DeepWalk

Online Learning of Social Representations

2021 Winter Internship

Daeyoung Kim

2021.12.28.

INDEX

Abstract

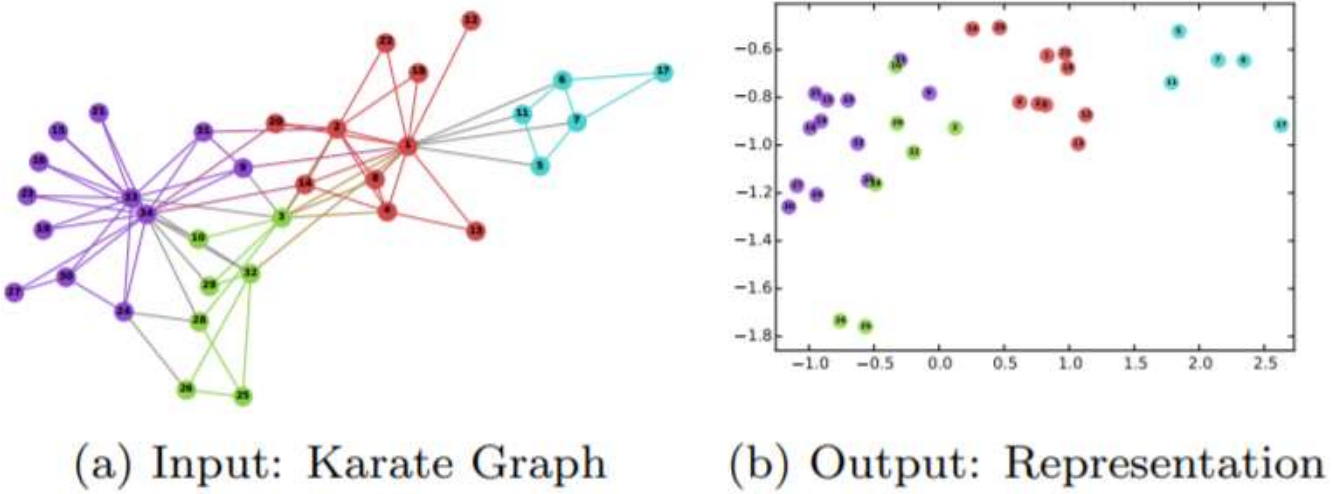
network에서 꼭짓점들의 **잠재 표현**을 학습하는 기법

꼭짓점 간의 관계를 **저차원**의 벡터 공간에 나타냄

Random Walk를 기반으로 학습

확장성과 쉬운 병렬화로 real-world application에 적합함

Introduction



(a) Input: Karate Graph (b) Output: Representation

Graph를 **2**차원 공간에 나타내도, 형태가 잘 유지된다

(실제로 가까운 node들이 2차원 공간에서도 가까이 위치)

Introduction : DeepWalk

- Deep learning 자연어 처리 기법을 network analysis에 **처음**으로 적용
- Input : Graph, Output : latent representation
- Learns **social representations** of a graph's vertices
  - Capture neighborhood similarity and community membership

Contribution

- DeepWalk learns **structural** regularities present within short random walks
- Evaluate representations on multi-label classification tasks
  on social networks
- Demonstrate the **scalability** of algorithm
  - Building representations of web-scaled graphs using a parallel implementation

Problem Definition

Classifying members of a social network into one or more categories

Input

$G = (V, E)$: social network

Where V : members of the network

$E \subseteq (V \times V)$: edges

Goal : learn $X_E \in \mathbb{R}^{|V| \times d}$

d : **small number** of latent dimensions

DeepWalk classified as…

Unsupervised method

Capture the graph structure ***independent*** of the labels' distribution

Characteristics

- Adaptability

  새로운 관계가 추가될 때 전체 학습 과정을 반복해서는 안 된다.

- Community aware

  latent space에서의 distance는 network에서 member 간의 사회적 유사성을 평가할 수 있어야 한다.

- Low dimensional

  저차원 모델로 표현해야 일반화가 더 잘 되며 속도도 빨라진다.

- Continuous

  smooth decision boundary를 가져서 robust한 classification이 이루어지게 한다.

To satisfy characteristics…

Use optimization techniques for **language modeling**

Input**: Stream of short random walks** for vertices

Random Walks

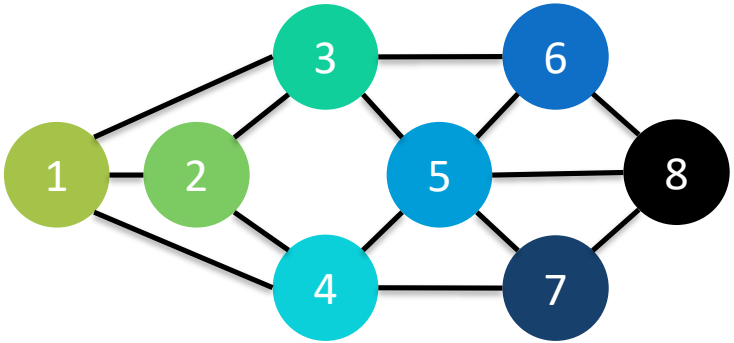
- 한 꼭짓점에서 시작해 연결된 이웃한 꼭짓점들로 확률적으로 이동
  이를 일정 횟수 반복해 거쳐간 꼭짓점들을 sequence로 나타낸 것

- W_{v_i} : random walk rooted at vertex v_i

- 각 꼭짓점으로 이동할 확률은 같다

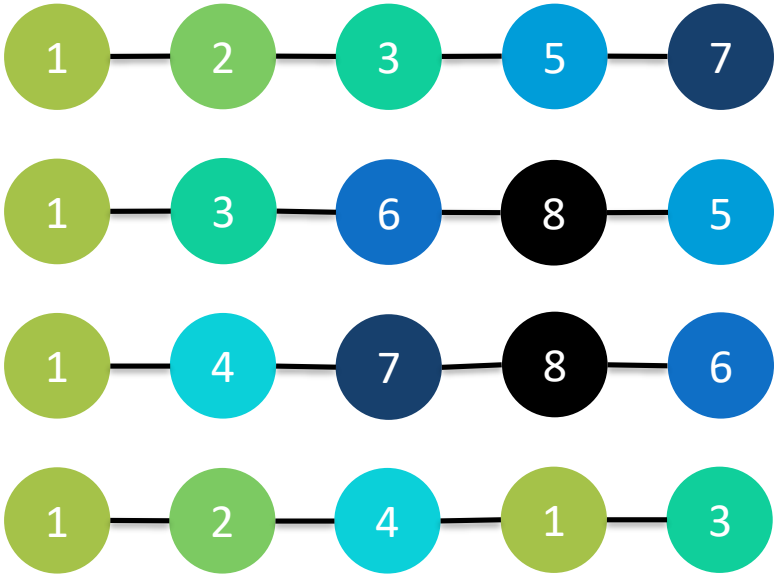
- Used as a similarity measure for content recommendation, community detection

Random Walks



Graph

Ex) Short random walks rooted at vertex 1

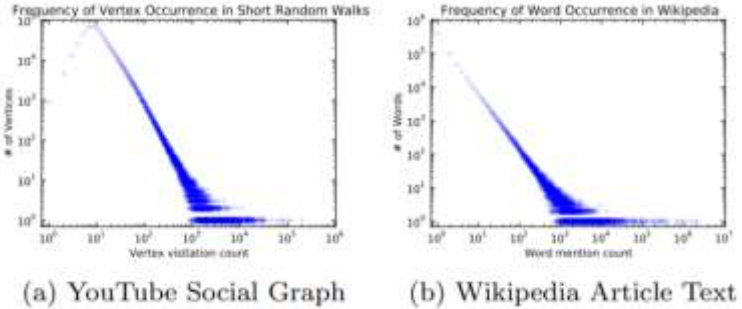


Random Walks : Advantages

- Local exploration is easy to parallelize
  : 여러 개의 random walk이 **동시에** graph의 여러 부분들을 확인

- 전체 재계산 없이 graph의 변화가 이뤄질 수 있다
  : 새로운 random walk을 사용해 학습된 모델을 반복해서 update

Power law



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

Graph의 **차수** 분포가 power law를 따르면, random walk에서 **꼭짓점 등장 빈도**의 분포도 power law를 따른다 (Zipf's Law)

Language Modeling에서 word 등장 횟수 분포와 random walk에서 vertex 등장 횟수 분포가 유사

Network Analysis에서 Power Law 적용의 의의 제공

Language Modeling

- In language modeling, given $W_1^n = (\omega_0,\ \omega_1,\ \cdots, \omega_n)$, $\omega_i \in V$ (vocabulary)
  Goal : maximize $\Pr(\omega_n \mid \omega_0,\ \omega_1,\ \cdots, \omega_{n-1}))$

- In DeepWalk, use short **random walk**s as sentences
  Goal : maximize $\Pr(v_i \mid (v_1,\ v_2,\ \cdots,\ v_{i-1}))$ (v_i : observing vertex)

- The purpose is learning a latent representation,
  let $\Phi : v \in V \ \rightarrow\ \mathbb{R}^{|V| \times d}$ (map latent representation associated with vertex v)
  Goal : maximize $\Pr(v_i \mid (\Phi(v_1),\ \Phi(v_2),\ \cdots,\ \Phi(v_{i-1}))$

Language Modeling

- To decrease computational cost, relaxation is required

  - uses **one word** to predict the **context**

  - context is composed of the words appearing to **both side** of **given word**

  - removes ordering constraint

  -> maximize the probability of **any word** appearing in the context

  without the knowledge of its offset from the given word

Goal : $\underset{\Phi}{\text{minimize}} -\log \Pr(\{v_{i-\omega}, \cdots, v_{i-1}, v_{i+1}, \cdots, v_{i+\omega}\}|\Phi(v_i))$

- Vertices which have similar neighborhoods will acquire similar representations

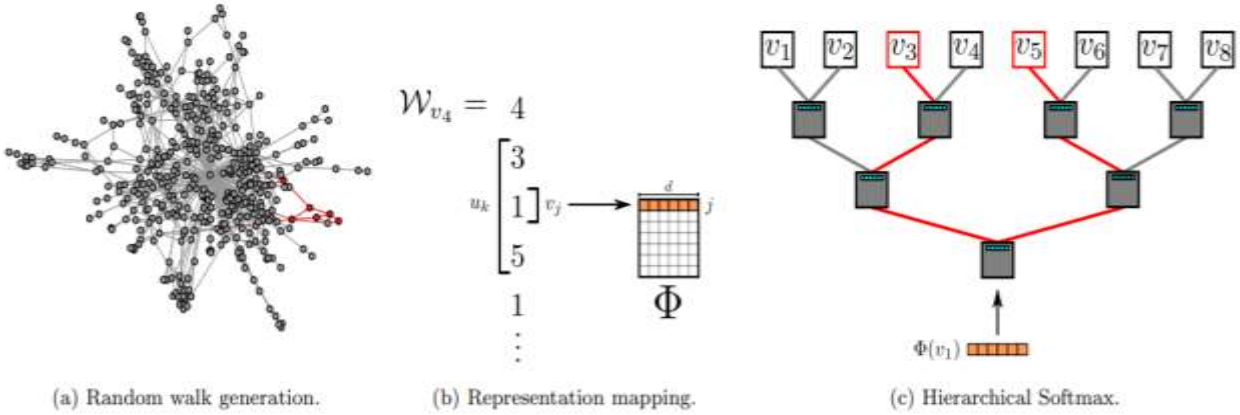
Overview : DeepWalk

Steps

1. Random Walk Generation

2. Mapping

3. Optimization



(a) Random walk generation. (b) Representation mapping. (c) Hierarchical Softmax.

Algorithm : DeepWalk

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

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

Input	설명
$G(V,E)$	그래프(꼭짓점, 간선)
ω	목표 앞뒤로 참조하는 꼭짓점 개수
d	Latent space의 차원
γ	꼭짓점마다 생성되는 Random walks 개수
t	Random walk의 길이
Φ	꼭짓점들을 latent space에 mapping하는 함수

Algorithm : DeepWalk

Algorithm 1 DeepWalk(G, w, d, γ, t)

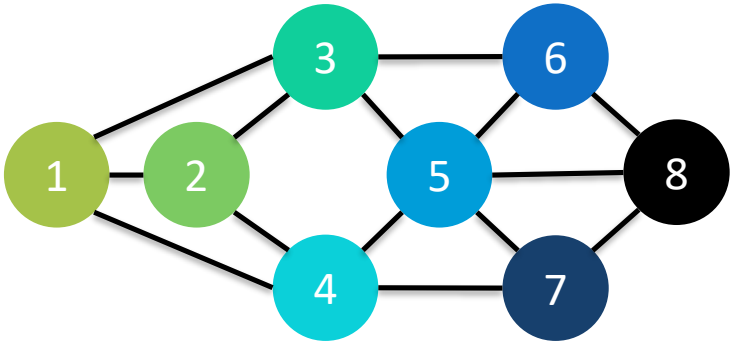
Input: graph $G(V, E)$
window size w
embedding size d
walks per vertex γ
walk length t
Output: matrix of vertex representations $\Phi \in \mathbb{R}^{|V| \times d}$

1: Initialization: Sample Φ from $\mathcal{U}^{|V| \times d}$
2: Build a binary Tree T from V
3: **for** $i = 0$ to γ **do**
4: $\mathcal{O} = \text{Shuffle}(V)$
5: **for each** $v_i \in \mathcal{O}$ **do**
6: $\mathcal{W}_{v_i} = RandomWalk(G, v_i, t)$
7: SkipGram(Φ, $\mathcal{W}_{v_i}$, w)
8: **end for**
9: **end for**

1. Graph를 input으로 받음
2. 반복 횟수 설정(walks per vertax)
3. 전체 노드를 무작위로 섞고,
   섞인 순서대로 학습 진행

- Inner loop
  - 각 노드에 대해 길이 t의 random walk 생성
  - SkipGram으로 최적화

Algorithm : SkipGram



Graph

중심 꼭짓점과 이와 인접한 꼭짓점들이
Random Walk에서 동시에 나타날
확률을 최대화

Algorithm : SkipGram

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**

각 꼭짓점에서 window size 안에 나타나는 모든 random walk에 대해 최적화 계산

속도 향상을 위해 확률 분포를 근사하는 방법을 적용

Hierarchical Softmax

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**

$$J(\Phi) = -\log \Pr\left(u_k \middle| \Phi\left(v_j\right)\right)$$

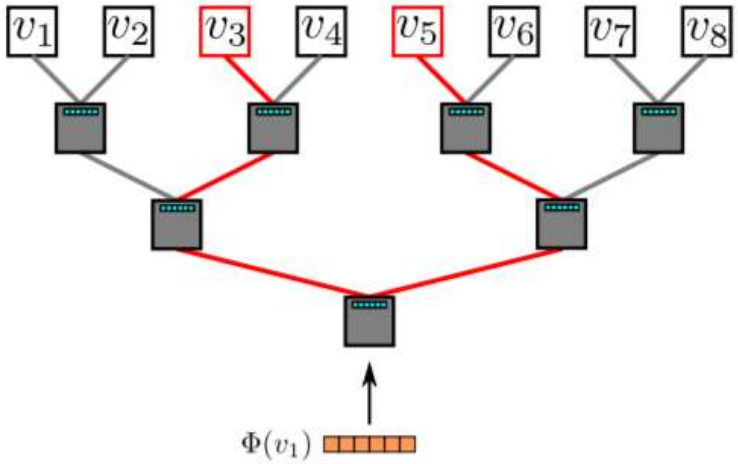
$\Pr\left(u_k \middle| \Phi\left(v_j\right)\right)$을 전부 계산하면 computational cost가 너무 크다

Binary Classification으로 문제를 바꿔서 복잡도를 크게 줄인다

Idea.

graph의 꼭짓점을 binary tree의 leaf node로 대응시키면, 특정 경로로 향하는 확률을 최대화하는 문제로 바뀜

Hierarchical Softmax



(c) Hierarchical Softmax.

$$\Pr(u_k|\Phi(v_j)) = \prod_{l=1}^{\lceil \log |V| \rceil} \Pr(b_l \mid \Phi(v_j))$$

v_i : leaf node, b_l : node of binary tree

계산 복잡도가 $O(|V|)$에서 $O(\log|V|)$로 감소

Optimization

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

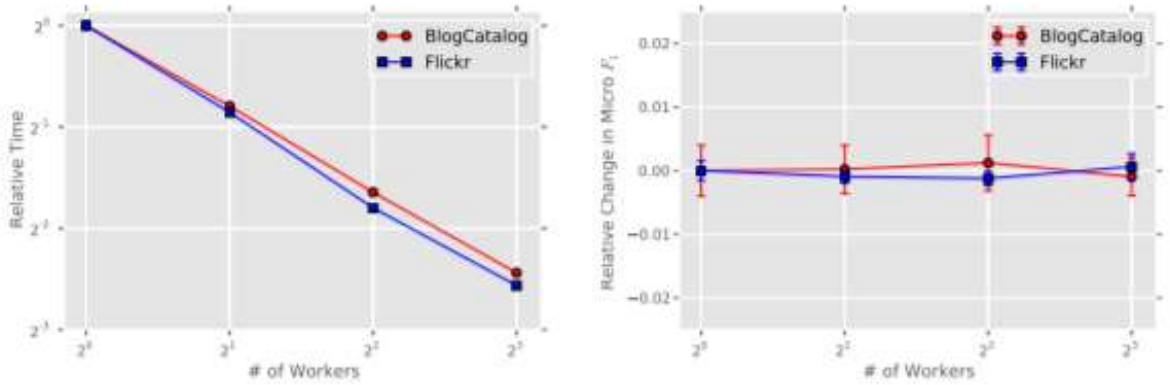
1: **for each** $v_j \in \mathcal{W}_{v_i}$ **do**
2: **for each** $u_k \in \mathcal{W}_{v_i}[j-w : j+w]$ **do**
3: $J(\Phi) = -\log \Pr(u_k \mid \Phi(v_j))$
4: $\Phi = \Phi - \alpha * \frac{\partial J}{\partial \Phi}$
5: **end for**
6: **end for**

$$\Phi = \Phi - \alpha * \frac{\partial J}{\partial \Phi}$$

Parameter : $\{\Phi, T\}$

SGD 적용해서 parameter 최적화

Parallelizability



(a) Running Time (b) Performance

Figure 4: Effects of parallelizing DEEPWALK

- worker의 수를 늘리면 **수행 시간** 감소, performance 유지
- Large scale에도 적용 가능

Datasets

Name	BlogCatalog	Flickr	YouTube
$\|V\|$	10,312	80,513	1,138,499
$\|E\|$	333,983	5,899,882	2,990,443
$\|\mathcal{Y}\|$	39	195	47
Labels	Interests	Groups	Groups

Table 1: Graphs used in our experiments.

Use Social Large-Scale Networks as Datasets

$|V|$: number of nodes

$|E|$: number of links

$|y|$: number of categories

Experiments : Multi-Label Classification

DeepWalk

- number of random walks(τ) = 10
- walk length(t) = 80
- window size(ω) = 10
- embedding size(d) = 128
- portion of labeled nodes(T_R)
  - BlogCatalog : 10%~90%
  - Flickr, YouTube : 1%~10%
- One-vs-rest logistic regression for classification method

Baseline Methods

SpectralClustering : d-smallest eigenvectors

Modularity : top-d eigenvectors

EdgeCluster : k-means clustering

wvRN : relational classifier

Majority : choose most frequent labels

dimensionality = 500

Results : BLOGCATALOG

	% 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

Results : FLICKR

	% 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

Results : YOUTUBE

	% 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

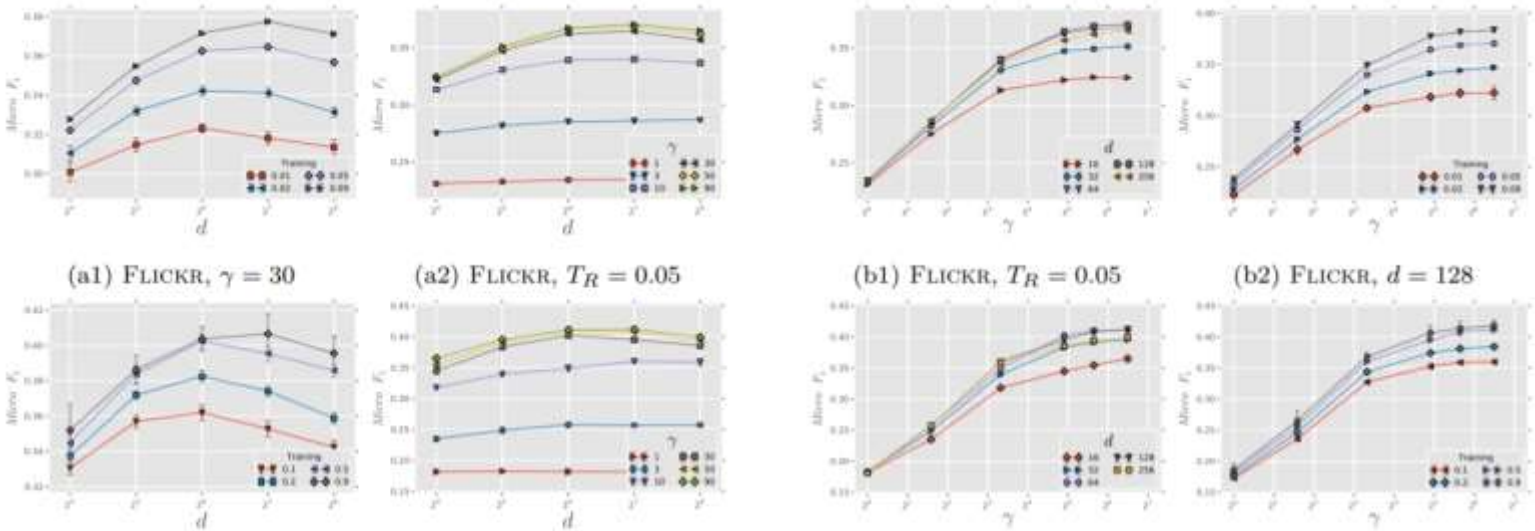
Experiments : Parameter Sensitivity

Hyperparameter가 performance에 미치는 영향 확인

Walk length = 40, window size = 10 (fixed)
γ(node당 Random Walk 개수), d(임베딩 차원), T_R(train data의 비율) 을 조정

Dataset: FLICKR, BLOGCATALOG

Results : Effect of Dimensionality



(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

Conclusions

- ✓ Random walk을 input으로 사용해 그래프의 잠재 표현을 발견
- ✓ 확장성으로 인해서 복잡한 그래프에서도 학습 결과가 유의미함
- ✓ 병렬화로 수행 시간을 크게 단축
- ✓ Language modeling과의 동반 발전이 기대됨

Algorithm Variants : DeepWalk

- Streaming

  - implemented **without** knowledge of the entire graph

  - when walks come, learning, updating model works directly

  - Necessary modifications

    1. decaying learning rate is impossible

    2. can’t necessarily build a tree of parameters anymore

    -> If cardinality of V is known of can be bounded, build Hierarchical Softmax tree

- Non-random walks

  - some graphs are created as a by-product of agents interacting with a sequence of elements(e.g. users’ navigation of pages on website)

  - available to feed the modeling phase directly

  - combined with streaming to train features on a continually evolving network

Limitations : DeepWalk

- Random walk로 그래프 구조를 학습하므로, random walk의 무작위성으로 인해서 전체 구조를 파악하기 어렵다

- 서로 멀리 떨어진 꼭짓점들이 유사성을 가질 경우 이를 표현하기 어렵다

node2vec

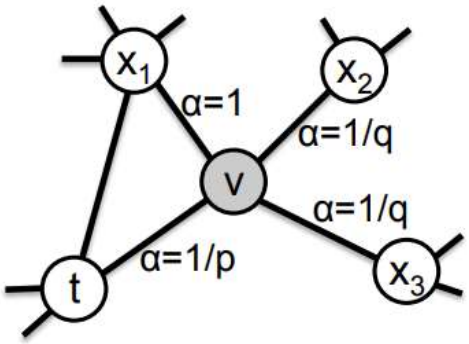


Figure 2: Illustration of the random walk procedure in *node2vec*. The walk just transitioned from t to v and is now evaluating its next step out of node v. Edge labels indicate search biases α.

Deepwalk에서는 인접 꼭짓점으로 이동할 확률이 **같다**

이동 확률에 가중치가 부여된다면?

Paper
Node2vec: Scalable Feature Learning for Networks

7. Further discussion

THANK YOU