YANES: Node Embedding for Community Detection and Link Prediction

Final Project Report

Kumaran Gunasekaran
Jeyavaishnavi Muralikumar
Sudarshan Srinivasa Ramanujam
Balasubramaniam Srinivasan
kugunase@eng.ucsd.edu
jmuralik@eng.ucsd.edu
sus046@eng.ucsd.edu
bsriniva@eng.ucsd.edu

ABSTRACT

The advancement of representation learning has allowed us to represent graphs as vector in a lower dimensional space. The biggest advantage of doing this is to apply known algorithms in vector space to understand properties of the underlying graphs. In this paper, Yet Another Node Embedding Scheme [YANES] we focus on the problems of community detection which can be seen as a unsupervised learning problem, and link prediction which is a supervised learning problem in the embedding space. We propose an embedding method inspired from the word embedding technique known as Glove. Various other vector embedding techniques such as LINE, Node2Vec were also analyzed and compared against our algorithm towards the task of community detection. A comparison was also made with Louvain community detection and spectral clustering methods. Furthermore, the embedding technique was used for the task of Link Prediction in a popular collaboration network and studied. The performance of the algorithm was compared with the performance of Node2vec for the task of Link prediction.

1 PROBLEM DEFINITION AND MOTIVATION

Through this project we aim to address the problems of community detection and link prediction in large graphs and networks through the use of representation learning. In specific we aim to learn low dimensional representations of nodes and their edges in the graph which could then be subjected to clustering algorithms as well as a supervised learning algorithm to predict links in the network.

Community detection in graphs is an important problem having concrete applications in balancing network loads, recommender systems, social networking, biological networks, etc. A variety of algorithms are used to identify the community structure in graphs, such as graph partitioning methods, clustering methods like hierarchical clustering, spectral clustering, etc., modularity based methods where we optimize global parameters and dynamic algorithms such as random walks to identify graph structure.[2].

Link Prediction is another important problem which helps in tackling applications such as recommending new friends in online social networks as well as suggesting interactions between the members of a company which are external to the hierarchical structure of the organization itself. This can also be used to suggest collaborations between researchers based on co-authorships.

Recent work in learning graph representations has led to new graph representations based on random walks and graph traversals. These algorithms perform such walks in order to learn a vector-space representation of graphs[11], [3], [12]. The resultant vector representation, which is low dimensional is then effectively used in supervised learning tasks such as multi-label classifications and link prediction. [3].

Furthermore, there have been other approaches which have built upon these works, where distributed representations of rooted sub-graphs in large graphs have been learnt for the purposes of classification [8]. Node similarity detection tasks have also been explored by learning node representations from Structural Identity [1].

Although learning representations, which are dependent on the underlying structure of the graphs appears to be positively correlated with the various communities, these have only been shown to work well in small networks (few ten to hundred nodes) where the communities exhibit unambiguous community structure. Large networks on the other hand have a nested core-periphery structure, consisting of a large moderately well-connected core and a large number of small very well-connected communities barely connected to the core [5]. We could find no literature that directly applied learning representations to the community detection problem for large networks (Link prediction was done before by both node2vec and LINE). We believe that a study of applying a learning representation, with modified embeddings to the aforementioned related works, could have a significant impact towards more efficient ways of determining communities in graphs.

By standardizing the methodology of clustering in the representation space, the project will be directed at learning more efficient

embeddings which would help to reproduce features that are rich in information, specific to community detection and link prediction.

2 BACKGROUND AND RELATED WORK

Community detection is a well-studied problem, with a variety of classes of solutions. These include clustering methods such as hierarchical and spectral clustering, algorithms that optimize a global parameter such as modularity, dynamic methods that depend on random walks, graph partitioning methods and methods that rely on graph connectivity, such as clique percolation [2]. In this work, we focus on vector space clustering techniques, since we apply representation learning to transform graph data to the vector space.

The link prediction problem can be visualized as the problem of finding missing links from an observed network: in most domains, a network is constructed using interactions based on observable data and then additional links are tried to infer from that, which while not directly visible, are likely to exist [6]. This line of thought slightly differs from our problem formulation in that it works with a static snapshot of a network, rather than considering network evolution; it also tends to take into account specific attributes of the node (which are inherently embedded in the low dimensional representations learnt), rather than evaluating the power of prediction methods based purely on the graph structure.

Numerous methods have been explored to learn word embeddings in the field of NLP, most notable one being word2vec algorithm[7]. A popular extrapolation of this algorithm to the graph space was the node2vec algorithm [3]. Node2vec utilizes the skipgram model proposed in word2vec and samples nodes from node samples generated using random walks. The idea is to preserve the neighborhood of a given node.

$$max_f \sum_{u \epsilon V} logPr\left(N_s\left(u\right)|f\left(u\right)\right) \tag{1}$$

In Equation 1, f corresponds to a matrix of size $|V| \times d$ parameters. Similar to the word2vec implementation in [7] where the neighbors of a given word are sampled as outputs, neighbors of a specific vertex are sampled in the implementation of node2vec. The sampling strategy employed combines random walks together with an interpolation between BFS and DFS search strategies, which ensures that a rich embedding based on the neighborhood is obtained for every vertex. The innovation in node2vec comes in assigning transition probabilities which biases the random walks towards BFS or DFS. Two hyper parameters (defined as p and q in the node2vec paper) control the amount of DFS and BFS. Negative sampling is employed to reduce the amount of computations drastically and make the algorithm practicable to implement.

LINE works by preserving the first order and the second order proximity between nodes in the original graph. This is also in some sense a neighborhood preserving algorithm. The KL distance between a defined probability distribution and the probability distribution generated by the embeddings is minimized. Similar methods using a different optimization function is defined for preserving the second order proximity. The second order proximity however cannot be applied to directed graphs and works only for undirected graphs. Moreover the optimization is applied separately for first order and second order nodes to generate embeddings. A combined optimization of both parameters is still an unsolved problem.

Joint probability between nodes i and j is defined as:

$$p_1\left(v_i, v_j\right) = \frac{1}{1 + exp\left(-\vec{u_i}^T.\vec{u_j}\right)} \tag{2}$$

where $u_i \epsilon R^d$ is the low dimensional representation of vertex v_i. Eqn 6 defines a probability distribution p over the space $V \times V$. The empirical probability distribution function as follows:

$$\hat{p}\left(i, j\right) = \frac{\omega_{ij}}{W} \tag{3}$$

where $W = \sum_{(i,j)\epsilon E} \omega_{ij}$. The KL distance between p and $\hat{p}$is taken as the objective function and minimized by updating node embeddings.

Deepwalk uses a maximum likelihood optimization function to compute node embeddings. One node is used to predict the context.

Using the conditional independence assumption (also used by the skip gram model), RHS can be reduced to $\prod_{j=i-w}^{i+w} Pr\left(v_j|\phi\left(v_i\right)\right)$

As we see from the above equations, the likelihood of vertices seen in the random walk is used as the optimizing function to learn embeddings. The skipgram model employed effectively in word2vec [7] is applied here as well together with hierarchical softmax in order to speed up computation.

Inspired by the node2vec algorithm, yet another NLP algorithm was explored and converted to graph domain to test its efficacy [10]. Glove (Global Vectors for Word Representation) which was effectively used in NLP to generate word embeddings was tried out in the graph domain. Glove effectively tries to capture similar words and makes sure relationship amongst words is maintained. Probability values based on relationship that two words have with a common word is captured and converted effectively to denote the relationship between the two words. When applying this concept to graphs, the intuition is that nodes that belong to the same communities would have similar embeddings and would be clustered together. More details on the working of he glove algorithm is elaborated in the Methodology section.

3 METHODOLOGY

The project was started with trial runs of various embedding algorithms using the popularly known Zachary's karate club graph. The resulting node embeddings were clustered using K-means clustering and the accuracy of the clustering was evaluated. Furthermore, various hyper parameters in the embedding algorithms were tweaked and resulting embeddings were evaluated.

Inspired by the usage of a NLP algorithm in graphs [3], Glove (Global Vectors for Word Representation), which is used for word embeddings effectively in NLP were tried out as a novel embedding strategy in the graph space.

3.1 Description of Modified GloVe for graphs

The original implementation of Glove uses a log bilinear model to derive vector representations of words, taking in to consideration both the word co-occurrence statistics as well as the word's context. GloVe is comparable if not superior to the skip gram model which considers only the word's local context to derive the word

representation [10]. Transforming this word representation learning into node representation learning involved finding a suitable analogy for the word-word co-occurrence matrix used by GloVe. The shortest path between individual pairs of nodes is used as a metric.

We define the co-occurrence matrix as a matrix containing the relation between individual elements in our graph. In the original implementation of Glove, word co-occurrences were taken and used to build the co-occurrence matrix. This intuition was that words with similar meanings when combined together with context words would generate high probability values. Extending this method to graphs, we tried out 2 variations:

(1) Random Walk Based Method: Node co-occurrence matrix was constructed based on number of times node neighbors were visited when performing a random walk from a starting vertex. Number of times neighborhood nodes was visited from the starting vertex was kept count and used to fill up elements of the co-occurrence matrix
(2) Distance based methods : The shortest path between individual pairs of nodes in the graph were used to populate the co-occurrence matrix. Since we wish to weigh the nodes closer to a another node in consideration higher, the inverse of the shortest between pairs of nodes is used to fill up the co-occurrence matrix. The same method is applied when building the embedding for weighted graphs. The inverse of the shortest path between the pairs of nodes computed using the weights is used to fill up the co-occurrence matrix.

The distance based method was seen to perform much better than the random walk based method. Both the methods discussed above can be extended trivially to directed and weighted graphs. The random walk based method utilizes the same loss function as that of Glove implementation for the NLP version. The model however changes for the distance based method. We will derive the modified loss function for this method based on original Glove implementation.

Let us denote the co-occurrence matrix as C. We need embeddings and a function which is capable of achieving the below mentioned equation.

$$F\left(w_i, w_j\right) = C_{ij} \tag{4}$$

where w_i and w_j correspond to node vectors. The RHS in equation 4 is a scalar quantity and the arguments of F is a vector quantity. One simple way to convert the vector quantity to a scalar is to take the dot product. In order to provide nodes that are at close proximity to a given node higher weight, we will modify 4 in the following manner.

$$F\left(w_i^T w_j\right) = \frac{1}{d_{ij}} \tag{5}$$

where d_{ij} corresponds to the shortest path between node i and node j in the graph. In the original implementation of Glove, F is chosen to be an exponential funtion in order to maintain homomorphism. However, we do not have such a constraint as we are not using anything equivalent to "probe words" in the original algorithm. We can simply extend equation 5 as follows:

$$w_i^T w_j = \frac{1}{d_{ij}} \tag{6}$$

Casting equation 6 as a weighted least squares problem, we can formulate the loss function as follows which is also partly inspired from [13]:

$$J = \sum_i \sum_{j|d_{ij}<k} f\left(\frac{1}{d_{ij}}\right)\left(w_i^T w_j - \frac{1}{d_{ij}}\right)^2 \tag{7}$$

where f corresponds to a weighting function. The weighting function needs to satisfy the following properties:

(1) $f(0) = 0$.
(2) $f(x)$should be non-decreasing so that rare occurrences are not over-weighted.
(3) $f(x)$should be relatively small for large values of x, so that frequent occurrences are not overweighted.

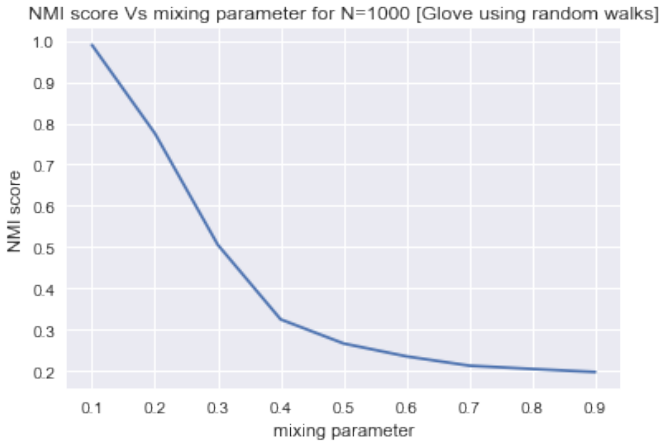
A large number of functions satisfy these properties, but the same function as the one that was used in the original implementation of Glove was reused.

$$f(x) = \begin{cases} \left(\frac{x}{x_{max}}\right)^{\alpha} & \text{if } x < x_{max} \\ 1 & \text{otherwise} \end{cases} \tag{8}$$

The hyper-parameter k is added in the loss function for controlling the amount of computation. Only nodes which are at most k distance away from any given node is considered and the rest are ignored. Using this hyper-parameter, the running time can be brought down drastically for large graphs.

4 EVALUATION OF COMMUNITY DETECTION

Figure 1: Glove with Co-occurrence matrix generated using random walks



As seen in fig:1, the algorithm performs very poorly for community detection. Henceforth the embedding generated using shortest path distances will be referred to as glove. The results for these embeddings are plot in the coming sections.

As mentioned in previous reports, we found that GloVe results in good community identification for small networks. Figures 2 and 3 show the communities identified by clustering GloVe embeddings for the Karate club and Les Miserables network, visualized using the spring layout. We can clearly see that the communities are separated as expected.

Figure 2: Community detection results from clustering Glove embeddings for the Zachary Karate club network

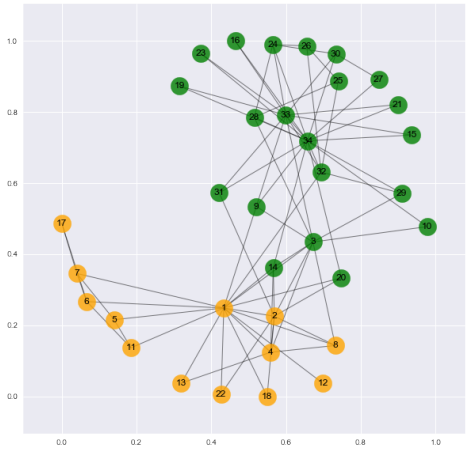
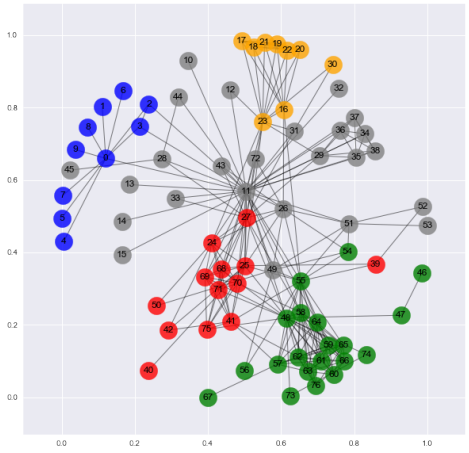


Figure 3: Community detection results from clustering GloVe embeddings for the *Les Miserables* network



Figures 4 and 5 show the elbow plot and the silhouette plot for the K-Means clustered GloVe embeddings of the Les Miserables network. We can see that there is a clear point in the elbow plot where the intra cluster distance to centroids starts declining steeply, as well as a clear point in the silhouette plot where the inter-cluster distance is maximum, indicating the reliability of the Glove embeddings for the task of community detection.

Figure 4: Elbow plot from clustering GloVe embeddings for the *Les Miserables* network

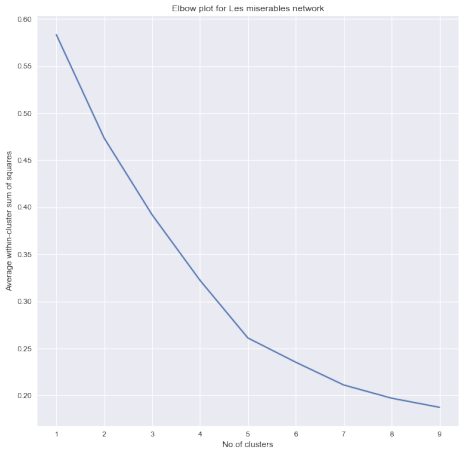
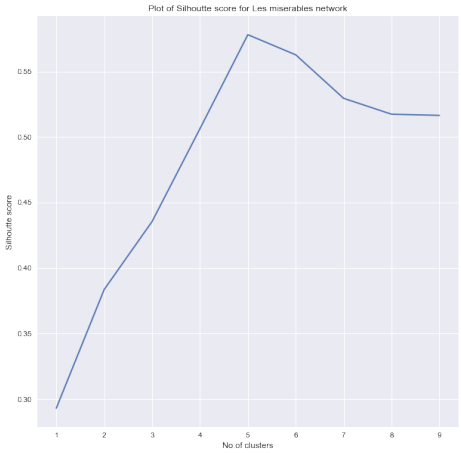


Figure 5: Silhouette plot from clustering GloVe embeddings for the *Les Miserables* network



4.1 Evaluation on artificial networks

K-means clustering was run on the resulting embeddings from Glove, Node2vec and LINE. We also compared the results of these algorithms with traditional methods such as spectral clustering and the Louvain algorithm.

Two benchmarks are generally used to evaluate the effectiveness of a community detection algorithm, namely the Girvan-Newman [GN] benchmarks and the LancichinettiâĂŞFortunatoâĂŞRadicchi [LFR] [4] benchmarks. The GN benchmark which is regularly used to test algorithms for community detection where different algorithms are compared based on their performance on this benchmark. The GN benchmark however has two major drawbacks:

- The expected degree of all nodes are the same
- Only equal size communities are created

These features are very unrealistic, as real world networks are known to be characterized by a wide range of expected degree and different community structures. The LFR benchmark generalizes the GN benchmark by introducing power law distributions of degree and community size. We consider this a generalization, since the GN benchmark is basically a special case in which the exponents of the distributions of degree and community sizes go to infinity. Majority of the community detection algorithms perform very well on the GN benchmark because of its highly simplistic structure. The LFR benchmark though discloses the downsides of the algorithms[4]. Additionally the LFR benchmark graphs can be constructed very quickly: the complexity of the construction algorithms is linear in the number of edges of the graph.

Taking the above into consideration the graphs we used for evaluation were generated by using the LFR benchmark algorithm [4]. The LFR model generates scale free networks and then rewires some links so that the ratio of internal to external links (as defined with respect to community structure) changes with respect to a mixing coefficient μ. If the value of μ is greater, the proportion of external links of a node (links to outside the community) increases, leading to ill-defined community structure. For an ideal community detection method, we want the performance and accuracy to be good even for high values of μ, or show less degradation at higher values of μ.

The LFR model also provides ground truth for generated graphs, which we used to evaluate the result of the clustering. Normalized mutual information between the results of the clustering and the ground truth was evaluated for different values of μ and different values of the average degree of nodes.

For the representational learning methods, we determined the number of clusters for K Means using the number of communities represented in the ground truth. As a sanity check, we plot the elbow plot for the LFR graphs to confirm that the optimal number of clusters in the LFR graphs8.

Higher average degree

Figure 6 shows the Normalized Mutual Information obtained for 5 different methods on an LFR set generated by setting the average degree as 20, maximum degree of nodes as 50 and with 1000 nodes overall. For all the three representation learning methods, we set the number of dimensions as 64. For LINE, we set 32 first-order dimensions and 32 second-order dimensions and concatenated them as suggested in [12].

We observe that all the methods decline steeply for values of μ greater than 0.6. At low values of μ, GloVe is comparable to Node2vec and LINE and representational learning methods are comparable to the Louvain algorithm. Spectral clustering fares poorly for low values of μ, but surpasses the performance of the Louvain algorithm for higher values.

Among Node2vec, GloVe and LINE, we find that Node2vec is the most sensitive to changes in μ. LINE and Node2vec perform comparably for higher values of μ, while GloVe performs better than both, which makes GloVe more suitable as a generic method for representation-based community detection.

Figure 7: Comparison of community detection methods for LFR graphs with average degree 10 and maximum degree 20

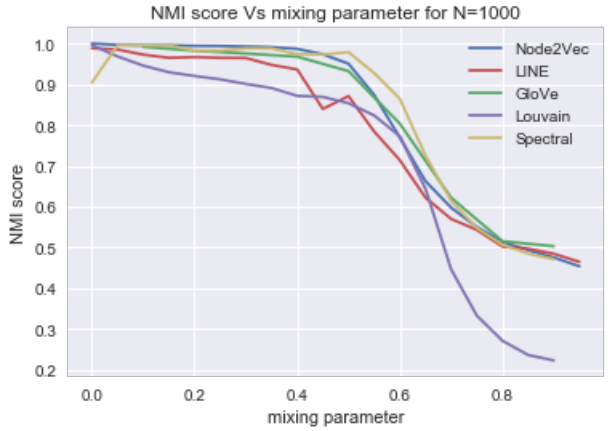
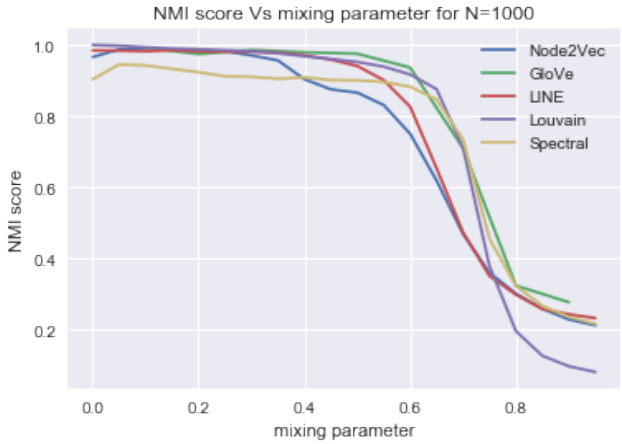


Figure 6: Comparison of community detection methods for LFR graphs with average degree 20 and maximum degree 50



Lower average degree

Based on [4] and on [9], we expect the performance of all the algorithms to decline for lower node degrees, with the degradation in performance starting at lower values of μ. However, the performance at higher values of μ is expected to be greater than with the higher degree case. Figure 7 shows observed NMIs with respect to μ for graphs generated with the number of nodes as 1000, average degree as 10 and maximum degree as 20.The power law exponent for the node degree was set at 2. The power law exponent

We observe that these expectations are generally realized, as with other algorithms that have been studied in literature. The improvement of GloVe's results over Node2vec and LINE for the case of ill-defined communities is less noticeable in this case, but still present. The degradation of the Louvain method is a departure from the general trend, but it is possible since modularity is not likely to be very high for most possible network configurations where external links between communities are more common than

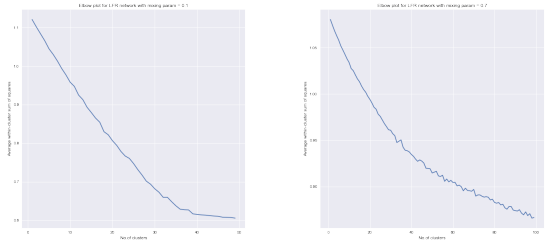
internal links within communities. The inferior performance of LINE is probably due to the fact that extracting 2nd-order proximity information is difficult for vertices with low degrees [12]. It will be interesting to see if a different combination of first and second order dimensions will improve community detection.

We do see that Node2vec performs better than GloVe for the case where $\mu < 0.5$. Node2vec is generally performs well with community detection tasks, with low parameter sensitivity and high performance at higher dimensions. The strength of GloVe is lies in how generalizable it is to networks with ill-defined communities.

Use of K-Means

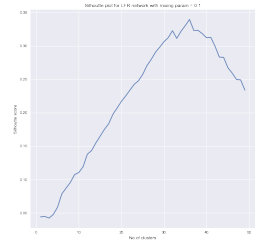
K-Means is not ideal for real world use, since it is unlikely that we would know the number of communities beforehand. However, we feel that the use of K-Means for our evaluation is justified, since it can be determined through features of the clustering. Figure 8 shows elbow and silhouette plots for graphs generated by the LFR model, for high and low values of μ. It is admittedly more difficult to discern the optimum value for higher values, but not impossible. For example, we can consider the ideal number of clusters as the value where the silhouette stops increasing.

Figure 8: Elbow plots for clustered GloVe embeddings, generated for LFR graphs with a low (0.1) and high (0.7) value of μ. We can see that the optimum number of clusters is easy to determine for graphs generated with low μ, whereas it is more difficult for graphs with greater μ

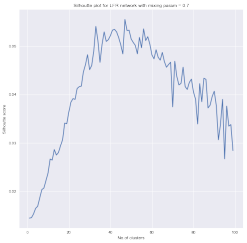


(a) Elbow plot, μ = 0.1

(b) Elbow plot, μ = 0.7



(c) Silhouette, μ = 0.1



(d) Silhouette, μ = 0.7

Performance and efficiency concerns

We note that LINE is capable of giving better results than shown, if the number of training samples is increased by many millions. However, this results in unreasonable inefficiency since the performance of generating embeddings deteriorates significantly as the number of training samples are increased.

5 EVALUATION ON LINK PREDICTION

The glove embeddings were used for the link prediction process and was compared with the performance of node2vec in link prediction. A popular collaboration network (GrQc network) was used to evaluate the algorithm.

5.1 Dataset Creation

The graph had to be split into training set and testing set for evaluation. Furthermore, nodes between which edges are not present needs to be added into the training set. In order to achieve this the following sequence of steps were implemented.

(1) 90 % of the edges in the original graph were randomly sampled and put into the training set
(2) The remaining edges were kept aside in the test set
(3) Pairs of nodes between which edges are not present are randomly sampled until we get an equal amount of node pairs as in step 1.
(4) Pairs of nodes between which edges are not present and are not present in the training set are randomly sampled and added to test set

5.2 Running the algorithm on the test set

(1) An undirected but weighted graph of the collaboration network is constructed using the data in training set. Nodes without edges are added with self loops to the graph
(2) Nodes present in test set, which are not present in training set are also added to the graph with self loops. (However in our experiments we did get any such nodes)
(3) We pass this newly generated graph to generate embeddings using Glove and node2vec
(4) In order to do the training, the embeddings which form an edge are combined using two techniques (i) Simple Averaging and (ii) Hadamard Operator
(5) Using the training set and the generated embeddings a soft SVM classifier is trained

5.3 Results

The results for the link prediction using Glove embeddings and node2vec embeddings are listed in Table: 1.

Algorithm	Operator	Test Error (%)
Node2vec	Hadamard	3.39 %
Node2vec	Simple Average	4.5 %
Glove	Hadamard	6.53 %
Glove	Simple Average	17.88 %
Adamic-Adar	-	5.47%
Preferential Attachment	-	24.70%

Table 1: Link Prediction Results

Yet another interesting find that we came across was that increasing the hyper-parameter k (discussed in section titled Glove), gave better performance with regards to link prediction, as can be seen from Table 2 The trade-off however was in increased runtime of the algorithm. This is in line with our intuition as well

since we populate more elements in the co-occurrence matrix with increasing value of k yielding better embeddings.

k = 3	Test Error = 25.4
k = 4	Test Error = 14.7
k = 5	Test Error = 6.5

Table 2: Test Error on Link Prediction with respect to k

6 CONCLUSIONS AND FUTURE WORK

To summarize our findings so far, we have described an adaptation of the Global Vectors concept [10] to generate network embeddings. We have then evaluated the results produced by GloVe by testing it for community detection and link prediction.

We have evaluated embeddings produced by node2vec, LINE and GloVe for the community detection problem. Based on our analysis, GloVe performs better than other graph representations for the general case, and is robust to lack of definition in community structure and graphs with low degrees.

For the link prediction task, we observe that GloVe performs better as nodes at greater distances are considered while constructing the co-occurrence matrix. We have not evaluated the performance of GloVe at higher values of k, which is a direction for future work. We have also not evaluated link prediction for other operators such as the weighted L1 and L2 norms.

It also remains to be seen how well GloVe performs with other predictive tasks such as graph classification. There is also a need for formal analyses of the performance of generating GloVe embeddings. We haven't formally benchmarked our GloVe implementation, but based on our experience, we are confident that it is less resource intensive than LINE. Another direction is to explore how the performance of GloVe for less perfect graph structures can be exploited, maybe by studying GloVe's efficacy when applied to sparse or sparsified networks.

Other directions for future work concern the application of these methods to tasks on large graphs. As the number of vertices increases, generating embeddings becomes more expensive. For the specific task of community detection, even this is eclipsed by the task of clustering the high-dimensional embeddings.

REFERENCES

[1] Daniel R Figueiredo, Leonardo FR Ribeiro, and Pedro HP Saverese. 2017. struc2vec: Learning Node Representations from Structural Identity. *arXiv preprint arXiv:1704.03165* (2017).

[2] Santo Fortunato. 2010. Community detection in graphs. *Physics reports* 486, 3 (2010), 75–174.

[3] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 855–864.

[4] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. 2008. Benchmark graphs for testing community detection algorithms. *Physical review E* 78, 4 (2008), 046110.

[5] Jure Leskovec, Kevin J Lang, Anirban Dasgupta, and Michael W Mahoney. 2009. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics* 6, 1 (2009), 29–123.

[6] David Liben-Nowell and Jon Kleinberg. 2007. The link-prediction problem for social networks. *journal of the Association for Information Science and Technology* 58, 7 (2007), 1019–1031.

[7] 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.

[8] Annamalai Narayanan, Mahinthan Chandramohan, Lihui Chen, Yang Liu, and Santhoshkumar Saminathan. 2016. subgraph2vec: Learning distributed representations of rooted sub-graphs from large graphs. *arXiv preprint arXiv:1606.08928* (2016).

[9] Günce Keziban Orman and Vincent Labatut. 2009. A comparison of community detection algorithms on artificial networks. In *International Conference on Discovery Science*. Springer, 242–256.

[10] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global Vectors for Word Representation.. In *EMNLP*, Vol. 14. 1532–1543.

[11] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 701–710.

[12] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*. ACM, 1067–1077.

[13] Allen Tran. 2015. Towards Anything2Vec. (2015). allentran.github.io/graph2vec