# Report on DeepWalk: Online Learning of Social Representations

Harsh Pathak (2016041), Nihesh Anderson (2016059)

***Abstract*— Deep Walk is an unsupervised learning technique that encodes social relationships of vertices in a continuous low dimensional vector space that captures neighbourhood similarities and community memberships. The algorithm proceeds by sampling a random walk and maximises the likelihood of the random walk using skipgram model, in the embedding space, thereby enforcing that nodes that have similar neighbourhoods in the graph metric are similar in euclidean metric as well. In this project, we implement Deep Walk algorithm to perform classification of nodes in Blog Catalog dataset. We also propose novel modifications to the existing algorithm to increase the evaluation metric on Blog Catalog dataset. Implementation: https://github.com/nihesh/GraphNodeClasification.git**

## I. INTRODUCTION

Over the years, many techniques have been proposed to perform learning on graphical data. Some of the techniques include Spectral Clustering [1], Edge Cluster [2], Modularity [3], wvRN [4] and Majority Voting.

**Dataset** Blog Catalog dataset has $10,312$ vertices and $3,33,983$ edges representing the affinity between bloggers (markov network) in the world. The bloggers are divided into 39 distinct groups based on blog genre.

The existing approaches in the literature are either too naive to work well (Majority) or it's computationally inefficient for large scale datasets (Spectral Clustering). Deep Walk [5] is a deep learning based online algorithm that draws a good balance between the issues addressed above. Owing to the size of the dataset and the need for computationally efficient algorithm, we resort to Deep Walk algorithm to classify the vertices.

Further Deep Walk leads to state of the art classification F-scores when the graph is sparsely labelled, which is a salient feature of this algorithm.

## II. METHOD - LIKELIHOOD MAXIMISATION OF RANDOM WALKS

### A. Problem Statement and Solution Sketch

Given a markov network G = (V,E), vertex feature vector $X \in R^{|V| \times s}$ and target labels Y (note that the graph can be sparsely labelled), we want to classify the unlabelled nodes as accurately as possible. In traditional machine learning, we simply learn a mapping between X and Y. In this problem, we would like to use the markov network given to us along with X to solve the classification problem.

To tackle this problem, we first try to embed the graph into a d-dimensional euclidean space to obtain feature vectors $\phi \in R^{|V| \times d}$ such that similar vertices in the graph are closer in the euclidean space, using deep walk algorithm. We then concatenate the learnt features with X to obtain a new feature vector and use traditional machine learning tools like SVM to perform node classification.

### B. Random Walk

Measure of similarity between two nodes can be subjective. One can measure similarity using shortest distance metric, density of nodes in the neighbourhood, etc. However, these similarity metrics are hand crafted and may not generalise well to different kinds of graphs. Can we come up with a generic similarity metric that captures the topology of the entire graph? This is where the idea of random walk comes in. A random walk is fundamentally "walking randomly". Start from a vertex v, move to one of its neighbours uniformly at random and recursively do this till we have a fixed number of nodes in the walk.

Random walks capture a lot of interesting properties in the graph. If a random walk has many repeated vertices, one may infer that the region around the repeated vertex may not be very dense (the probability of returning to the same vertex is higher). Similarly, if two vertices don't co-occur in a random walk, it is highly likely that both the vertices are located far apart in the graph. There are many more statistical properties of the graph that a random walk models. Therefore, we would like to transfer all the random walk statistics in the graph to the euclidean embedding we wish to learn. We achieve this by maximising the likelihood of the random walk in the euclidean space by suitably modelling probabilities in the embedding space.

### C. Likelihood Maximisation

Consider a random walk $W = (v_1, v_2, ..., v_t)$ sampled using the procedure described in Subsection II-B. To evaluate the likelihood of this walk, we use the skip gram model:

For every vertex $v_i$ in the walk $W$, consider a neighbourhood region $(v_{i-w}, v_{i-w+1}, ..., v_{i+w})$. We want to maximise the joint occurrence of the nodes in this window (w is the window size) conditioned on $v_i$ for all the vertices in the walk. Assuming conditional independence, we can write the joint probability as a product of conditional marginals and deduce the probability to

$$P(window|\phi(v_i)) = \prod_{j=i-w}^{i+w} P(v_j|\phi(v_i)) \tag{1}$$

Now, the problem reduces to modelling $P(u|\phi(v))$ for any two nodes in the embedding space. This is achieved using hierarchical softmax. Simple Softmax over all nodes can not be used when the number of nodes are large. Hierarchical Softmax is a simplification over Simple Softmax that reduces

computation from $O(|V|)$ to $O(log|V|)$ using a complete tree. We use Huffman coding to create shorter codes for more frequent nodes. Using hierarchical softmax, $P(u|\phi(v))$ gets formulated as -

$$P(u|\phi(v_i)) = \prod_{l=1}^{l=(log|V|)} P(b_l|\phi(v_i)) \tag{2}$$

where

$$P(b_l|\phi(v)) = sigmoid(\psi(b_l)^T\phi(v)) \tag{3}$$

Here $\psi$ is parameter of Hierarchical softmax tree(specifically for the internal nodes). Maximising product of probabilities in Eqn 1 leads to vanishing gradients. Therefore we maximise the log likelihood so that product gets transformed to summation.

### D. Novelty

1) We observed a minor flaw in the paper. Modelling the probability as described in Eqn 3 is not accurate. This is because the dot product is proportional to the cosine angle between the vectors. Sigmoid activated dot product yields an output close to 1 for vectors with large magnitude even if they have a large separating angle between them. This is not desirable. Therefore, we normalise the probabilities by the length of the vectors, scaled to [0, 1]. More formally,

$$P(u|\phi(v)) = \frac{1 + \frac{\phi(u)^T\phi(v)}{|\phi(u)||\phi(v)|}}{2} \tag{4}$$

2) Normalising the probabilities using softmax or any other technique that iterates over all the vertices in the graph is infeasible. Although Hierarchical Softmax is a well known trick that works well in practice, we try to explore negative sampling introduced in word2vec [6] to normalise the probabilities computed in Eqn 4. Computationally, negative sampling is as fast as hierarchical softmax.

## III. EXPERIMENTS AND RESULTS

We implemented deep walk algorithm from scratch and ran it on Karate Network. Karate Network is a fairly small graph with 34 nodes and a total of 78 edges. We set skip gram window length of 3, random walk length of 10 and embedding dimension as 2. We also sample a total of 500 random walks for each vertex. We use a learning rate of 0.025 throughout training. Figure 2 shows the visualization of nodes in karate network using our re-implementation of deep walk. We now train our baseline re-implementation and novel implementation on the blog catalog dataset and fit a linear SVM on the learnt embeddings. The results are summarized in table 1 and 2. The scores of baseline implementation are similar to scores in the original paper, while our novel additions to deepwalk have slightly higher scores than our baseline.

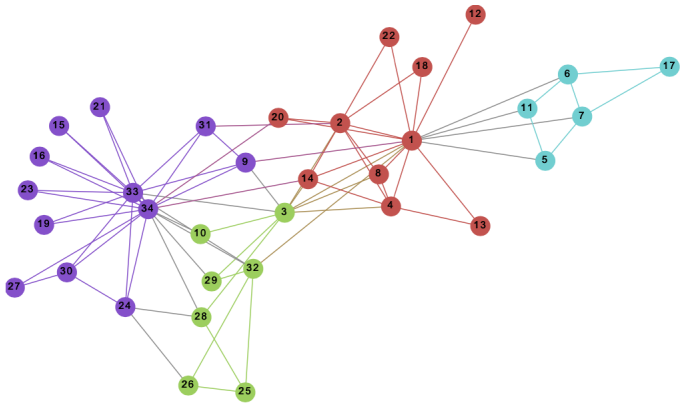

Fig. 1. Karate Network

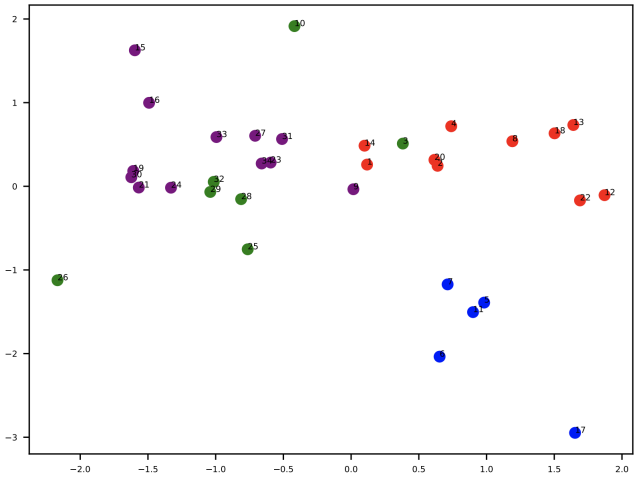

Fig. 2. Karate Network

## IV. OBSERVATIONS

On the Karate Network, we find that nodes that are similar to each other are closer to each other in the two dimensional eucledian space. The color of nodes in the Karate graph is ground truth of node membership which is similarly clustered in our deep walk output. It is clearly evident from the figure that the algorithm is functioning as desired, thereby serving as a proof of concept.

In addition to the quantitative findings that our novel implementation outperforms baseline, we also establish qualitatively that our method works. To this measure, we pick a random source node in the blog catalog dataset and plot the embeddings of its direct and indirect neighbors on scatter plots using TSNE. We claim that nodes that are near to our source node should be clustered together around the source node in euclidean space, whereas nodes that are further away in the graph should be scattered away from this cluster. In Fig. 3, we show visualisations of our claim for both novel as well as baseline implementation, and find that novel implementation creates well separated

TABLE I

F1 SCORES OF DEEPWALK BASELINE IMPLEMENTATION

| % Labeled Nodes in Training | 10% | 20% | 30% | 40% | 50% | 60% | 70% | 80% | 90% |
|---|---|---|---|---|---|---|---|---|---|
| Train F1 Scores | | | | | | | | | |
| Macro F1 | 33.7 | 31.3 | 29.54 | 29.51 | 29.94 | 28.22 | 30.04 | 27 | 30.7 |
| Micro F1 | 40 | 38.7 | 36.8 | 37.12 | 36.6 | 35.43 | 36.16 | 34.83 | 36 |
| Test F1 Scores | | | | | | | | | |
| Macro F1 | 17.1 | 20.11 | 23.55 | 23.71 | 24.77 | 26.68 | 26.07 | 28.09 | 28.15 |
| Micro F1 | 28.3 | 29.8 | 31.9 | 31.83 | 31.98 | 34 | 32.94 | 35.72 | 33.45 |

TABLE II

F1 SCORES OF NOVELTY DEEPWALK IMPLEMENTATION

| % Labeled Nodes in Training | 10% | 20% | 30% | 40% | 50% | 60% | 70% | 80% | 90% |
|---|---|---|---|---|---|---|---|---|---|
| Train F1 Scores | | | | | | | | | |
| Macro F1 | 41.73 | 34.92 | 35.01 | 30.87 | 30.26 | 30.28 | 31.49 | 31.67 | 30.7 |
| Micro F1 | 46.55 | 43.33 | 42.77 | 41.43 | 41.42 | 35.88 | 36.35 | 36.65 | 36.7 |
| Test F1 Scores | | | | | | | | | |
| Macro F1 | 18.27 | 19.43 | 20.9 | 24.85 | 25.13 | 29.33 | 29 | 30.04 | 31.35 |
| Micro F1 | 29.75 | 31.84 | 32.96 | 33.7 | 34.47 | 34.59 | 34.04 | 34.89 | 36.69 |

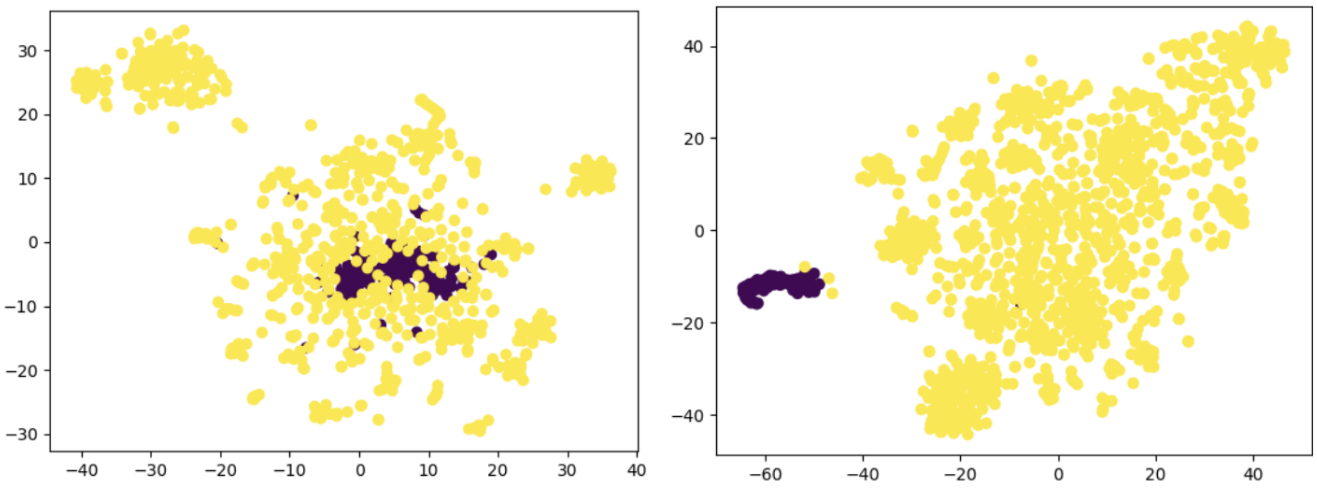


Fig. 3. Visualization of embeddings of close nodes (violet) and far nodes (yellow) in left: deepwalk baseline, right: deepwalk novelty implementation.

clusters, whereas the baseline fails to do so. This is because of introducing normalization of embedding vectors during training and negative sampling of cosine angles.

## V. INDIVIDUAL CONTRIBUTION

Both of us worked on the problem independently to understand it from various perspectives and then pooled our ideas to make a better system. Therefore, we have two different implementations of the project. Harsh implemented the baseline model that uses skipgrams with softmax tree. Nihesh implemented novel additions to deepwalk by coding negative sampling, normalization of embeddings and replaced the dot product in softmax with cosine angles.

## REFERENCES


[1] L. Tang and H. Liu. Leveraging social media networks for classification. Data Mining and Knowledge Discovery, 23(3):447–478, 2011.

[2] 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 1107–1116. ACM, 2009.

[3] L. Tang and H. Liu. Relational learning via latent social dimensions. In Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '09, pages 817–826, New York, NY, USA, 2009. ACM.

[4] S. A. Macskassy and F. Provost. A simple relational classifier. In Proceedings of the Second Workshop on Multi-Relational Data Mining (MRDM-2003) at KDD-2003, pages 64–76, 2003.

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

[6] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed Representations of Words and Phrases and their Compositionality. Accepted to NIPS 2013.