DeepWalk

Online Learning of Social Representations

Jihyeong Jung
Department of Mathematical Sciences, KAIST

KAIST Data Science & Artificial Intelligence Lab
2021 Summer Internship Seminar
2021.07.05.

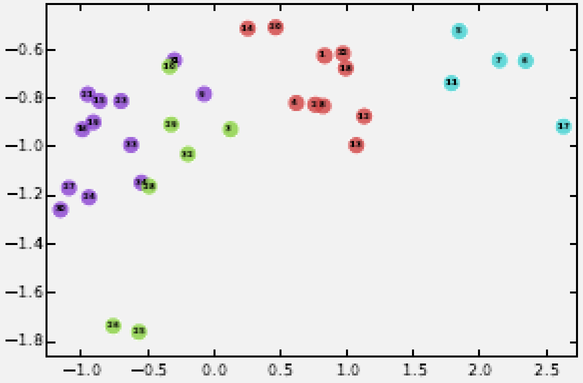
Contents

Introduction

- What this paper aims?

- Graph structure to features
  - DeepWalk is one of the Graph Imbedding method(ex. Node2Vec)
  - It's aimed at learning Good Latent Representations of vertices of the graph(network)
  - It is Unsupervised method; it captures 'topological', structural information from the graph.

(a) Input: Karate Graph



(b) Output: Representation

Introduction

- Problem Definition

- Given settings
  - Given network(graph) $G = (V, E)$, V: vertices(members) of the network, $E \subseteq V \times V$: Edges
- Goal of DeepWalk
  - Learning latent representation $X_E \in \mathbb{R}^{|V| \times d}$, d : 'small' number of latent dimensions

Introduction

- Learning Social Representations

- DeepWalk's output representation want to satisfy the followings
  - Adaptability : New social relations should not require repeating learning process all over again > Online
  - Community aware : The distance b/w of representation dimension should represent a metric for evaluating social similarity b/w the corresponding members of the network
  - Low dimensional : Low-dimensional models generalize better, speed up convergence & inference, if labeled data is scarce
  - Continuous : Continuous representation has smooth decision boundaries, which allows more robust classification

Language Modeling

- Brief explanations

- Language Modeling
  - Goal : 'Estimate the likelihood' of a specific seq. of words appearing in a corpus
  - Further Goal : Building general representations of words

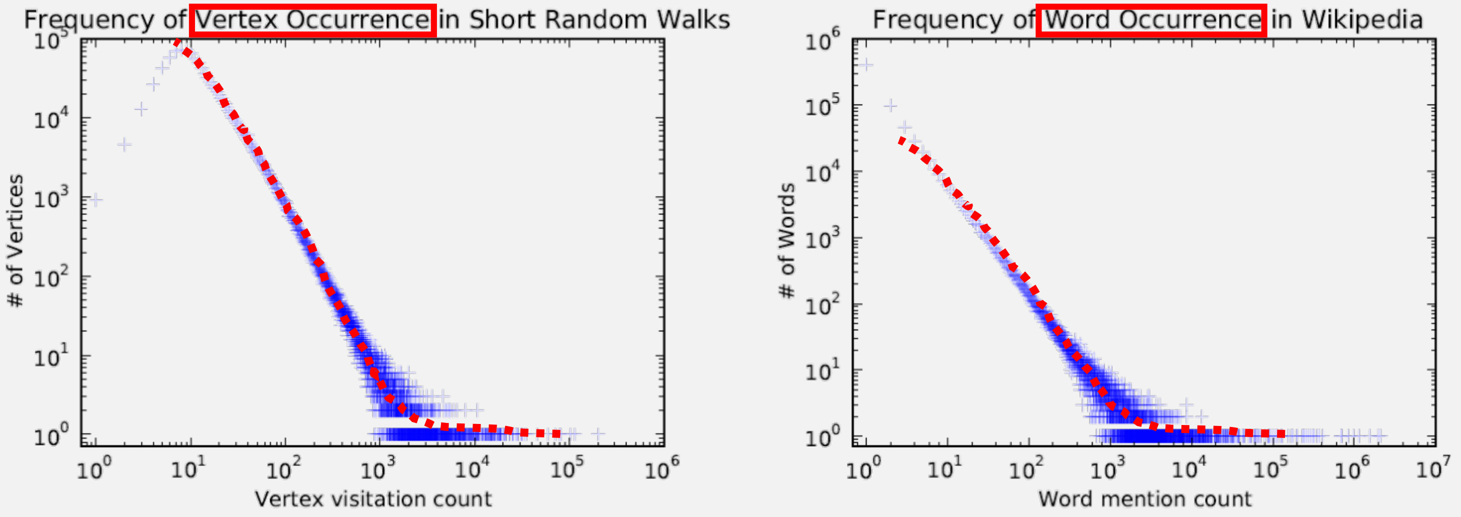
Language Modeling

- Why we apply language modeling method on DeepWalk?

- 'Distribution similarity' b/w vertex frequency & word frequency
  - Power Law : functional relationship b/w two quantities; one quantity varies as a power of another (ex. 80/20 rule)
  - Vertex frequency in the short random walks follows 'power-law' distribution (with some assumption)
  - Word frequency in natural language follows a similar, 'power-law' distribution
- Cool Idea : "Short Random Walks = Sentences"
  - In DeepWalk, it presents 'generalization' of language modeling

Language Modeling

- Why we apply language modeling method on DeepWalk?



(a) YouTube Social Graph (b) Wikipedia Article Text

Quite Similar!

Figure 2: The power-law distribution of vertices appearing in short random walks (2a) follows a power-law, much like the distribution of words in natural language (2b).

Language Modeling

- Further explanations

- In Language Modeling, We have to maximizing $\mathbb{P}(v_t | (v_{t-1}, \dots, v_{t-n+1}))$
  - Maximizing the next(target) word(vertex) v_t given context $(v_{t-1}, \dots, v_{t-n+1})$ of previous vertices
- But, we need to learn a 'Latent Representation'; not only prob. of co-occurrences
  - Learning representation by learning a mapping $\Phi : v \in V \mapsto \mathbb{R}^{|V| \times d}$
  - In practice, we consider Φ as an linear transformation b/w $\mathbb{R}^{|V|}$ and $\mathbb{R}^{d}$, serving as X_E
  - Then we have to maximize $\mathbb{P}(v_t | (\Phi(v_{t-1}), \dots, \Phi(v_{t-n+1})))$ -> infeasible as walk length grows

Language Modeling

- Further explanations

- We can change this prediction problem with relaxations
  - Using one word to predict the context, Context consist of not only the left-side words but also right-side words, Removing ordering constraint
  - Then we have optimization problem as $\underset{\Phi}{\text{minimize}} - \log \mathbb{P}(\{v_{i-w}, \dots, v_{i-1}, v_{i+1}, \dots, v_{v+w}\} \mid \Phi(v_t))$
- These relaxations are desirable for social representation learning
  - Order Independence captures 'nearness' provided by random walks
  - They also help speeding up training time by building small models for one vertex given at a time
- By solving above optimization problem, we can get representation that captures similarities in local graph structures
  - Vertices has similar 'neighborhoods' will acquire similar representations

Method

- Outline of DeepWalk Algorithm

Algorithm 1 DeepWalk(G, w, d, γ, t)

Input: graph $G(V, E)$

window size w
embedding size d
walks per vertex γ
walk length t

Hyperparameters

Output: matrix of vertex representations $\Phi \in \mathbb{R}^{|V| \times d}$

Initialization: Sample $\boxed{\Phi}$ from $\mathcal{U}^{|V| \times d}$
Build a binary Tree $\boxed{T}$ from V
for $i = 0$ to γ **do**
    $\mathcal{O} = \mathrm{Shuffle}(V)$
    for each $v_i \in \mathcal{O}$ **do**
        $\mathcal{W}_{v_i} = RandomWalk(G, v_i, t)$
        $\mathrm{SkipGram}(\Phi, \mathcal{W}_{v_i}, w)$
    end for
end for

Φ, T : Parameters

- Process
  1. Graph as an input
  2. Sampling γ Random Walks of length t for each nodes
  3. Updating Representation
- Additionals
  - Building Tree to use hierarchical softmax; 'tree' is parameter for this process
  - Shuffling makes SGD converge faster
  - Optimizing by SGD
  - Random Walk samples neighbor uniformly.

Method

- Applied Language Model : SkipGram

- SkipGram
  - One of the Word2Vec models, it maximizes co-occurrence probability among the words within a window in a sentence
  - Calculating posterior in SkipGram needs $O(|V|)$ operations; to avoids this, we adopts hierarchical model

Algorithm 2 SkipGram(Φ, $\mathcal{W}_{v_i}$, w)

1: **for each** $v_j \in \mathcal{W}_{v_i}$ **do**
2:     **for each** $u_k \in \mathcal{W}_{v_i}[j - w : j + w]$ **do**
3:         $J(\Phi) = -\log \Pr(u_k \mid \Phi(v_j))$
4:         $\Phi = \Phi - \alpha * \frac{\partial J}{\partial \Phi}$
5:     **end for**
6: **end for**

Method

- The reason for adopting Hierarchical Softmax

- Linear time complexity is too slow
  - If we have really large dataset, it is too long for training
  - By hierarchical decomposition, we can get 'exponential' speed-up
- Idea
  - Instead of computing directly $\mathbb{P}(Y\,|X)$, using simple truth $\mathbb{P}(Y|X)$ = $\mathbb{P}(Y|C,X)\mathbb{P}(C|X)$ & by defining clustering partition of Y into word classes $C = c(Y)$
  - $c(Y)$ can be any function. But to get more better generalization, we have to choice it that is easier to learn $\mathbb{P}(C = c(Y)|X)$

Method

- The reason for adopting Hierarchical Softmax

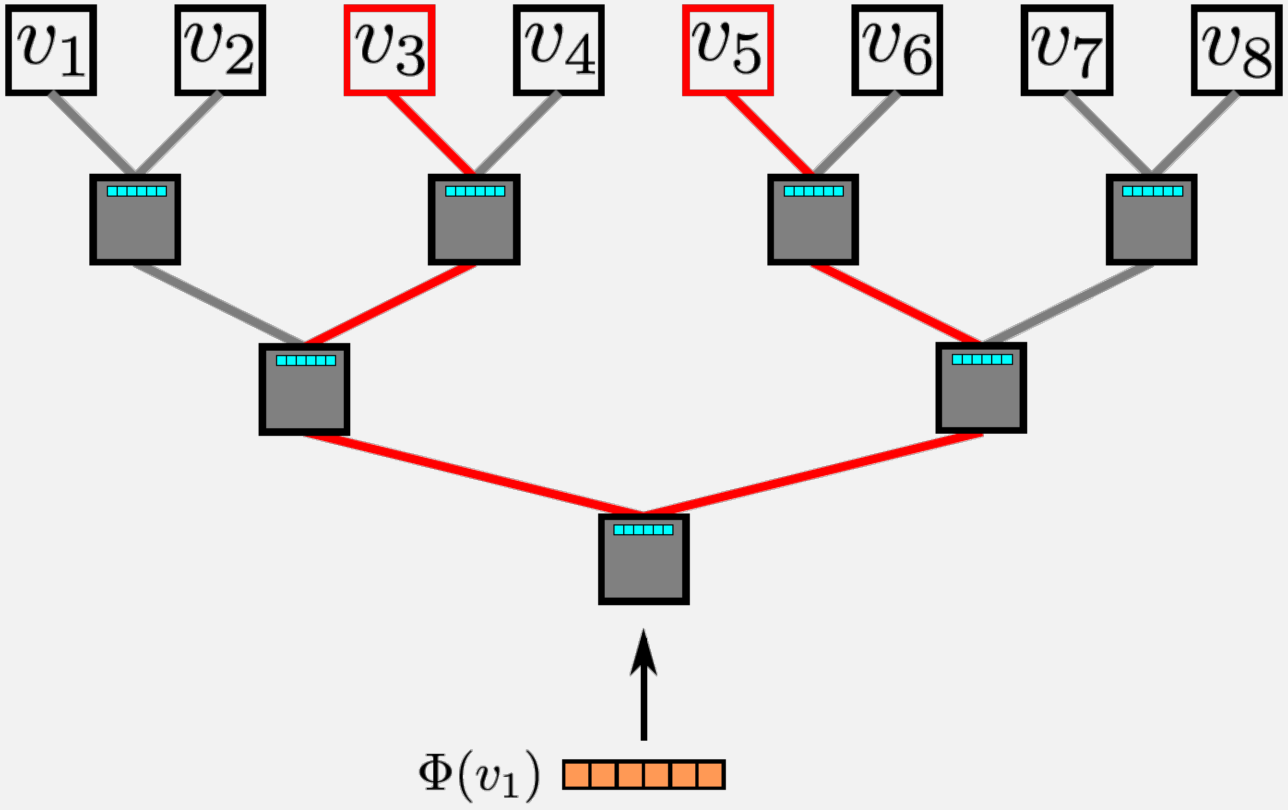
- Here the binary tree decomposition comes
  - By expressing the word w as a bit vector $\left(b_1(w), \ldots, b_m(w)\right)$, the next-word conditional probability can be computed as;

$$\mathbb{P}(w|(w_{t-1}, \ldots, w_{t-n+1})) = \prod_{i}^{m} \mathbb{P}(b_i(w)|b_1(w), \ldots, b_{i-1}(w), w_{t-1}, \ldots, w_{t-n+1})$$

  - By this, we only need some $O(\log|V|)$ operations rather than $O(|V|)$ operations

Method

- Hierarchical Softmax : approximating co-occurrence probability



(c) Hierarchical Softmax.

- Hierarchical Softmax
  - Now the Probability $\mathbb{P}(u|(\Phi(v_j))$ can be modeled by sequence of binary stochastic decisions
  - $\mathbb{P}(u|(\Phi(v_j)) = \prod_i^{\lceil \log|V| \rceil} \mathbb{P}(b_i|\Phi(v_j))$
  - Usage of Huffman coding can speed up the this process further

Experiments

- Experimental Design

- Datasets
  - BlogCatalog
    - Network of social relationships; Labels represent the topic categories provided by blogger authors
    - # of vertices = 10,312 , # of edges = 333,983 , # of labels = 39
  - Flickr
    - Network of contacts b/w users of the photo sharing website; Labels represent interest groups of users
    - # of vertices = 80,513 , # of edges = 5,899,882 , # of labels = 195
  - YouTube
    - Network b/w users of the 'popular' video sharing website; Labels represent groups of viewers enjoying common video genres
    - # of vertices = 1,138,499 , # of edges = 2,990,443 , # of labels = 47

Experiments

- Experimental Design

- Baseline Methods – the competitors
  - Catches Latent Dimensions
    - Spectral Clustering : based on eigenvectors of normalized graph Laplacian
    - (Max)Modularity : based on eigenvectors of modularity matrix
    - EdgeCluster : based on k-means clustering to cluster adjacency matrix; has advantage of scalability compared to above 2 methods.
  - Approximate Inference Technique
    - Weighted vote Relational Neighbor(wvRN) : relational classifier
- Experiment task : Multi-Label Classification
  - Training data : randomly sampled 'portion' of labeled vertices, Test data : Rest vertices
  - Model : One-vs-Rest logistic regression
  - Hyperparameter setting : DeepWalk with (γ = 80, w = 10, d = 128), Others with d = 500

Experiments

- Results

	% Labeled Nodes	10%	20%	30%	40%	50%	60%	70%	80%	90%
Micro-F1(%)	DeepWalk	**36.00**	**38.20**	**39.60**	**40.30**	**41.00**	**41.30**	41.50	41.50	42.00
	SpectralClustering	31.06	34.95	37.27	38.93	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
	wvRN	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
Macro-F1(%)	DeepWalk	**21.30**	**23.80**	25.30	26.30	27.30	27.60	27.90	28.20	28.90
	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	22.00	23.00	23.64	23.82	24.61	24.92
	Modularity	17.36	20.00	20.80	21.85	22.65	23.41	23.89	24.20	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 2: Multi-label classification results in BlogCatalog

Experiments

- Results : More sparse setting

	% Labeled Nodes	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
Micro-F1(%)	DeepWalk	**32.4**	**34.6**	**35.9**	**36.7**	**37.2**	**37.7**	**38.1**	**38.3**	**38.5**	**38.7**
	SpectralClustering	27.43	30.11	31.63	32.69	33.31	33.95	34.46	34.81	35.14	35.41
	EdgeCluster	25.75	28.53	29.14	30.31	30.85	31.53	31.75	31.76	32.19	32.84
	Modularity	22.75	25.29	27.3	27.6	28.05	29.33	29.43	28.89	29.17	29.2
	wvRN	17.7	14.43	15.72	20.97	19.83	19.42	19.22	21.25	22.51	22.73
	Majority	16.34	16.31	16.34	16.46	16.65	16.44	16.38	16.62	16.67	16.71
Macro-F1(%)	DeepWalk	**14.0**	17.3	**19.6**	**21.1**	**22.1**	**22.9**	**23.6**	**24.1**	**24.6**	**25.0**
	SpectralClustering	13.84	**17.49**	19.44	20.75	21.60	22.36	23.01	23.36	23.82	24.05
	EdgeCluster	10.52	14.10	15.91	16.72	18.01	18.54	19.54	20.18	20.78	20.85
	Modularity	10.21	13.37	15.24	15.11	16.14	16.64	17.02	17.1	17.14	17.12
	wvRN	1.53	2.46	2.91	3.47	4.95	5.56	5.82	6.59	8.00	7.26
	Majority	0.45	0.44	0.45	0.46	0.47	0.44	0.45	0.47	0.47	0.47

Table 3: Multi-label classification results in Flickr

Experiments

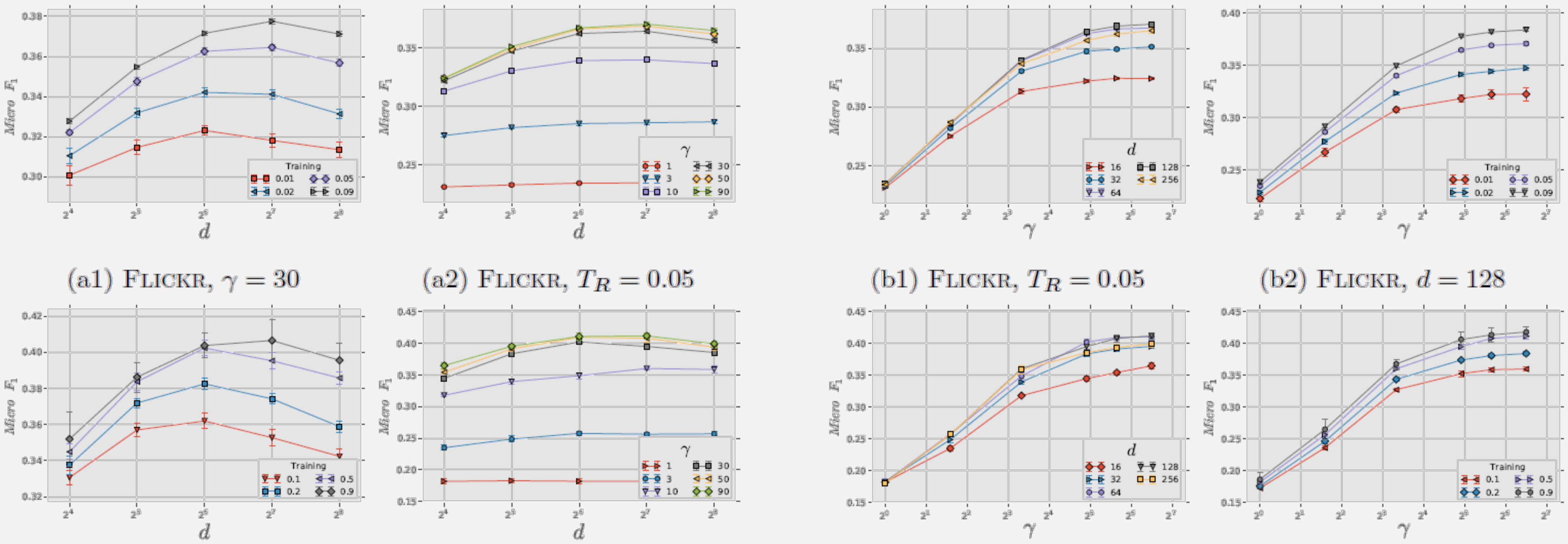
- Results : More sparse setting

	% Labeled Nodes	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
Micro-F1(%)	DeepWalk	**37.95**	**39.28**	**40.08**	**40.78**	**41.32**	**41.72**	**42.12**	**42.48**	**42.78**	**43.05**
	SpectralClustering	—	—	—	—	—	—	—	—	—	—
	EdgeCluster	23.90	31.68	35.53	36.76	37.81	38.63	38.94	39.46	39.92	40.07
	Modularity	—	—	—	—	—	—	—	—	—	—
	wvRN	26.79	29.18	33.1	32.88	35.76	37.38	38.21	37.75	38.68	39.42
	Majority	24.90	24.84	25.25	25.23	25.22	25.33	25.31	25.34	25.38	25.38
Macro-F1(%)	DeepWalk	**29.22**	**31.83**	**33.06**	**33.90**	**34.35**	**34.66**	**34.96**	**35.22**	**35.42**	**35.67**
	SpectralClustering	—	—	—	—	—	—	—	—	—	—
	EdgeCluster	19.48	25.01	28.15	29.17	29.82	30.65	30.75	31.23	31.45	31.54
	Modularity	—	—	—	—	—	—	—	—	—	—
	wvRN	13.15	15.78	19.66	20.9	23.31	25.43	27.08	26.48	28.33	28.89
	Majority	6.12	5.86	6.21	6.1	6.07	6.19	6.17	6.16	6.18	6.19

Table 4: Multi-label classification results in YouTube

Experiments

- Results : Parameter sensitivity



(a1) FLICKR, $\gamma = 30$ (a2) FLICKR, $T_R = 0.05$ (b1) FLICKR, $T_R = 0.05$ (b2) FLICKR, $d = 128$

(a3) BLOGCATALOG, $\gamma = 30$ (a4) BLOGCATALOG, $T_R = 0.5$ (b3) BLOGCATALOG, $T_R = 0.5$ (b4) BLOGCATALOG, $d = 128$

(a) Stability over dimensions, d

(a) Stability over number of walks, γ

Figure 5: Parameter Sensitivity Study

With fixed w, t

Conclusions

- Main differences b/w DeepWalk and previous Related Works

- DeepWalk learns 'latent social representations'
  - Instead of computing statistics related to centrality or partitioning
- Do not attempt to extend the classification process itself
- Scalable & online method that uses local information
  - While most other methods require global information or offline
- Unsupervised representation learning of graphs

Conclusions

- Conclusion

- DeepWalk is Scalable & Online algorithm
  - Experiments show that DeepWalk can create meaningful representations of graph that is too large to run spectral methods
  - It is also Pararellizable; can reduce time to update while maintaining its performance
- DeepWalk is appealing Generalization of language modeling
  - In this paper's view, language modeling can be considered as sampling from an unobservable language graph
- Summary
  - Language Modeling techniques can be used for online learning of network representations

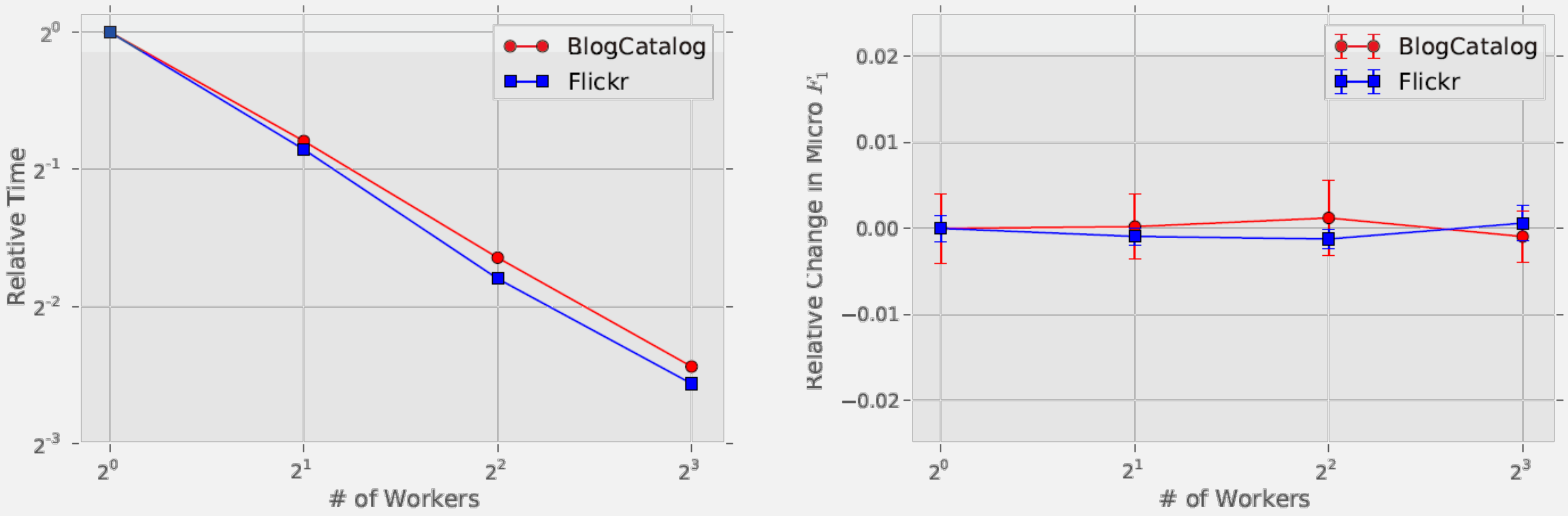
Further Talks

- Algorithm Variants

- Streaming approach
  - Don't have to know entire graph
  - In this variant, representation can be built & updated directly as new data(walk) comes in
  - Has 2 necessary modifications
    - Decaying learning rate cannot be used
    - Cannot necessarily build a binary tree parameter anymore; we can build hierarchical softmax tree if cardinality of V is known or can be bounded.
- Non-random walks
  - Many graphs are created as a by-product of agents interacting with seq. of elements
  - If the graph is created by such a stream of this 'non-random' walks, the walks can be passed into model directly

Further Talks

- Parallelizability



(a) Running Time

Decreases

(b) Performance

Remains constant

Figure 4: Effects of parallelizing DeepWalk

Impressions & Comments

- Some Impressions on this paper

- It's simple and efficient.
  - DeepWalk's approach is quite simple; it just treats random walks on the graph as a sentence, and applies language model to get latent representations.
  - This kind of 'generalization' of language modeling is quite impressive.
  - It is quite fast(because of the hierarchical softmax & parallelizability) and effective.
  - Experiments on variety graphs shows its effectiveness on multi-label classification tasks
- It's really 'fresh' in my opinion
  - Usage of random walk as a sentence.

Impressions & Comments

- Further Comments & Thoughts

- So they really achieved 4 properties they desired?
  - Adaptability : DeepWalk uses local information to catch latent structure. If new data(nodes) are added, we only have to sample walks from them, and update our parameters
  - Community aware : DeepWalk gets its representation by **maximizing co-occurrence** within the window
  - Low-dimensional : In experiments, we can see that DeepWalk's latent dimension is low compared to other method
  - Continuous : By **imbedding** its representation **in the space that has many good properties**

Impressions & Comments

- Further Comments & Thoughts

- Remained Questions
  - Exact usage of 'Huffman Coding' in the algorithm : this question maybe solved by learning more about Word/Node imbedding.
  - 'Continuous' space?
  - The usage/meaning of the 'sparsity'
- Further Thoughts
  - I think it might be more better algorithm that concerning more features about the nodes; maybe node features have some (local) structural information

Thank you!

Reference

- A. Mnih and G. E. Hinton. A scalable hierarchical distributed language model. *Advances in neural information processing systems,* 21:1081–1088, 2009.
- F. Morin and Y. Bengio. Hierarchical probabilistic neural network language model. In *Proceedings of the international workshop on artificial intelligence and statistics*, pages 246–252, 2005.