



DEEPWALK: ONLINE LEARNING OF REPRESENTATIONS

Apoorva Vinod Gorur

Feature Engineering

github.com/apoorvavinod/DeepWalk_implementation



Table of Contents

1. Introduction	3
2. Method	3
2.1 Random Walk generation	3
2.2 Representation Learning with Skip-gram.....	4
2.3 Optimization.....	4
3. Experiments	4
3.1 Pre-requisites	4
3.2 Usage.....	5
3.3 Dataset	5
3.4 Results.....	6
5. Conclusion	8
6. References	8

Abstract

The field of Graph Analysis has had major developments ever since the rise of Social Networks and Map navigation. DeepWalk takes a completely different approach to Graph/Network analysis compared to its contemporaries. The rise of deep learning gave rise to many innovative methods for Natural Language Processing. A popular method is to generate alternate representations for words in a vocabulary with the help of neural networks. This was made popular by the Word2Vec model proposed by Tomas Mikolov [2]. DeepWalk's model makes use of NLP techniques by using the skip-gram model of word2vec to generate embeddings for nodes in a graph. The creation of DeepWalk has led to many derivative models which fuse NLP methods with Graph analysis.

1. Introduction

Graph/Network analysis has many applications today such as link prediction, content recommendation, anomaly detection and efficient route calculation. DeepWalk makes use of Natural Language Processing techniques to learn social representations of nodes in a graph. The social representation, also called embedding represents the relationship of the node with its neighbors and the graph as a whole. They denote the node's social similarity between the members of the graph and also the community structure of the node. The embeddings generated by the model can be used as features for a variety of applications. This project aims to implement Bryan Perozzi's DeepWalk model and compare results with the original model as well as other noteworthy models.

2. Method

This sections presents the main components of the algorithm and the reasons behind choosing them for the model. The workflow of the model is as follows:

1. Generate a corpus of random walks from the node
2. Build node embeddings using the walk corpus
3. Use embeddings for classification, prediction or any other task

2.1 Random Walk generation

A random walk in a graph denoted by $W_{v1}, W_{v2} \dots W_{vk}$ is a walk starting from $v1$ and ending at vk where the subsequent nodes are chosen by randomly selecting one of the neighbors of the current node. The length of each walk is denoted by t . Random walks are generated γ times for each node. The random walks are treated as sentences as in the case of an NLP task. The reason behind choosing this approach is because of the fact that the node frequency of the random walks generated in this step follows a power-law distribution (Zipf's law) [3]. It is also observed that word frequency in natural language vocabulary follows a power-law distribution. It is evident that walks generated by random sampling of neighbors capture node similarity and community structure information.

2.2 Representation Learning with Skip-gram

Representation learning of a node is a process to determine a vector of values of dimension d accurately represent the embeddings/features of the node. In this step, the skip-gram model of word2vec [2] is used to learn the embeddings. If a function $\Phi: v \in V \rightarrow R^{V \times d}$ represents the embeddings of the nodes, then the purpose of learning embeddings is to estimate the likelihood of observing node v_i given all the previous nodes visited so far in the random walk. This can be written as

$$\Pr(v_i | \Phi(v_1), \Phi(v_2), \dots, \Phi(v_{i-1}))$$

The original skip-gram model however poses a problem when the graph is huge and the walk length grows big. It uses a softmax model to estimate the probabilities. The calculation of the denominator term for the softmax formula becomes computationally expensive when there is a huge number of nodes whose probabilities have to be calculated. To solve this problem of expensive computation, two methods called Hierarchical softmax and Negative sampling [4] have been proposed.

2.3 Optimization

The optimization algorithm used here is Stochastic Gradient Descent (SGD). SGD optimizes the embedding function as it iterates over the data. The model's parameters are updated at each step with a learning rate α by calculating gradients according to back-propagation algorithm. The OneVsRest Logistic Regression by LibLinear [5] has been used to compare the generated embedding's efficiency.

3. Experiments

The DeepWalk model has been implemented using Python. The project is open-source and is available for download [here](#). Various libraries have been used to aid the implementation. The following sections show instructions on how to run the model and also the structure of the project with descriptions of important functions.

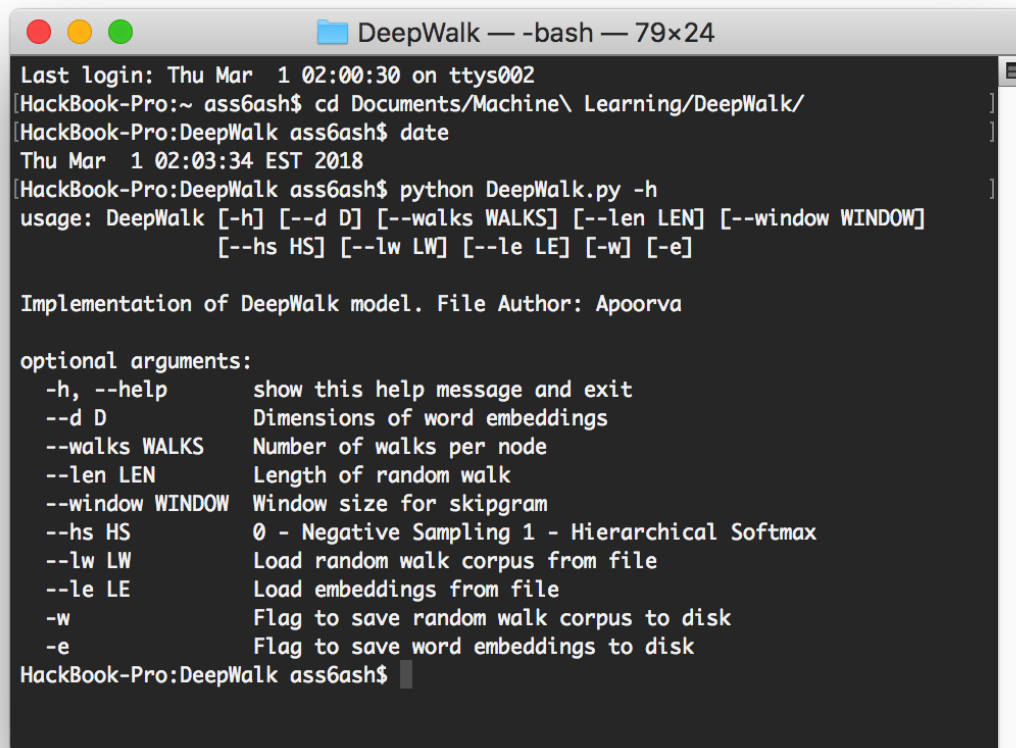
3.1 Pre-requisites

The following are needed to run the files. Download project files and run 'pip install -r requirements.txt' from the project folder.

- Python 3.6
- Networkx
- Gensim
- Matplotlib
- Scikit-learn
- Scipy

3.2 Usage

Install pre-requisite dependencies from previous section. Open terminal and change your directory to where the project files are located. The files needed to run are DeepWalk.py, Graph.py, Classifier.py and blogcatalog.mat. You can run the model by entering '`python DeepWalk.py -h`'. This will give you a list of command line arguments that can be used to set the parameters for the DeepWalk model. The arguments and their descriptions are given below



```
DeepWalk — -bash — 79x24
Last login: Thu Mar  1 02:00:30 on ttys002
[HackBook-Pro:~ ass6ash$ cd Documents/Machine\ Learning/DeepWalk/
[HackBook-Pro:DeepWalk ass6ash$ date
Thu Mar  1 02:03:34 EST 2018
[HackBook-Pro:DeepWalk ass6ash$ python DeepWalk.py -h
usage: DeepWalk [-h] [--d D] [--walks WALKS] [--len LEN] [--window WINDOW]
               [--hs HS] [--lw LW] [--le LE] [-w] [-e]

Implementation of DeepWalk model. File Author: Apoorva

optional arguments:
  -h, --help            show this help message and exit
  --d D                Dimensions of word embeddings
  --walks WALKS        Number of walks per node
  --len LEN            Length of random walk
  --window WINDOW      Window size for skipgram
  --hs HS              0 - Negative Sampling 1 - Hierarchical Softmax
  --lw LW              Load random walk corpus from file
  --le LE              Load embeddings from file
  -w                   Flag to save random walk corpus to disk
  -e                   Flag to save word embeddings to disk
HackBook-Pro:DeepWalk ass6ash$
```

Figure 1 Command Line arguments

Note that options to save/load the walk corpus file and the embeddings file to/from the disk have been provided. This has been done so because generating the corpus and embeddings take quite a long time if the graph is huge or the number walks and walk lengths are large.

3.3 Dataset

The data used for the experiment is the BlogCatalog dataset. It can be downloaded [here](#). The nodes in the graph represent Blog authors and the edges represent the friendship between the users. There are 39 groups to which each user can subscribe to. Figure 2 shows more information about the dataset.

```
[HackBook-Pro:DeepWalk ass6ash$ python DeepWalk.py
Name: blogcatalog.mat
Type: Graph
Number of nodes: 10312
Number of edges: 333983
Average degree: 64.7756
-----
```

Figure 2 Graph info

3.4 Results

This section reports results of the model with different parameters and also compares the performance with other models such as SpectralClustering [6], EdgeCluster [7], Modularity [8], wwRN[9] and majority [10]. Table 1 shows the average Micro and Macro F1 scores for training across 10% to 90% training data with varying parameter values for Dimensions(d) and number of walks(γ).

Dimensions	Walk Length	Avg. Micro-F1 score (%)	Avg. Macro-F1 score (%)
64	40	40.5	25.32
	60	40.83	25.24
	80	40.81	25.76
128	40	39.51	25.6
	60	40.12	26.19
	80	40.59	26.65
256	40	38.38	25.22
	60	39.07	26.14
	80	39.09	25.94

Table 1 Average Macro and Micro F1 Scores

Table 2 and Table 3 show the comparison of best performance of DeepWalk with other models. The values in bold show the highest performance for the corresponding training size. The Spectral Clustering, EdgeClustering and Modularity have their dimensions(d) set to 500.

	Training Size	10%	20%	30%	40%	50%	60%	70%	80%	90%
Micro-F1(%)	DeepWalk	35.83	39.61	40.89	41.0	41.83	41.93	42.54	42.6	42.52
	SpectralClustering	31.06	34.95	37.27	39.83	39.97	40.99	41.66	42.42	42.62
	EdgeCluster	27.94	30.76	31.85	32.99	34.12	35.00	34.63	35.99	36.29
	Modularity	27.35	30.74	31.77	32.97	34.09	36.13	36.08	37.23	38.18
	wwRN	19.51	24.34	25.62	28.82	30.37	31.81	32.19	33.33	34.28
	Majority	16.51	16.66	16.61	16.70	16.91	16.99	16.92	16.49	17.26

Table 2 Comparison of best Micro F1 scores across different training sizes

	Training Size	10%	20%	30%	40%	50%	60%	70%	80%	90%
Macro-F1(%)	DeepWalk	21.79	23.95	25.7	27.02	27.89	29.1	29.34	29.67	27.66
	SpectralClustering	19.14	23.57	25.97	27.46	28.31	29.46	30.13	31.38	31.78
	EdgeCluster	16.16	19.16	20.48	22.0	23.00	23.64	23.82	24.61	24.92
	Modularity	17.36	20.0	20.8	21.85	22.65	23.41	23.89	24.2	24.97
	wvRN	6.25	10.13	11.64	14.24	15.86	17.18	17.98	18.86	19.57
	Majority	2.52	2.55	2.52	2.58	2.58	2.63	2.61	2.48	2.62

Table 3 Comparison of best Macro F1 scores across different training sizes

Figure 3 shows an F1 score Vs Training size graph for the results of running the model with the following parameters. Dimensions(d) = 128, window size for skip-gram = 10, number of walks per node(γ) = 40 and walk length(t) = 80.

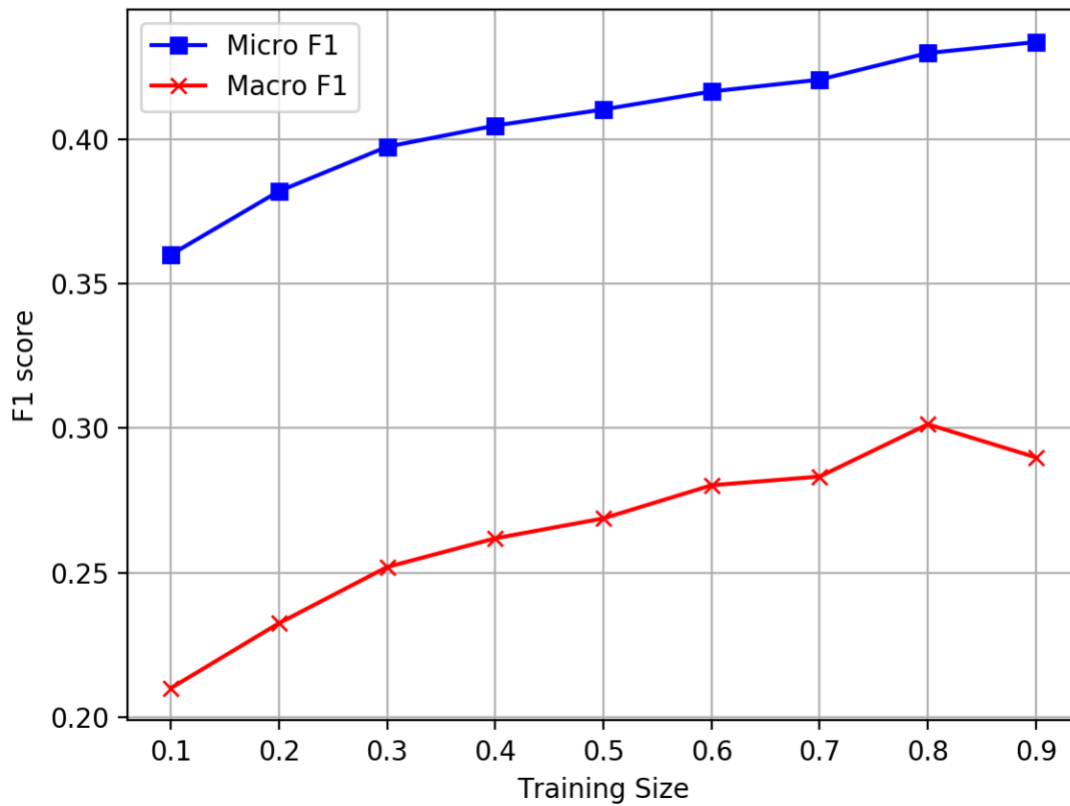


Figure 3 F1 score Vs Training size

5. Conclusion

The results reported in Section 4 clearly show that the DeepWalk model performs better than its contemporaries in most cases. The SpectralClustering model comes close to DeepWalk's performance on this dataset. But the paper written by Bryan Perozzi [1] reports performance comparison for other datasets as well and it can be seen that DeepWalk outperforms all other models. The parallelizable nature of random walk generation and the skip-gram embedding process makes DeepWalk scalable enough to be very much viable for real-world datasets which are huge. DeepWalk is a good example to show that techniques from other domains such as Natural Language Processing can also be ported and used efficiently in other domains as well. Future improvements to the model include support for multi-graphs, weighted graphs and implementation of other embedding techniques.

6. References

- [1] B. Perozzi, A. Rami and S. Skiena. DeepWalk: Online learning of Social Representations. In *Proceedings of the 20th ACM SIGKDD international Conference on Knowledge Discovery and Data Mining, KDD '14*, pages 701-710, New York, NY, USA, 2014. ACM.
- [2] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. CoRR, abs/1301.3781, 2013.
- [3] <https://nlp.stanford.edu/IR-book/html/htmledition/zipfs-law-modeling-the-distribution-of-terms-1.html>
- [4] Rong, Xin. word2vec Parameter Learning Explained. 2014
- [5] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.
- [6] L. Tang and H. Liu. Leveraging social media networks for classification. *Data Mining and Knowledge Discovery*, 23(3):447–478, 2011.
- [7] 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.
- [8] 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.
- [9] 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