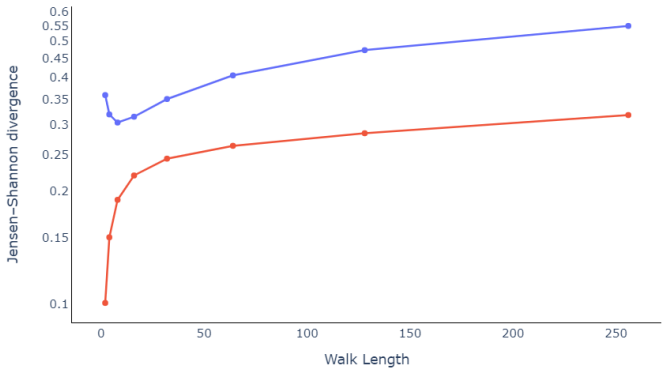

Fig. 5. Jensen-Shannon divergence of random and balanced walks based on the walk length.

could not measure the performance of GraRep on the Reddit Hyperlink dataset due to out of memory error. This is because GraRep is not scalable on graphs with near one million edges. Based on the results, BalNode2Vec performs better than all the other methods on every data set. The performance improves more significantly when dealing with networks that have less reciprocal connections like CourseNetworking and Reddit Hyperlink datasets. When we look at operators individually (Table I), BalNode2Vec outperforms the others in all different operators. Node2Vec works better than the other methods except on Reddit Hyperlink involving the Average Binary operator in which DeepWalk performs better than Node2Vec. When Hadamard operator is used, the performance of the BalNode2Vec shows more improvement than the other methods across all datasets.

D. Parameter Sensitivity

The proposed BalNode2Vec has several parameters related to the sampling strategy and network training that need to be fine-tuned for each network. With the same procedure for link prediction, Figure6 shows the sensitivity of the model

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

TABLE IV
AREA UNDER CURVE(AUC) SCORES FOR LINK PREDICTION USING THE WEIGHTED-L1 BINARY OPERATOR.

Algorithm	CourseNetworking	Reddit Hyperlink	PPI	arXiv	Facebook
BalNode2Vec	**0.8793**	**0.9754**	**0.7701**	**0.9463**	**0.9676**
DeepWalk	0.8312	0.9499	0.7455	0.9334	0.9517
Walklet	0.8368	0.9670	0.7114	0.9048	0.9483
LINE	0.8219	0.9428	0.7211	0.8947	0.9315
Node2Vec	0.8547	0.9684	0.7647	0.9352	0.9579
GraRep	0.8697	-	0.7671	0.9388	0.9497

TABLE V
AREA UNDER CURVE(AUC) SCORES FOR LINK PREDICTION USING THE WEIGHTED-L2 BINARY OPERATOR.

Algorithm	CourseNetworking	Reddit Hyperlink	PPI	arXiv	Facebook
BalNode2Vec	**0.8724**	**0.9620**	**0.7514**	**0.9481**	**0.9648**
DeepWalk	0.8349	0.9476	0.7428	0.9294	0.9516
Walklet	0.8407	0.9596	0.7150	0.9127	0.9429
LINE	0.8258	0.9428	0.7213	0.8920	0.9317
Node2Vec	0.8511	0.9644	0.7428	0.9341	0.9606
GraRep	0.8687	-	0.7601	0.9447	0.9542

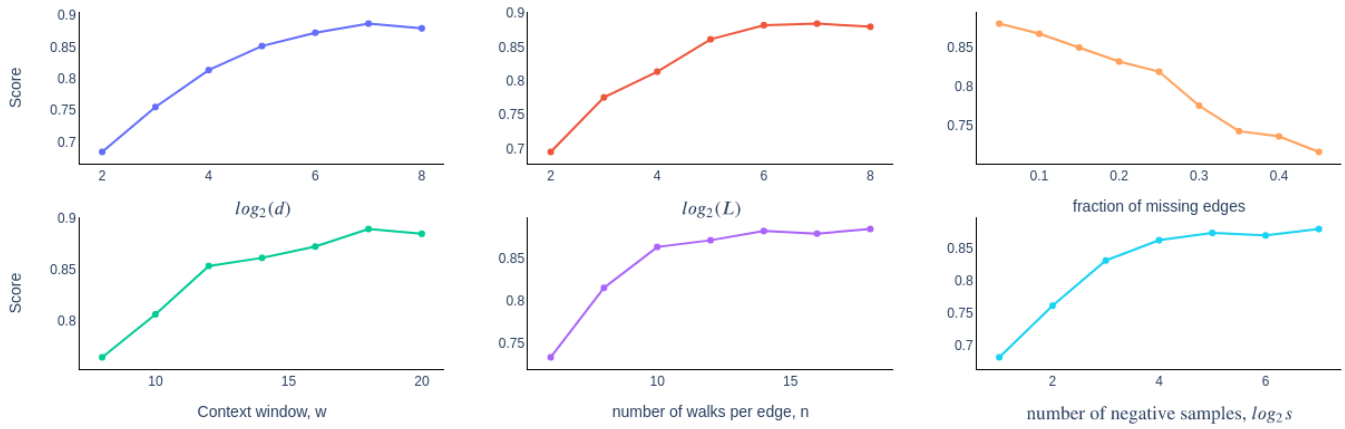

Fig. 6. Area Under Curve(AUC) scores for link prediction on the CourseNetworking dataset.

to various parameters. All the parameters are set to default value except the one being tested. The default values for this experiment were: walk per edge: 12, embedding dimension d=128, context window w=18 and walk length l=64.

We measure the AUC score against the number of features d, the walk length l, and context window size w and how they affect the performance. We observe that performance tends to saturate once the dimensions of the representations reach around 100. Similarly, we observe that increasing the number and length of walks improves performance. The context window size, w, also improves performance and training depth at the cost of increased training time. However, the performance differences are not that large when w is close to 20. The number of negative samples also has a diminishing effect on the accuracy of the model, and after some point(around 32), it is not computationally efficient to increase the number of negative samples.

V. DISCUSSION, CONCLUSION AND FUTURE WORK

In this research, our objective is to provide a scalable, adaptable, and efficient node embedding method - BalNode2Vec. We investigate the limitations of the random walk sampling strategy for sequence generation for the Skip Gram model and propose the BalNode2Vec model, which is capable of devising a model and improving the performance of the baseline Node2Vec model. By adopting a new sampling direction, BalNode2Vec can analyze the graph structure and modify the path accordingly, hence, ensuring that all the nodes are adequately sampled. We compare the proposed BalNode2Vec model against five different state-of-the-art models on five different data sets. The results demonstrate the BalNode2Vec can provide more consistent and accurate results and performs better than the existing models.

In the future, we plan to use the embeddings generated using this technique to improve the recommendation engine and

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

provide offer community and role-based results based on these embeddings. As for the algorithm, in the future, our effort will be to make the optimizer and sampler more interactive so that the graph sampling will be based on the optimizer's result, and the sampler would sample what the optimizer needs instead of performing random walks.

VI. Acknowledgements

We would like to thank cyberlab.iupui.edu for the support of this research as part of the Rumi Agent project and providing us with the required data to carry out the experiments. This research was supported in part by Lilly Endowment, Inc., through its support for the Indiana University Pervasive Technology Institute.

References

[1] Oren Barkan and Noam Koenigstein, "Item2vec:Neural Item Embedding for Collaborative Filtering." Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.

[2] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. LINE: Large-scale Information Network Embedding. In WWW, 2015.

[3] X. Li, N. Du, H. Li, K. Li, J. Gao, and A. Zhang. A deep learning approach to link prediction in dynamic networks. In ICDM, 2014.

[4] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In NIPS, 2013.

[5] Jing Zhang, Jie Tang, Cong Ma, Hanghang Tong, Yu Jing, and Juanzi Li. 2015. Panther: Fast top-k similarity search on large networks. In KDD 15. ACM, 14451454.

[6] Aditya Grover and Jure Leskovec. 2016. Node2Vec: Scalable Feature Learning for Networks. In KDD 16. ACM, 855864.

[7] X. Li, N. Du, H. Li, K. Li, J. Gao, and A. Zhang. A deep learning approach to link prediction in dynamic networks. In ICDM, 2014.

[8] Mahalanobis, P. C. (1936). On the generalized distance in statistics.

[9] S. Zhai and Z. Zhang. Dropout training of matrix factorization and autoencoder for link prediction in sparse graphs. In SDM, 2015

[10] K. Li, J. Gao, S. Guo, N. Du, X. Li, and A. Zhang. LRBM: A restricted boltzmann machine based approach for representation learning on linked data. In ICDM, 2014

[11] Ting Chen and Yizhou Sun. 2017. Task-Guided and Path-Augmented Heterogeneous Network Embedding for Author Identification. In WSDM 17. ACM

[12] L. Tang and H. Liu. Scalable learning of collective behavior based on sparse social dimensions. In Proceedings of the 18th ACM conference on Information and knowledge management, pages 11071116. ACM, 2009.

[13] B. Perozzi, R. Al-Rfou, and S. Skiena. DeepWalk: Online learning of social representations. In KDD, 2014

[14] B. Perozzi, V. Kulkarni, H. Chen, and S. Skiena. Dont walk, skip!: Online learning of multi-scale network embeddings. In Advances in Social Networks Analysis and Mining (ASONAM), 2017.

[15] Shuicheng Yan, Dong Xu, Benyu Zhang, Hong-Jiang Zhang, Qiang Yang, and Stephen Lin. 2007. Graph embedding and extensions: A general framework for dimensionality reduction. IEEE TPAMI 29, 1 (2007)

[16] Yoav Goldberg and Omer Levy. 2014. word2vec Explained: deriving Mikolov et al.s negative-sampling word-embedding method. CoRR abs/1402.3722 (2014).

[17] Amr Ahmed, Nino Shervashidze, Shravan Narayanamurthy, Vanja Josifovski, and Alexander J. Smola. 2013. Distributed Large-scale Natural Graph Factorization. In WWW 13. ACM, 3748.

[18] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In NIPS, 2001.

[19] Yuxiao Dong, Jing Zhang, Jie Tang, Nitesh V. Chawla, and Bai Wang. 2015. CoupledLP: Link Prediction in Coupled Networks. In KDD 15. ACM, 199208.

[20] J. Yang and J. Leskovec. Overlapping communities explain core-periphery organization of networks. Proceedings of the IEEE, 102(12):18921902, 2014.

[21] A. Vazquez, A. Flammini, A. Maritan, and A. Vespignani. Global protein function prediction from protein-protein interaction networks. Nature biotechnology, 21(6):697700, 2003.

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

[23] Xiang Ren, Wenqi He, Meng , Clare R Voss, Heng Ji, and Jiawei Han. 2016. Label noise reduction in entity typing by heterogeneous partial-label embedding. In KDD 16. ACM.

[24] Zhipeng Huang, Yudian Zheng, Reynold Cheng, Yizhou Sun, Nikos Mamoulis, and Xiang Li. 2016. Meta structure: Computing relevance in large heterogeneous information networks. In KDD 16. ACM, 15951604

[25] Martn Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jerey Dean, MaŁhieu Devin, Sanjay Ghemawat, Georey Irving, and others. 2016. TensorFlow: A system for large-scale machine learning. In OSDI 16.

[26] Yann LeCun, Yoshua Bengio, and Georey Hinton. 2015. Deep learning. Nature 521, 7553 (2015), 436444.

[27] Ming Ji, Jiawei Han, and Marina Danilevsky. 2011. Ranking-based classification of heterogeneous information networks. In KDD 11. ACM, 12981306.

[28] Jian Tang, Meng, and Qiaozhu Mei. 2015. PTE: Predictive Text Embedding rough Large-scale Heterogeneous Text Networks. In KDD 15. ACM, 11651174.

[29] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. ArnetMiner: Extraction and Mining of Academic Social Networks. In KDD 08. 990998

[30] L. Tang and H. Liu. Scalable learning of collective behavior based on sparse social dimensions. In Proceedings of the 18th ACM conference on Information and knowledge management, pages 11071116. ACM, 2009.

[31] L. Tang and H. Liu. Leveraging social media networks for classification. Data Mining and Knowledge Discovery, 23(3):447478, 2011.

[32] Yoshua Bengio, Rejean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. The Journal of Machine Learning Research, 3:11371155, 2003

[33] Dong, Yuxiao Chawla, Nitesh Swami, Ananthram. (2017). metapath2vec: Scalable Representation Learning for Heterogeneous Networks. 135-144. 10.1145/3097983.3098036.

[34] S. Cao, W. Lu, and Q. Xu. Grarep: Learning graph representations with global structural information. In KDD, 2015.

[35] S. Cao, W. Lu, and Q. Xu. Deep neural networks for learning graph representations. In AAAI, 2016.

[36] P. Goyal, E. Ferrara, Graph Embedding Techniques, Applications, and Performance: A Survey (2018), Knowledge-Based Systems.

[37] B.P. Chamberlain, J. Clough, and M.P. Deisenroth. Neural embeddings of graphs in hyperbolic space. arXiv preprint arXiv:1705.10359, 2017.

[38] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. 2013. B.-J. Breitkreutz, C. Stark, T. Reguly, L. Boucher, A. Breitkreutz, M. Livstone, R. Oughtred, D. H. Lackner, J. Bhler, V. Wood, et al. The BioGRID interaction database. Nucleic acids research, 36:D637D640, 2008.

[39] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014

[40] B.-J. Breitkreutz, C. Stark, T. Reguly, L. Boucher, A. Breitkreutz, M. Livstone, R. Oughtred, D. H. Lackner, J. Bhler, V. Wood, et al. The BioGRID interaction database. Nucleic acids research, 36:D637D640, 2008.

[41] Hamilton et al. 2017. Representation Learning on Graphs: Methods and Applications. IEEE Data Engineering Bulletin on Graph Systems.

[42] Scarselli et al. 2005. The Graph Neural Network Model. IEEE Transactions on Neural Networks.

[43] thecn.com, Cyberlab.iupui.edu

[44] K. Henderson, B. Gallagher, T. Eliassi-Rad, H. Tong, S. Basu, L. Akoglu, D. Koutra, C. Faloutsos, and L. Li. RolX: structural role extraction mining in large graphs. In KDD, 2012.

[45] David Liben-Nowell and Jon Kleinberg. 2003. The link prediction problem for social networks. In Proceedings of the twelfth international conference on Information and knowledge management (CIKM '03). ACM, New York, NY, USA, 556-559. DOI: https://doi.org/10.1145/956863.956972